Sviluppo e Implementazione di un Sistema per il Rilevamento di Frodi Bancarie

Stefano Angaran

Dipartimento di Ingegneria dell'Informazione

Università degli Studi di Padova

Corso di Laurea Magistrale in Ingegneria Informatica

12 Marzo 2013

Abstract

Garantire la sicurezza e l'integrità delle transazioni operate attraverso un portale di home banking è diventata una sfida per gli istituti finanziari. La crescente sofisticazione degli attacchi informatici, l'evoluzione di una vera e propria economia sotterranea del cybercrimine e il cambiamento repentino delle abitudini degli utenti hanno messo in crisi le attuali soluzioni di fraud-prevention. È perciò necessario integrare un ulteriore livello di protezione, costituito da un sistema di fraud-detection, in grado di mettere a frutto le informazioni e i dati storici a disposizione delle banche ed elaborarne una descrizione dei clienti. Questa tesi analizza lo scenario attuale delle frodi bancarie, concentrandosi sulla minaccia più subdola; gli attacchi Man-in-the-Browser. Viene descritto un sistema, basato su tecniche mutuate dall'anomaly detection, per l'elaborazione di profili descrittivi del comportamento di spesa e di navigazione degli utenti di servizi di home banking. Il sistema è stato implementato a livello di prototipo dimostrando risultati molto incoraggianti e la capacità di individuazione di nuovi episodi fraudolenti.

Ai miei genitori e alla mia famiglia.

Ringraziamenti

Queste pagine stampate non sono solo una tesi. Sanciscono, per me, la conclusione di un lungo, a volte tortuoso, percorso di studi. Per mia fortuna lungo questa strada non sono mai stato solo. Ad accompagnarmi, lungo di essa, c'era la mia famiglia, che mi sempre ha sostenuto e non solo economicamente. Dedico loro la mia tesi e a loro va la mia più grande gratitudine e il merito se oggi posso presentare questo lavoro, se oggi sono quello che sono. Ringrazio i miei nonni, che nonostante tutto hanno sempre creduto in me. A Raffaella un grazie di cuore, senza di lei il resto non avrebbe senso; con la sua grinta e il suo ottimismo mi ha aiutato a superare più di un momento difficile. Ringrazio gli amici, tutti. I giovani del Collegio Domenico Savio di Padova e il direttore, Don Antonio, un uomo che con il suo impegno e la sua forza ha conquistato anche me, sebbene distante per molti versi; per tutta la vita conserverò un bellissimo ricordo di quegli anni durante i quali sono cresciuto, soprattutto, come persona. E i ragazzi del PAPP, voi sapete chi siete. Ringrazio i colleghi, mi sento di definirli così, di SEC Servizi, in particolare l'Ing. Zandolin, che mi ha seguito lungo lo sviluppo di questa tesi con grande disponibilità e l'Ing. Zuppa, per avere riposto fiducia in me permettendomi di lavorare a questo progetto. Ringrazio il mio relatore, il Prof. Pietracaprina, per i suoi consigli e il suo supporto. Vorrei ringraziare anche K. L. Ingham, per il suo prezioso contributo e la sua implementazione di uno degli algoritmi utilizzati in questa tesi. Grazie a tutti voi, e anche a tutti gli altri, che non si nega a nessuno, un grazie, specie in momenti come questi. E così non dimentico qualcuno.

Indice

Elenco delle figure

ELENCO DELLE FIGURE

Elenco delle tabelle

1

Introduzione

All'inizio degli anni '80 gli istituti finanziari degli Stati Uniti lanciarono i primi progetti pilota per consentire ai propri clienti di usufruire dei loro servizi anche da casa: nasceva così il concetto di *home banking*. I primi prototipi, ancora piuttosto limitati, utilizzavano la tecnologia *videotex*, una sorta di videoterminale operante tramite collegamento telefonico. La loro accoglienza sul mercato non fu però affatto favorevole, decretandone il fallimento commerciale. Per il primo servizio di *home banking* attraverso Internet bisogna attendere fino alla fine del 1994, anno in cui la statunitense Stanford Federal Credit Union lancia il proprio nuovo portale web offrendo il servizio a tutti i clienti [106]. L'anno seguente, nel 1995, nasce la Security First Network Bank (SFNB), la prima banca a condurre le proprie operazioni solamente tramite il canale Internet [105], di fatto introducendo un vero e proprio nuovo modello di business: le *virtual banks*.

Oggi sono più di 70 milioni le famiglie, soltanto negli Stati Uniti ad usufruire dei servizi di *online banking*, l'80% del totale tra quelle con accesso ad Internet [10]. Un sondaggio del 2011 dell'American Bankers Association (ABA) ha evidenziato come il 62% degli intervistati preferisca utilizzare i servizi online quando possibile rispetto ad un valore del 36% dell'anno precedente [19]. Anche l'Europa è interessata a questa tendenza. Una ricerca della Deutsche Bank individua 4 principali aree geografiche con un diverso tasso di adozione [85]. La Scandinavia, assieme ad altri Stati del Nord Europa, fa parte delle zone in cui la fetta di utenti dei servizi online è maggiore, 62-77%. L'Europa Centrale (Germania, Francia, Regno Unito, ...) costituisce una seconda area con percentuali variabili tra il 35% e il 54%, valori paragonabili a quelli degli Stati Uniti (41%). La maggior parte degli Stati con un tasso inferiore al 32% appertengono alle aree

meridionali e orientali dell'Europa, ad eccezione dell'Irlanda. In quest'ultima categoria si colloca anche l'Italia con una percentuale indicata al 16%. Infine esiste un piccolo gruppo di stati, principalmente localizzato nella zona dei Balcani, dove gli strumenti di *home banking* non hanno ancora attecchito. La stessa ricerca evidenzia come la tendenza ad utilizzare i servizi bancari via Internet sia in costante ascesa: da meno del 20% dei cittadini Europei nel 2004 a oltre il 40% del 2012 [86], in previsione di raggiungere più del 60% entro il 2020.

La tecnologia alla base dei portali di *home banking* è ormai matura. La quasi totalità delle banche offre un sito Internet attraverso cui l'utente può effettuare tutta una serie di operazioni per le quali era in passato necessario ricorrere a canali alternativi (*call-center*, filiale fisica, ...). Tra i servizi comunemente messi a disposizione troviamo:

- visualizzazione del saldo e delle transazioni recenti
- visualizzazione e download di documenti
- prenotazione e situazione assegni
- giroconto
- trasferimento di fondi (e.g. bonifici bancari)
- *trading* online
- operazioni di ricarica di schede prepagate (e.g. ricariche celluari, servizi TV in abbonamento, ...)
- gestione del mutuo
- compilazione e pagamento del modello F24
- pagamenti MAV
- gestione dell'account

Sempre più numerose sono le banche che iniziano ad offrire i loro servizi anche per piattaforme *mobile* come smartphone e tablet, grazie alla diffusione di terminali sempre più "intelligenti" e di semplice utilizzo e di tariffe convenienti per l'accesso ad Internet.

Diverse sono le motivazioni che spingono gli utenti verso l'utilizzo dei servizi bancari online. Il fattore determinante, secondo una ricerca del 2002 [66] su un campione rappresentativo di consumatori statunintensi, è la possibilità di poter accedere alle proprie informazioni finanziarie 24 ore su 24. Segue il risparmio di tempo derivante dal non doversi recare fisicamente in filiale, un miglior controllo e gestione delle proprie finanze e una maggiore privacy (non è necessario comunicare con un operatore).

Oltre alla convenienza del consumatore occorre prendere in considerazione quella degli istituti finanziari. Una maggior diffusione dei servizi di *home banking* consente alle banche di ridurre i costi relativi al mantenimento delle strutture e alla retribuzione del personale di filiale. Come detto alcuni istituti sono andati oltre aprendo delle vere e proprie attività bancarie esclusivamente online mentre altri hanno inaugurato nuove divisioni "virtuali". La diminuzione dei costi non è l'unico *driver* che spinge lo sviluppo di soluzioni *home banking* sempre migliori. Il nuovo modello emergente di società centrata sull'informazione e sull'utilizzo di Internet sta contribuendo alla crescita delle domanda di servizi online con una richiesta di funzionalità sempre più ricche; venire incontro in maniera adeguata a queste aspettative è diventato quindi un importante vantaggio competitivo.

Le potenzialità di questo settore non hanno però attratto soltanto nuovi clienti. Cybercriminali e frodatori hanno ben presto rivolto la loro attenzione al mondo dell'*home banking* intravedendo una nuova opportunità di guadagnare denaro illegalmente. Una delle prime, e più note, tecniche adottate per compromettere la sicurezza di questi servizi online è il *phishing*. Si tratta di un tipo di frode online ideata con lo scopo di carpire con l'inganno informazioni quali username, password o numero di carta di credito dell'utente vittima. Attuata generalmente tramite e-mail si basa sull'invio da parte di un frodatore di messaggi che simulano la grafica e i colori di un'entità di fiducia (come può essere un istituto bancario) i quali richiedono all'ingenuo utente l'inserimento di informazioni personali. Un'evoluzione più sofisticata di questa tecnica è denominata *pharming* e consiste in un sovvertimento del normale funzionamento del servizio DNS[1], all'interno di una rete o più frequentemente a livello di sistema operativo. In questo modo l'ignaro utente che cerchi di visitare, ad esempio, un portale di home banking

[1] ad esempio attraverso una modifica del file *hosts* del computer della vittima o sfruttando una vulnerabilità nel software del server DNS

verrà redirетto verso una versione fittizia dello stesso, opportunamente sviluppata per ottenerne le credenziali.

Alternative a queste metodologie base di attacco prevedono l'installazione nel sistema operativo dell'utente di un *malware*, termine che deriva dalla contrazione di "malicious" e "software". Per malware si intende un qualsiasi programma progettato allo scopo di causare danni più o meno gravi ad un sistema informatico, sia direttamente che indirettamente. Ad esempio un software di questo tipo può monitorare l'attività online di un utente memorizzando i dati che vengono inseriti durante il processo di autenticazione e inviandoli successivamente, tramite Internet, al frodatore. Varianti più complesse sono in grado di eseguire una scansione automatica dei file personali dell'utente alla ricerca di informazioni sensibili ivi conservate.

Nel 2005, e successivamente nel 2011, il Federal Financial Institutions Examination Council (FFIEC), un organismo di coordinamento tra le varie agenzie di vigilanza bancaria per l'applicazione uniforme dei principi di sorveglianza sul sistema bancario e finanziario negli Stati Uniti, ha emanato una serie di linee guida [52; 53] per la sicurezza dei sistemi bancari online. Tra le varie misure indicate i documenti prevedono l'adozione di forme di autenticazione multi-fattore per contrastare l'efficacia delle metodologie di attacco più diffuse. Formalmente si considerano multi-fattore quei processi di autenticazione in cui le informazioni richieste all'utente derivano da due o più categorie (fattori). I tre fattori di base sono storicamente riconducibili alle locuzioni "qualcosa che l'utente sa", "qualcosa che l'utente è", "qualcosa che l'utente possiede". Un'autenticazione a due fattori si contrappone dunque ad una comune autenticazione basata sulla conoscenza della sola password e può prevedere, ad esempio, l'introduzione di un ulteriore codice numerico generato da un dispositivo esterno in possesso dell'utente.

L'aumento della generale consapevolezza dei consumatori nei confronti delle tematiche di sicurezza e la diffusione di sistemi di autenticazione più robusti hanno costretto i frodatori a rivedere le proprie tecniche e gli strumenti utilizzati. Questo braccio di ferro ha prodotto una nuova generazione di malware, denominati *banking trojan*, in grado di perpetrare una classe di minacce nota come *Man-in-the-Browser*[1] (MITB) [70].

Questi nuovi malware operano sfruttando la sessione attiva dell'utente (*session hijacking*[2]) per manipolarne le attività senza che questi possa rendersene conto. In ma-

[1] a volte viene utilizzato anche il termine *content manipulation attacks*[89]

[2] letteralmente "dirottamento della sessione"

niera trasparente possono iniziare autonomamente delle nuove transazioni o modificare "al volo" i dati delle transazioni inserite durante la navigazione. Se necessario possono alterare le risposte del server in modo da nascondere le tracce della propria attività. Le tecniche di autenticazione a più fattori sono purtroppo inefficaci verso questo tipo di minaccia. Infatti il funzionamento del trojan viene innescato dopo l'avvenuto accesso dell'utente al servizio online, di fatto aggirando la fase di autenticazione. Allo stesso modo l'utilizzo di connessioni sicure, come nel caso del protocollo HTTPS, non garantisce l'integrità o la confidenzialità delle informazioni.

Gli attacchi MITB non sono confinati ad una particolare area geografica ma sono invero una minaccia a livello globale. Ciononostante sono maggiormente prevalenti in quelle zone dove l'autenticazione a due fattori ha conosciuto una penetrazione più capillare, costringendo i cybercriminali ad adottare strategie più elaborate. In Regno Unito ad esempio un singolo istituto bancario ha riportato perdite per £600.000 come risultato di una serie di attacchi del trojan PSP2-BBB[45; 78]. Altri stati Europei come Italia, Germania, Spagna, Paese Bassi, Francia e Polonia hanno introdotto l'autenticazione multi-fattore negli ultimi anni, circostanza che ha portato ad un incremento nel numero di attacchi MITB in queste regioni. La Germania in particolare è stata duramente colpita dato che questo tipo di attacchi risulta essere una delle poche possibilità di successo per commettere una frode bancaria online in questo Paese. Anche gli Stati Uniti non sono immuni a questo fenomeno: nel 2010 l'FBI ha smantellato un'organizzazione criminale che aveva derubato gli istituti bancari americani di una cifra complessiva di oltre 70 milioni di Dollari infettando i computer delle proprie vittime con il banking trojan Zeus [61], uno dei più avanzati e noti software di questo tipo.

Sono state individuate alcune strategie per mitigare l'effetto della diffusione di questi sofisticati malware e ridurne così le probabilità di successo. RSA [45] delinea due principali aree verso cui concentrare gli sforzi per assicurare la sicurezza delle transazioni. La prima è la cosiddetta autenticazione *Out-Of-Band* (OOB) ovvero l'utilizzo di un secondo canale di comunicazione alternativo attraverso il quale l'utente può confermare la transazione, disaccoppiando le operazioni di inserimento e di approvazione. Varie tecnologie sono state sviluppate che implementano questo modello utilizzando allo scopo dispositivi dedicati (chipTAN, EMV/CAP, ...) o dispositivi già in possesso dell'utente finale (mTAN[1], chiamate telefoniche automatizzate, fax, ...). L'altra im-

[1] *mobile TAN*

portante linea di difesa è quella presidiata dalle tecnologie di *transaction monitoring*, sistemi quindi che effettuano *fraud detection* in contrapposizione con le tecnologie di *fraud prevention* prima menzionate.

I sistemi di fraud detection sono diffusi da anni in molteplici settori industriali. Tra le applicazioni troviamo l'identificazione di frodi assicurative, frodi telefoniche, frodi sanitarie e abusi della garanzia. È però proprio nel settore finanziario e più specificatamente nell'ambito delle transazioni con carta di credito che questi sistemi hanno conosciuto una grande popolarità. Storicamente infatti gli enti emittenti di carte di credito sono stati restii all'introduzione di misure di autenticazione dell'utente (per ragioni relative all'usabilità del prodotto finanziario) e si sono perciò affidati a sistemi di monitoraggio per garantire la sicurezza dei pagamenti e scoprire eventuali tentativi di frode attraverso l'analisi dei dati transazionali. Questo approccio è stato successivamente adottato anche nei moderni contesti *Card Not Present* (CNP), che corrispondono a tutte quelle situazioni in cui non è necessaria la presenza fisica della carta di credito per completare la transazione (e.g. un pagamento via Internet) e non è possibile quindi ottenere una firma dell'utente che supporti l'autorizzazione dell'operazione come lo è invece nel caso di una transazione commerciale tradizionale presso un negozio fisico.

È opportuno valutare alcune proprietà che caratterizzano il problema della fraud detection in generale ma che sono ancora più evidenti nel settore bancario online. L'attività di ricerca e sviluppo dei sistemi di sicurezza antifrode è complicata dal fatto fondamentale che lo scambio di idee in questo campo è fortemente limitato. Secondo Bignell [37] descrivere apertamente le tecniche di fraud detection non è logico in quanto fornisce agli stessi frodatori le informazioni necessarie per aggirare il meccanismo di identificazione. Per problemi di privacy e sicurezza la letteratura riguardante le tecniche di rilevazione di frodi finanziarie è ridotta al minimo e generalmente non di dominio pubblico data la sensibilità delle informazioni trattate, spesso contenenti dettagli finanziari confidenziali dei clienti di un particolare istituto di credito, e per tale ragione vi è una fondamentale carenza di risultati sperimentali e dataset rappresentativi del mondo reale (non generati quindi in maniera sintetica). Quand'anche questi dati fossero disponibili bisogna ugualmente tener conto di altri importanti fattori:

1. **Il dataset è di grandi proporzioni e fortemente sbilanciato.** Il numero di transazioni quotidiane è molto elevato; malgrado questo la percentuale di frodi

sul totale delle transazioni è molto bassa. Si viene dunque a creare la difficoltà di individuare poche frodi localizzate in un insieme molto più vasto di transazioni legittime.

2. **L'individuazione delle frodi deve avvenire possibilmente in modalità *real-time* o *near real-time*.** Attualmente la maggior parte delle banche processa le proprie transazioni in *batch*, spesso a fine giornata, garantendo così una certa finestra temporale per l'analisi delle operazioni più rischiose. La tendenza verso una maggior velocità nei pagamenti e l'introduzione di innovazioni come l'area SEPA[1] stanno però stressando questo modello riducendo i margini temporali di investigazione e portando alla necessità di sistemi antifrode efficienti e in grado di segnalare i casi a rischio in tempo utile.

3. **L'occorrenza di una frode interessa un arco molto limitato di tempo.** A differenza di altri settori (e.g. telecomunicazioni) la frode non sottende un arco temporale di lunga durata ma spesso consta di un singolo episodio frodatorio, finalizzato all'ottenimento del massimo vantaggio economico. Questo determina sia una minore possibilità per la vittima di individuare autonomamente l'attività illecita in corso ma soprattutto richiede un sistema di identificazione in grado di segnalare, con precisione, eventi intrinsecamente puntuali e non ripetibili.

4. **Lo scenario delle frodi è dinamico.** I cybercriminali devono costantemente adattare e sviluppare le loro tecniche di attacco per superare le nuove barriere difensive. Il numero di nuovi malware rilevati è aumentato esponenzialmente negli ultimi anni. Fino a 100.000 nuove varianti vengono individuate giornalmente [1]. Il sistema antifrode deve dunque essere in grado di difendere da un numero in continua crescita di tipologie di attacchi.

5. **L'informazione disponibile è limitata.** Le informazioni disponibili ad un sistema antifrode bancario sono quasi esclusivamente transazionali come l'importo, sorgente e destinatario del pagamento o causale. Nel nostro caso il sistema è in grado di accedere anche ai dati relativi alla navigazione dell'utente. Cionondimeno, dato lo spostamento del punto di attacco sempre più verso il computer della

[1] *Single Euro Payments Area*: prevede, tra le altre disposizioni, la disponibilità di un versamento sul conto del beneficiario entro la fine della giornata successiva all'ordine di pagamento

vittima, è difficile raccogliere sufficienti prove per supportare l'ipotesi di frodi sempre più sofisticate, in quanto non vi è alcune informazione relativa all'intero processo di compromissione.

6. **Il comportamento dei singoli utenti è differenziato.** Un portale bancario online offre ai clienti un punto d'accesso unificato ad una moltitudine di servizi. Nell'operare le proprie attività usufruendo di questi servizi ogni utente può agire in maniera anche sensibilmente differente dagli altri. Inoltre lo stesso account può avere più modalità di utilizzo ad esempio se condiviso con un'altra persona o se la tipologia di spesa varia dalla postazione in cui viene effettuato il pagamento (al lavoro piuttosto che a casa). Gli attacchi più sofisticati a loro volta cercando di emulare il comportamento degli utenti. La combinazione di questi fattori complica non solo la caratterizzazione stessa di una frode ma rende complicato perfino il riconoscimento di una frode dal comportamento genuino di un utente.

L'individuazione di una frode deve avvenire in tempi molto brevi poichè risulta molto difficile recuperare la perdita se la frode viene scoperta soltanto molto tempo dopo. Purtroppo sono rari gli utenti dei servizi di home banking che consultano con regolarità la propria situazione finanziaria e i movimenti del conto corrente personale perciò spesso non sono in grado di scoprire e riportare i casi di frode in tempi utili, riducendo di molto le probabilità di un recupero dei fondi illegittimamente trasferiti. Inoltre tutti gli allarmi generati dal sistema di individuazione devono passare al vaglio di un operatore per un'ulteriore investigazione manuale, un processo che può risultare costoso, sia in termini di tempo che di risorse. Per questo un sistema di fraud detection finalizzato all'individuazione di frodi bancarie online deve garantire un'elevata accuratezza e un alto tasso di individuazione pur mantenendo un basso tasso di falsi positivi tale da generare un numero di allarmi gestibile, in relazione ovviamente al volume di transazioni giornaliere e alle risorse dedicate all'investigazione all'interno di ogni particolare istituto bancario.

Le caratteristiche del problema sopra descritte e gli stringenti requisiti di *business* pongono quindi una seria sfida alle tecniche di fraud detection attualmente adottate per la sicurezza di altri settori.

1.1 Obiettivi

Il lavoro oggetto di questo elaborato è stato realizzato nell'ambito di un progetto di *stage* organizzato dall'Università degli Studi di Padova. Lo stage, della durata di 6 mesi, si è svolto all'interno di SEC Servizi S.c.p.a[1], un'importante azienda padovana specializzata nella fornitura di servizi informatici in *outsourcing* rivolti soprattutto al settore finanziario/bancario. SEC Servizi è una realtà di dimensioni significative (oltre 300 dipendenti) che vanta più di 40 clienti nel settore finanziario per un totale di oltre 4.000.000 di utenti serviti sull'intero territorio nazionale. Chi scrive è stato inserito nel contesto aziendale, a stretto contatto con i dipendenti, lavorando a tempo pieno con la supervisione di un responsabile, l'Ing. Zandolin.

Nel pacchetto di servizi offerti dall'azienda vi è la fornitura di un portale di *online banking*, tramite il quale gli utenti delle banche clienti possono consultare le informazioni relative al proprio conto corrente e disporre pagamenti attraverso la rete Internet. Questo contesto, fortunatamente, non è stato finora impattato significativamente da fenomeni di frodi informatiche, date sia la ridotta dimensione dei singoli clienti sia una frequente adozione tra questi della tecnologia OTP (cfr. Capitolo 3) ma la diffusione di attacchi di crescente sofisticazione e la veloce evoluzione dell'economia sotterranea del cybercrimine hanno portato il *management* a investire preventivamente per potenziare la sicurezza delle transazioni online degli utenti e mantenere alto il livello di fiducia dei clienti nel servizio erogato. Lo scenario attuale impone agli istituti finanziari di affiancare ai dispositivi di sicurezza distribuiti agli utenti un sistema lato banca di transaction monitoring, in grado di fare leva sulla grande mole di dati storici per tracciare dei profili comportamentali dei clienti e compiere un'analisi differenziale che permetta di individuare le operazioni anomale, ovvero quelle che si discostano significativamente dai parametri determinati.

Questo progetto nasceva quindi con l'obiettivo di determinare il contributo che questa grande quantità di informazioni poteva offrire nella costruzione dei profili descrittivi degli utenti. In particolare erano evidenti due ambiti fondamentali da affrontare: l'attività online e le abitudini di spesa. Attraverso l'esame critico della letteratura nell'ambito della fraud detection e delle materie correlate si richiedeva di sviluppare

[1] `http://www.secservizi.it`

un modello utente efficiente e di implementarlo in seguito come parte di un sistema prototipo per l'individuazione di frodi. Il sistema doveva rispettare i seguenti requisiti:

- valutazione della legittimità delle nuove transazioni (solo bonifici bancari in questa fase sperimentale)
- modalità di esecuzione in offline, con tempi ridotti di reazione agli eventi fraudolenti
- generazione di report ad intervalli regolari
- disponibilità di un'interfaccia grafica con un buon livello di usabilità in grado di servire anche come strumento per ulteriori indagini degli *incident*, presentando tutte le informazioni necessarie all'investigazione
- configurabilità per singolo cliente-banca
- integrazione trasparente con il sistema informativo corrente

Al di là dei requisiti indicati, la spiegabilità dell'ouput del sistema era infine una proprietà desiderabile in quanto facilita la comprensione delle segnalazioni e una migliore e semplificata divulgazione delle informazioni all'interno del contesto aziendale.

La scelta di considerare soltanto i trasferimenti tramite bonifico non è stata dettata dalla volontà di semplificare in prima istanza il problema ma deriva da una valutazione delle frodi registrate precedentemente a questo progetto in SEC Servizi; queste infatti risultavano tutte perpetrate attraverso bonifici bancari.

I risultati del progetto sono incoraggianti. Entro il termine del periodo di stage è stato sviluppato un sistema antifrode che, secondo le specifiche dettate, incorpora differenti tecniche per la sintetizzazione dei profili descrittivi dei comportamenti di navigazione e di spesa degli utenti. Gran parte del lavoro è stato dedicata allo studio di un modello per la descrizione dell'attività online a partire dalle tracce contenute nei log dei web server. Un modello statistico è stato inoltre proposto per la valutazione del livello di anomalia di una transazione sulla base dell'importo. Una metodologia efficiente di estrapolazione e memorizzazione dei dati di navigazione è stata implementata per garantire le prestazioni del sistema.

È stata inoltre sviluppata un'interfaccia grafica, basata su tecnologie web, che soddisfa i requisiti di progetto, benché non adatta ancora ad un utilizzo in produzione. Il

sistema prototipo è attualmente in fase di prova ma ha già permesso a SEC Servizi di individuare 2 casi di frode (la totalità di quelli ufficialmente riportati dalla data di attivazione del progetto), dal valore complessivo superiore ai 10.000 €, senza generare un eccessivo *overhead* per gli operatori addetti all'investigazione. Il tasso di falsi positivi è infatti contenuto.

1.2 Organizzazione della tesi

Il seguito di questo documento è organizzato come di seguito. Una panoramica relativa al settore di ricerca sulla fraud-detection è presentata nel Capitolo 2, accompagnata da una breve rassegna della letteratura di riferimento per quanto riguarda le applicazioni nell'online banking. Nel Capitolo 3 viene esposto lo scenario attuale del cybercrimine con un'attenzione particolare agli aspetti organizzativi ed economici di questa articolata struttura sotterranea. Ad una descrizione degli attacchi MITB è dedicato il Capitolo 4 che affronta anche le tecniche di infezione più comuni e le possibili misure di protezione. Il Capitolo 5 affronta la discussione delle fonti di dati a disposizione, analizzandone il contributo informativo nell'ottica della costruzione di profili descrittivi del comportamento degli utenti. I frutti di questa analisi sono raccolti nel Capitolo 6 in cui viene formalmente descritto il framework alla base del sistema anti-frode progettato e ne vengono esplicitati i componenti. Una descrizione dell'implementazione del sistema nel contesto applicativo è fornita nel Capitolo 7. Il Capitolo 8 espone i risultati ottenuti mentre al Capitolo 9 sono riservate le conclusioni.

2

Panoramica

In questo capitolo verranno esposti i lavori e i risultati più importanti relativamente alla ricerca nell'ambito della fraud detection. L'attenzione verrà posta in particolare su quei campi di applicazione nei quali, storicamente, questo tipo di sistemi ha avuto maggior sviluppo, con un occhio di riguardo al settore finanziario.

2.1 Frodi

Una frode si configura come l'ottenimento, attraverso artifizi e raggiri, di un vantaggio a scapito di un altro soggetto. Il tipo di vantaggio ottenuto è molto spesso monetario ma non sempre. La frode scientifica, nella quale vengono prodotti o falsificati dei dati [71], è finalizzata a incrementare la reputazione del frodatore; le truffe elettorali sono finalizzate a mantenere od ottenere potere [84] mentre altre frodi possono essere perpetrate per ragioni ideologiche. Le frodi, o truffe, non sono un fenomeno recente ma anzi la loro evoluzione ha seguito di pari passo la storia dell'umanità. Le nuove tecnologie, oltre a modificare i nostri stili di vita e i nostri comportamenti e a consentirci di accedere a servizi attraverso nuove modalità, hanno contemporaneamente aperto nuove possibilità ai frodatori. Alcune delle attività criminali più tradizionali come il riciclaggio di denaro sono diventate più semplici da perpetrare mentre nuove frodi si sono affiancate a quelle esistenti come ad esempio le frodi nella comunicazione cellulare o l'intrusione nei sistemi informatici. Varie soluzioni sono state sviluppate sia per prevenire che identificare le truffe: si parla rispettivamente di fraud prevention e fraud detection. Per quanto riguarda la prevenzione alcuni meccanismi di questo tipo sono

ad esempio l'utilizzo delle filigrana o di disegni olografici nella stampa delle banconote, numeri di identificazione personale (PIN) per le carte di debito, sistemi di sicurezza Web per la transazioni con carta di credito, Subscriber Identity Module (SIM) per la telefonia mobile o password per l'accesso a sistemi informatici o servizi bancari telefonici. Vari metodi di prevenzione sono discussi in Sezione 4.3 in merito alle frodi bancarie online. Chiaramente nessuno di questi metodi è infallibile, anzi spesso le soluzioni sviluppate sono un compromesso tra vari fattori come il costo, l'usabilità (per l'utente finale) e l'efficacia della misura adottata.

Se le soluzioni di fraud prevention mirano a contrastare i frodatori costituendo di fatto un ostacolo all'attività criminale, lo scopo dei sistemi di fraud detection è quello di individuare una frode il più velocemente possibile una volta che questa è stata effettivamente perpetrata. In questo senso i sistemi di fraud detection entrano in gioco nel caso di fallimento di una o più misure di prevenzione pur dovendo, in pratica, operare in maniera continuativa in quanto non si è tipicamente informati dell'aggiramento delle stesse.

La fraud detection è una disciplina in continua evoluzione. Metaforicamente la "battaglia" tra ricercatori e frodatori si può rappresentare come un costante braccio di ferro. I frodatori, lungi dal voler interrompere le proprie attività criminali, devono reagire all'introduzione di nuovi sistemi di identificazione adattando le proprie strategie o elaborandone di nuove. Cionondimeno la ricerca è complicata dal ridotto scambio di idee e informazioni che caratterizza questo settore. Spesso le tecniche di individuazione sviluppate non vengono descritte dettagliatamente al pubblico per non fornire ai frodatori le informazioni necessarie per aggirarle. I dataset non vengono resi disponibili e i risultati sono spesso censurati, rendendo difficoltosi la valutazione e il confronto di tecniche differenti.

L'attività di ricerca relativa alla fraud detection sembra focalizzarsi su tre principali settori economici, vale a dire l'industria delle carte di credito, il settori assicurativo e quello delle telecomunicazioni [92]. Il numero di pubblicazioni relative invece al settore bancario risulta molto ridotto. Questa situazione, con tutta probabilità, non va imputata alla mancanza di attività di ricerca in questo campo ma piuttosto a superiori esigenze di privacy, segretezza e interessi economici legati a questo specifico dominio. Cionondimeno consideriamo importante un'analisi più generale della letteratura in quanto metodi e tecnologie utilizzati in una particolare applicazione possono essere tradotti

ed utilizzati in ambiti differenti. In questo senso è conveniente studiare le metodologie applicate ad una certa tipologia di dataset piuttosto che concentrare l'attenzione sul tipo di frode per cui sono state originariamente sviluppate. Dati gli obiettivi di questo progetto ci limiteremo allo studio degli approcci di tipo *machine learning* (cfr. 2.2). Un'altra area di ricerca che spesso viene associata alla fraud detection è l'*intrusion detection* nei sistemi informatici. Le tecniche elaborate in questo ambito possono essere infatti utilizzate per individuare l'insorgere di attività malevola finalizzata alla frode. I due problemi sono perciò profondamente legati tra loro e si ha che l'attività di ricerca in un campo risulta valida anche nell'altro.

2.2 Fraud detection e machine learning

Molti prodotti commerciali e non realizzati per individuare frodi all'interno di specifici domini sono essenzialmente sistemi esperti, spesso basati su una serie di regole determinate aprioristicamente dalla conoscenza del dominio. Anche nel caso in cui le regole vengano generate automaticamente spesso è necessario un intervento di configurazione manuale dei parametri per adattare il sistema a specifici contesti e requisiti di business. A partire dall'insieme di regole ogni istanza (una transazione con carta di credito, una chiamata cellulare, i dati di una richiesta di rimborso, ...) viene esaminata e eventualmente segnalata come possibile tentativo di frode. Data la natura stessa di questi sistemi essi producono un'elevata quantità di falsi positivi e hanno tipicamente un basso tasso di individuazione. Un aspetto ancora più importante è che le regole non sono adattive e perdono quindi di efficacia man mano che i frodatori evolvono le loro strategie richiedendo un costante lavoro di aggiornamento manuale.

Per questi motivi il focus del progetto di tesi, fin dalla stesura degli obiettivi, è stato posto nella studio e nella realizzazione di un sistema antifrode che utilizzi metodologie e approcci di tipo machine learning. Nel seguito daremo una breve definizione di machine learning fornendo alcuni richiami dei concetti fondanti. Su tale base esploreremo le caratteristiche del dominio della fraud detection in relazione allo sviluppo di tecniche legate a questa branca dell'intelligenza artificiale, evidenziando le problematiche più incisive.

Il campo del *machine learning* (apprendimento automatico) studia la progettazione di sistemi software in grado di indurre schemi, caratteristiche di continuità o regole

da un dataset costituito da esperienze passate. Caratteristica generale dei sistemi di machine learning è l'abilità di migliorare le prestazioni future attese sulla base dei nuovi input ricevuti. L'output prodotto può essere discreto, nel qual caso il problema affrontato è detto di classificazione, mentre, se al contrario l'uscita assume valori continui, si fa riferimento a problemi di regressione.

Le tecniche di machine learning sono divisibili in base alla tipologia di dati cui sottostanno. A seconda delle informazioni disponibili essi determinano la scelta tra l'utilizzo di metodi detti di apprendimento supervisionato o non-supervisionato. Se è nota a priori la classificazione associata agli elementi contenuti nel dataset è possibile ricorrere a tecniche supervisionate. In tal caso il compito del sistema sarà quello di creare un modello che impari questa mappatura e di predire una classificazione corretta delle nuove istanze. Algoritmi molto popolari in questa tipologia sono reti neurali artificiali, Support Vector Machines (SVM), alberi di decisione o classificatori bayesiani e rules-based. Nell'apprendimento non-supervisionato la classificazione iniziale dei dati non è disponibile e il sistema può solamente elaborare un modello a partire dalle proprietà dei dati, individuando specifiche correlazioni tra le istanze. I principali algoritmi utilizzati per affrontare questa classe di problemi ricadono nella comune definizione di tecniche di clusterizzazione.

Approcci generali

Nell'ambito della fraud detection l'obiettivo di un sistema è quello di fornire una classificazione di ogni evento analizzato, sia esso una transazione tramite carta di credito o una chiamata telefonica internazionale. Questo tipo di problema prevede due sole classi, positiva e negativa, ad indicare rispettivamente gli eventi illegittimi e quelli genuini. L'utilizzo di tecniche supervisionate richiede dunque la disponibilità di un dataset che contenga esemplari di record, sia fraudolenti che non, precedentemente etichettati. Entrambe le tipologie vengono utilizzate per costruire un modello (un classificatore) che permette di assegnare ad ogni nuova istanza osservata una delle due classi. Ovviamente questo approccio richiede una certa confidenza sulla qualità dell'etichettatura dei record nel dataset: la classe indicata deve corrispondere alla classe reale per non incorrere in situazioni di *mislabeling*. Un punto debole, che deriva dalla natura stessa di questi sistemi, sta nella capacità di individuazione che è purtroppo limitata al riconoscimento

dei soli tipi di frode precedentemente noti e sottoposti al sistema per l'analisi. È opportuno inoltre porre l'accento sulla necessità di disporre di dati appartenenti ad entrambe le classi. Una variante di questa tipologia di apprendimento utilizza solamente istanze provenienti dalla classe considerata genuina; si parla in questo caso di apprendimento semi-supervisionato.

I metodi di apprendimento non-supervisionato, abbinati a dataset non classificati a priori, cercano invece di identificare quegli account, clienti, transazioni e così via che si discostano dalla definizione di normalità elaborata dal sistema a partire dall'analisi del dataset. Sfortunatamente, data la differenza di materiale informativo rispetto al caso delle tecniche supervisionate, questi approcci soffrono di un superiore tasso di falsi positivi, ovvero esemplari genuini incorrettamente classificati come frodi.

Dati

La struttura e la dimensionalità dei dati processati nei vari sistemi sperimentali per l'individuazione di frodi è piuttosto varia. A seconda del dominio applicativo i dataset di partenza sono profondamente differenti, sia orizzontalmente (numero di attributi) che verticalmente (numero di record). Ad esempio in [92] vengono esaminati relativamente a questo aspetto numerosi lavori in letteratura. Si passa da meno di 10 attributi in diversi dataset relativi a sistemi progettati per identificare frodi interne a più di 100 attributi presenti in un particolare dataset nel campo delle frodi su carta di credito. Anche il numero di record di un dataset presenta una forte variabilità tra i vari campi di applicazione. Buona parte dei dataset contenenti transazioni su carta di credito contengono più di un milione di record e in un caso si arriva fino a 12 milioni di transazioni in un anno [56]. Nel settore delle telecomunicazioni si trovano infine i dataset più numerosi, comprensivi di transazioni generate da centinaia, migliaia o addirittura milioni di account quotidianamente.

In alcuni settori questa imponente mole di dati viene a creare un vero e proprio *stream* di record che richiede un'analisi continua e una ripetitiva applicazione degli algoritmi. La situazione diventa particolarmente complicata se si considera che i sistemi di fraud detection devono spesso lavorare in modalità *real-time* o *near real-time* per avere migliori *chance* di identificare un account soggetto a frode nel più breve tempo possibile minimizzando di conseguenza le perdite: un sistema in grado di classificare perfettamente (con il 100% di accuratezza) tutte le transazioni ma che impiegasse un

mese per farlo risulterebbe completamente inutile in pratica. L'enorme quantità di dati pone inoltre serie questioni riguardanti la memorizzazione degli stessi e le relative strategie per ricercare e aggiornare i dettagli dei profili dei vari account.

Abbiamo già menzionato nel Capitolo introduttivo come sia complicato per i ricercatori accedere a dataset reali sui quali effettuare analisi comparative dei differenti algoritmi o addirittura sperimentare un nuovo algoritmo. L'unico caso noto a chi scrive è quello di un piccolo dataset relativo a richieste di risarcimento nel settore delle assicurazioni per auto [91]. Per ovviare a questa problematica talvolta si è ricorso a dati puramente sintetici per analizzare le prestazioni di una particolare tecnica [33]. I motivi che trattengono le aziende o le organizzazioni dal rendere pubblici questi dati sono molto forti: da un lato si vuole giustamente tutelare la privacy dei propri clienti e dall'altro si vuole conservare il vantaggio competitivo verso la concorrenza.

Abbiamo già visto come l'applicazione delle tecniche di apprendimento supervisionato non possa prescindere da un'etichettatura iniziale dei record nel dataset. Ciò equivale a richiedere un partizionamento di tutti i dati in esemplari fraudolenti e genuini. Purtroppo questa situazione ideale è ben distante dalla realtà. Infatti nei contesti applicativi una certa etichettatura non è a priori definitiva. Consideriamo il caso in cui una certa transazione su carta di credito viene inizialmente considerata legittima dal sistema di fraud detection; dopo un mese però il cliente riceve il resoconto delle sue spese e scopre un'operazione sconosciuta. Successivamente ad una segnalazione al reparto anti-frode della società emittente il record sarà infine ri-etichettato come fraudolento. C'è un altro caso che può essere anche più deleterio e che ci sarà utile spiegare con un esempio simile. Un cliente, dopo aver speso una cospicua somma con la propria carta, si pente e cerca di rimediare allo sconsiderato gesto segnalando l'operazione come una frode: in questa situazione il record viene effettivamente etichettato come fraudolento (anche se di una frode differente da quella realmente avvenuta) ma l'utilizzo della carta è stato legittimo. Senza scomodare la malafede di un cliente lo stesso caso si ha anche qualora un utente dimentichi un'operazione che ha realmente effettuato. Inoltre non tutte le frodi vengono individuate portando ad ulteriori record etichettati incorrettamente.

Questa discussione mette in luce come la classificazione iniziale degli esemplari sia tutt'altro che affidabile; malgrado ciò la maggior parte degli articoli in letteratura non tiene in considerazione questo aspetto e al più si ipotizza che il numero di casi

etichettati in maniera scorretta sia in numero sufficientemente basso da non influire sulla qualità dell'analisi del sistema anti-frode. Questo è il caso di molti sistemi semi-supervisionati nei quali si fa spesso l'ipotesi che tutte le transazioni nel dataset siano legittime (e non si fa uso degli esemplari fraudolenti per la produzione dei modelli) o non supervisionati i quali semplicemente non prevedono alcun tipo di etichettatura o separazione a priori in classi dei record. Molte volte non c'è alternativa all'utilizzo di queste tecniche in quanto non si possiede inizialmente un dataset già partizionato ed è troppo costoso provvedere ad un'etichettatura manuale dei record oppure non sono effettivamente disponibili sufficienti esemplari per rappresentare entrambe le classi (in pratica quello che si verifica è che non ci sono abbastanza record fraudolenti).

L'ultimo problema evidenziato è molto noto in letteratura come il *class imbalance problem* (problema dello sbilanciamento delle classi) e risulta una questione molto importante e decisiva da affrontare anche in casi meno estremi rispetto a quello sopra indicato. Una stima spesso citata relativa al tasso di frode è quella di 1 esemplare fraudolento ogni 1000 [92].

Japkowicz ha discusso l'effetto dello sbilanciamento in un dataset [76], valutando differenti strategie per modificare la distribuzione iniziale delle classi: le tecniche esposte prendono il nome di undersampling e oversampling. Per *undersampling* si intende un procedimento che trasforma la distribuzione delle classi nel dataset eliminando, in maniera casuale o focalizzata, una parte degli esemplari della classe maggioritaria (esemplari legali) mantenendo invece la totalità degli esemplari minoritari, fino al raggiungimento della proporzione desiderata. Al contrario con l'*oversampling* si cerca di aumentare artificialmente il numero di esemplari minoritari reinserendoli più volte casualmente (*resampling* con ripetizione) . Sempre per mitigare questa problematica Chawla et al. hanno sviluppato una tecnica più sofisticata denominata SMOTE (Synthetic Minority Over-sampling TEchnique) [47] per la sintetizzazione di nuovi esemplari a partire da quelli esistenti.

Finora abbiamo esplorato la morfologia e la struttura dei dati solamente nella direzione "verticale". Soffermiamoci ora brevemente sulla seconda dimensione, quella longitudinale, ovvero gli attributi. In generale gli attributi possono essere binari ("è piovuto oggi?"), numerici (variabili continue come la temperatura o discrete come l'importo di una transazione), categorici (nominali come nel caso del colore di un'auto od ordinali come il grado di severità di un danno dovuto ad un incidente) o di tipo misto

quando sono presenti attributi di più tipologie. Se ogni record è costituito da un singolo attributo si parla di dati univariati; alternativamente se il numero degli attributi è superiore si parla di dati multivariati nel qual caso gli attributi possono essere tutti dello stesso tipo oppure un misto delle differenti tipologie indicate sopra. La natura degli attributi non è una questione secondaria: ad esempio quando si utilizzano tecniche statistiche differenti modelli devono essere prodotti a seconda se gli attributi da modellare sono di natura numerica o categorica, continua o discreta.

Criteri di valutazione delle prestazioni

Le prestazioni di un algoritmo di machine learning sono tipicamente valutate attraverso la corrispondente matrice di confusione (*confusion matrix*). Le colonne sono la classe predetta dall'algoritmo mentre le righe rappresentano la classe reale. Nel caso di classificatori binari, che precedono cioè due sole classi in uscita, la matrice assume le dimensioni 2×2 come in Figura 2.1. Questa situazione rappresenta il caso tipico nel settore della fraud detection dove gli esemplari da classificare si distinguono in legittimi (classe negativa) e fraudolenti (classe positiva).

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

Figura 2.1: Matrice di confusione

Nella matrice di confusione TN è il numero di esemplari negativi correttamente classificati (*True Negatives*), FP è il numero di esemplari negativi incorrettamente classificati (*False Positives*), FN è il numero di esemplari positivi incorrettamente

classificati (*False Negatives*) e TP è il numero di esemplari positivi correttamente classificati (*True Positives*). Chiaramente sono desiderabili algoritmi e modelli che presentano valori elevati di TP e TN a fronte di bassi valori per FP e FN. Una misura che viene spesso ricavata da questa matrice è l'accuratezza (*accuracy*), definita come $Accuracy = (TP + TN)/(TP + FP + TN + FN) = (TP + TN)/(P + N)$. Alternativamente viene utilizzata la metrica complementare, ovvero l'*error rate*, definito come $1 - Accuracy$.

Sfortunatamente queste metriche di valutazione non risultano applicabili nel caso di dataset molto sbilanciati come nel caso della fraud detection. Un esempio illuminante è quello relativo alla misura dell'accuratezza che rappresenta in pratica il rapporto tra gli esemplari classificati correttamente e quelli totali. Valori tipici di probabilità di frode, spesso citati anche in letteratura, sono intorno ad un caso ogni 1000. Con queste cifre un classificatore *naïve* che indicasse ogni esemplare come legittimo otterrebbe un'accuratezza del 99,9% pur senza alcuna segnalazione di frode.

Diverso ma dello stesso tono è invece l'argomento a sfavore del *true positive rate* (TPR). Il motivo è dato dal fatto che, nelle applicazioni di fraud detection, i costi corrispondenti all'errata classificazione (costo dei falsi positivi e dei falsi negativi) sono differenti, spesso non conosciuti a priori e possono variare da caso a caso ed evolvere anche nel tempo. Ad esempio nel settore finanziario si hanno costi superiori all'errore dato da un falso negativo (frode non segnalata che si traduce in perdita finanziaria) rispetto a quelli in cui si incorre nel caso di un falso positivo (investigazione superflua, potenziale disturbo per il cliente,...).

Altri criteri di valutazione utilizzati in letteratura sono l'analisi delle curve *Receiver Operating Characteristic* (ROC), *Area Under Curve* (AUC), *cross entropy*, il punteggio di Brier e l'indice di Gini. Le curve ROC forniscono una rappresentazione grafica della sensibilità di un classificatore binario al variare della soglia di discriminazione. Modificando il valore di questa vengono tracciati su di un grafico i punti corrispondenti alle percentuali di veri positivi e falsi positivi ottenute dal sistema, ottenendo così una curva (per quanto tipicamente approssimata da tratti rettilinei). La Figura 2.2 mostra un esempio di curva ROC. In questo caso la percentuale di falsi positivi è mappata sull'asse x mentre l'asse y corrisponde alla percentuale di veri positivi. Un sistema ideale dovrebbe fornire un tasso del 100% di veri positivi senza produrre alcun falso positivo, condizione rappresentata dal punto in alto al sinistra nel grafico. Nella

applicazioni reali un aumento nella capacità di individuazione del sistema determina spesso un compromesso dando luogo ad errori di classificazione che rendono la curva più simile a quella visibile in Figura 2.2. E' utile, per meglio comprendere il ruolo di questo grafico, notare come la retta che collega l'origine al punto in alto a destra rappresenti la condizione di un classificatore che esprima un output perfettamente casuale.

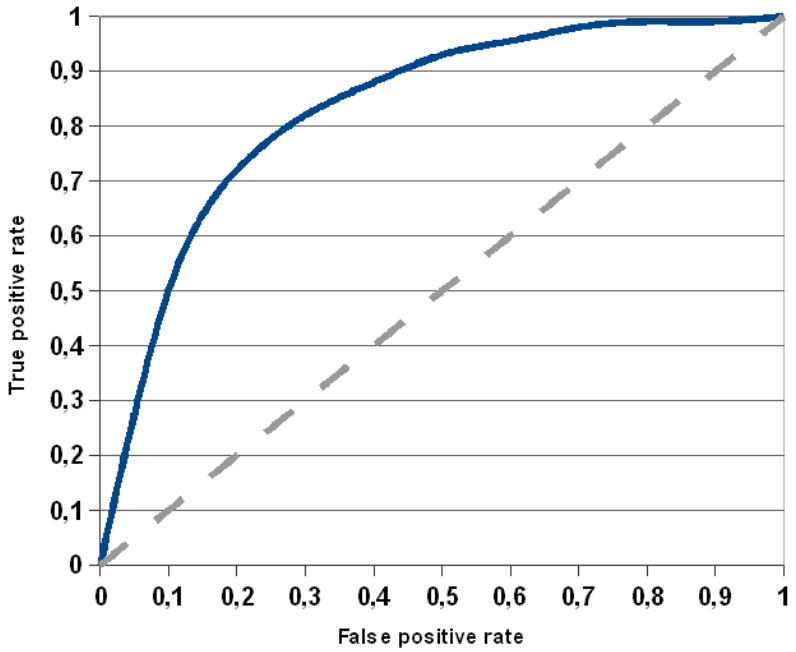

Figura 2.2: Curva ROC

Spesso per riassumere in un solo valore numerico il grado di una curva ROC si utilizza la misura dell'area al di sotto della curva stessa, l'AUC. Informalmente questa misura rappresenta la probabilità che un esemplare casuale della classe positiva possa essere incorrettamente classificato come appartenente alla classe negativa; in pratica risulta equivalente all'indice di Gini. Questa misura sintetica ha però ricevuto numerose critiche sia nell'ambito della comunità di ricerca sul machine learning sia, in particolare, in quello più specifico della fraud detection [72].

Un approccio migliore è quello di minimizzare un'appropriata funzione di costo o di fissare un qualche parametro (ad esempio il numero massimo di casi che è possibile investigare giornalmente in dettaglio) e cercare di massimizzare il numero di casi

fraudolenti individuati rispettando i vincoli imposti. La definizione di una struttura di costo e di una relativa misura è stata affrontata in [98].

Un'analisi delle prestazioni di un sistema di fraud detection reale non può prescindere dal considerare anche alcuni parametri temporali. Uno di questi è la velocità con cui le frodi vengono individuate (il *detection time* o *time to alarm*). Questa variabile dipende certamente dalla tipologia del sistema ovvero se questo è online (*real-time* o anche *event-driven*) oppure offline (*batch-mode* o *time-driven*) ma può includere una valutazione del tempo necessario per processare ogni singolo caso analizzato, se questo è influente.

Un aspetto che viene trascurato in letteratura ma che risulta poi importante una volta che il sistema viene effettivamente implementato è la sua usabilità. Buona parte dei sistemi antifrode infatti prevede un processo di revisione da parte di uno o più operatori (denominati anche auditor) per vagliare i casi segnalati e decidere se proseguire o meno con le politiche aziendali previste in caso di sospetta frode (blocco dell'account, annullamento dell'operazione, telefonata al cliente, ...). Un buon sistema antifrode dovrebbe essere in grado di fornire a questi operatori tutti gli strumenti necessari per effettuare l'attività di analisi nel minor tempo possibile con tutte le informazioni necessarie per poter approfondire ed giudicare il caso.

2.3 Intrusion detection

L'idea degli *Intrusion Detection System* (IDS) è stata proposta in un report del 1980 da J.P Anderson [34]. Anderson definisce un tentativo di intrusione o una minaccia come la potenziale possibilità di un tentativo deliberato e non autorizzato di

1. accedere ad informazione,
2. manipolare informazione,
3. rendere un sistema inaffidabile o inutilizzabile.

In seguito il concetto di IDS ha subito un esteso lavoro di ricerca e sviluppo che ne ha modificato i contorni. Oggigiorno un IDS monitora essenzialmente un sistema e l'interazione tra questo e gli utenti. Ogni interazione viene comparata attraverso una lista di regole o altro tipo di modello che definisce i limiti entro i quali un'interazione

è considerata normale; se questa serie di controlli non viene superata l'interazione è etichettata come anomala e segnalata. Ciò non determina con certezza la natura maligna dell'attività ma fornisce piuttosto un forte indicatore di cui un amministratore di sistema deve tenere conto.

I sistemi IDS differiscono in molti aspetti ma, ai fini dela nostra analisi, i più interessanti risultano essere la localizzazione (il punto di raccolta delle informazioni analizzate) e le tecniche di individuazione utilizzate. Il seguito di questa sezione si focalizza proprio sulle caratteristiche chiave di questi aspetti.

Localizzazione

Possiamo distinguere due tipologie di IDS in base al punto di localizzazione dell'informazione analizzata: *host-based* (HIDS) e *network-based* (NIDS). I sistemi host-based raccolgono i dati da elaborare da un singolo computer (host), spesso attraverso i meccanismi di *auditing* forniti dal sistema operativo, mentre i sistemi network-based esaminano dati scambiati tra diversi computer che comunicano in rete cercado di individuare pacchetti riconducibili ad un attacco.

Entrambi i sistemi perseguono lo stesso obiettivo ma con differenti vantaggi e svantaggi. Un evidente punto di forza dei sistemi NIDS è la possibilità di essere implementati in maniera trasparente, senza cioè richiedere uno stravolgimento del sistema monitorato. Essi però sono fortemente limitati dall'impossibilità di analizzare il traffico criptato. I sistemi HIDS superano questo limite ma sono legati ad un singolo host mancando quind di una visione globale. Inoltre, mentre gli NIDS non hanno alcun impatto sulle prestazioni di un singolo computer ciò non è vero per i sistemi HIDS. La Tabella 2.1 mostra una comparazione delle caratteristiche dei sistemi NIDS e HIDS, evidenziando i punti di forza e le debolezze di ciascuna delle due tipologie.

Tecniche di individuazione

Le tecniche utilizzate per il rilevamento di un'intrusione sono divise in tre macrocategorie:

- *misuse detection* (anche indicata con la dicitura *signature-based detection*)
- *specification*

HIDS	NIDS
Centralizzato	Distribuito
Forte impatto sul sistema	Piccolo o nessun impatto sul sistema
Possibile collo di bottiglia	Nessun impatto sulle prestazioni
Esamina l'attività su di una singola macchina	Esamina l'attività globalmente sulla rete
Può analizzare traffico criptato	Non può analizzare traffico criptato
Dipendente dal sistema monitorato	Generico

Tabella 2.1: Comparazione di punti di forza e debolezze dei sistemi NIDS e HIDS

- *anomaly detection*

Gli IDS di tipo *misuse detection* basano il loro funzionamento sulla conoscenza di pattern e attributi di attacchi precedentemente codificati, rendendo questo approccio corrispondente alle tecniche di apprendimento supervisionato. Ogni interazione viene esaminata alla ricerca di questi indicatori, denominati *signature*. I sistemi IDS di questo tipo sono generalmente veloci, adatti anche ad un utilizzo real-time, e spesso identificano con accuratezza gli attacchi per i quali sono provvisti di signature. Si tratta di uno dei metodi tradizionalmente adottati anche dai software anti-virus per l'identificazione delle applicazioni malevole. Purtroppo, siccome è necessario analizzare manualmente un attacco per svilupparne una descrizione formale, i tempi di risposta ad una nuova minaccia (un cosiddetto attacco "0-day") possono essere nell'ordine di ore o anche giorni, ovvero fintanto che un nuovo set di signature non sarà disponibile rendendo vulnerabile il sistema. Inoltre è difficile proteggere un'applicazione o un sistema sviluppati internamente, tanto più che alcune tipologie di attacchi potrebbero non avere caratteristiche comuni facilmente identificabili e modellabili.

La tecnica *specification* prevede l'intervento di un esperto che, in seguito ad uno studio approfondito, produca una serie di specifiche (da cui il nome) del sistema da monitorare. Durante l'utilizzo del sistema le interazioni degli utenti sono comparato con queste specifiche e ogni deviazione è segnalata. Un approccio complementare a questo è quello di descrivere i comportamenti indicativi di un attacco; a volte infatti è più semplice descrivere il comportamento non atteso piuttosto di quello atteso. Un

problema di questo approccio è che richiede un elevato grado di esperienza per redarre le specifiche, spesso più alto di quello necessario per sviluppare l'applicazione stessa. Inoltre il sistema protetto può evolvere e richiedere dei cambiamenti alle specifiche precedentemente individuate.

L'ultima tipologia di IDS utilizza una serie di tecniche che ricadono sotto la definizione di *anomaly detection*. Questi sistemi operano cercando di individuare i comportamenti che deviano da ciò che è definito come comportamento normale, interpretando tali scostamenti come sintomi di un'attività malevola. L'assunto cui sottostanno questi sistemi è ovviamente che i comportamenti anomali (da cui il nome) siano sensibilmente diversi da quelli normali, in una maniera cioè effettivamente esprimibile, qualitativamente o quantitativamente. Mentre le prime due tipologie di IDS non sono state particolarmente interessate a sviluppi o ricerca, molto florida è invece la letteratura sulle tecniche di anomaly detection applicate all'individuazione delle intrusioni. Questa maggiore attenzione è certamente dovuta alla potenzialità di tale categoria di sistemi di individuare anche attacchi precedentemente non noti o nuove metodologie di attacco.

I sistemi di anomaly detection operano in due fasi: una fase iniziale di *training* e una fase di individuazione. Nella fase di training l'IDS induce un modello di comportamento normale dall'analisi del *training set*. Il training set solitamente non contiene tutti gli esemplari, positivi o negativi, possibili. Perciò l'algoritmo di induzione deve essere in grado di produrre, a partire dai dati disponibili, un modello più generale. Questo capacità dell'algoritmo è anche detta generalizzazione. Lo sviluppo di una tecniche di anomaly detection necessita di un bilanciamento di questa proprietà in quanto se da un lato consente l'individuazione di nuove minacce simili a quelle note un livello eccessivo può portare ad una riduzione della sensitività del sistema. I dati possono provenire dal log di un'applicazione, dall'output di un altro software o attraverso interfacce definite. Nella fase di individuazione il sistema analizza i nuovi dati che gli vengono somministrati alla ricerca di eventi anomali, in base al modello creato nella fase di training. Se viene determinata una differenza, nei termini di una qualche metrica, superiore ad una soglia definita a livello di sistema l'evento analizzato viene considerato anomalo e potrebbe essere la prova di un'attività illegittima in corso.

Per completezza segnaliamo infine l'esistenza di sistemi cosiddetti ibridi che uniscono due o più approcci tra quelli elencati cercando di combinare i punti di forza di un'architettura per sopperire alle debolezze di un'altra.

2.3.1 Letteratura di riferimento

In questa Sezione diamo una panoramica ad alto livello di alcuni dei lavori più significativi di applicazione dei concetti del machine learning e dell'anomaly detection nell'ambito dello sviluppo di sistemi IDS, sia network-based che host-based.

L'utilizzo di metafore e concetti mutuati dall'osservazione della natura ha spesso ispirato la ricerca nel settore dell'informazione. Non si distingue infatti da questa tendenza nemmeno il settore dell'intrusion detection. Negli anni '90 alcuni ricercatori hanno elaborato tecniche che si rifanno ai sistemi biologici autoimmuni i quali sono grado di distinguere ciò che è "l'essere in sè" da tutto ciò che è "alieno. Lavoro seminale di questo filone di ricerca è un articolo di Forrest et al. [65] nel quali gli studiosi descrivono un "senso del sè" applicato ai processi UNIX. Analizzando le sequenze di chiamate di sistema di un determinato processo gli autori hanno realizzato un database di sequenze legittime di lunghezza predefinita; l'esecuzione del processo viene poi monitorata osservando la presenza o meno di sequenze anomale. Diversi autori hanno sperimentato variazioni di questo semplice modello portando nell'analisi nuovi elementi come ad esempio gli attributi delle singole chiamate o modellando con tecniche più sofisticate il flusso del codice [64].

Un filone di ricerca che ha conosciuto recentemente un intensificarsi dell'attenzione riguarda i sistemi di intrusion detection specifici per il particolare sotto-dominio delle applicazione web su protocollo HTTP. Questo protocollo ormai ubiquo è alla base delle comunicazioni Internet; l'enorme diffusione dei server web unita allo scarso livello di sicurezza con cui vengono spesso implementate applicazioni poi esposte su Internet ha catalizzato l'interesse dei cyber-criminali e di conseguenza ha evidenziato la necessità di svluppare adeguati sistemi di sicurezza specifici per questa tipologia di traffico.

In questo solco si colloca la ricerca di Kruegel et al. [80; 81]. Questi ricercatori hanno sviluppato un sistema di anomaly detection che utilizza diversi modelli per descrivere il comportamento normale di una serie di applicazioni Web attraverso l'analisi delle richieste HTTP memorizzate in un file di log di un web-server Apache. Il sistema deriva una serie di profili specifici per ogni singola applicazione *server-side* installata e i suoi parametri. Gli attributi valutati vanno dalla lunghezza alla distribuzione dei caratteri fino all'ordine degli stessi nella richiesta utilizzando semplici modelli statistici basati su media e varianza e tecniche più sofisticate come *Hidden Markov Model* (HMM) e

distribuzioni di caratteri. Lo scopo di un modello è di assegnare un valore di probabilità alla richiesta globalmente o ad uno dei suoi parametri. Sulla base dell'ouput dei singoli modelli ogni richiesta viene valutata come normale o come potenziale attacco. La decisione viene raggiunta combinando il valore di uscita dei vari modelli e sollevando un allarme se in almeno un caso l'ouput supera una certa soglia stabilita in fase di training.

Corona et al. [49] hanno implementato HMM-Web, un prototipo di IDS costituito da un classificatore multiplo, anche detto *Multiple Classifier System* (MCS), basato sulla fusione dell'output di un *ensamble* di HMM.

Ingham et al. [75] utilizzano invece una rappresentazione delle richieste HTTP basata sulla teoria dei linguaggi formali. Nell'articolo gli autori delineano una metodologia per l'induzione di una rappresentazione tramite *Deterministic Finite Automata* (DFA) delle richieste dirette ad un'applicazione web considerando nell'analisi l'intera intestazione dei messaggi HTTP.

2.4 Lavori correlati

Non sono molti gli articoli in letteratura che si sono occupati dell'individuazione delle frodi nel settore dell'online banking. Gran parte della ricerca in questo settore si è invece interessata ad un altro aspetto, peraltro importante, ovvero la prevenzione delle frodi[1] [37; 55]. Aggelis [32] ha proposto un sistema di fraud detection offline. Benchè l'articolo non discuta ampiamente i dettagli implementativi, motivando questa scelta nell'ottica di salvaguardare la sicurezza e l'efficacia del sistema stesso, propone un'analisi del dominio dei dati identificando alcune *features* che possono, secondo l'autore, essere decisive nel determinare le capacità di riconoscimento di un sistema antifrode. Il lavoro è precedente alla diffusione degli attacchi MITB, pertanto alcune delle valutazioni proposte risultano obsolete.

In [79] gli autori propongono un sistema che combina aspetti globali e locali nell'analisi. Per ogni singolo utente viene mantenuto un buffer contenente un certo numero delle più recenti transazioni mentre un secondo buffer contiene le transazioni della sessione corrente. Sulla base di questi due insiemi vengono calcolati il profilo corrente

[1] una panoramica delle misure di fraud prevention adottate da molti istituti di credito è discussa nella Sezione 4.3

di utilizzo e il profilo medio che sono poi confrontati con metodi statistici. Il sistema in questo modo monitora la frequenza dei pagamenti, il numero di tentativi falliti di inserimento della password e la frequenza degli accessi al servizio effettuando un'analisi differenziale. L'analisi globale richiede invece l'installazione di uno specifico software su ogni dispositivo impiegato per collegarsi al portale di online banking. Questo software fornisce un servizio di *device identification* che consente di determinare, in maniera sicura, le coppie account-dispositivo utilizzate per l'accesso. Il sistema sfrutta queste informazioni per rafforzare l'ipotesi di frode nei casi in cui un account si colleghi tramite un dispositivo precedentemente coinvolto in casi fraudolenti o alternativamente per diminuire il livello di allarme qualora la combinazione account-dispositivo sia considerata legittima. Infine le probabilità di frode ottenute da questi due differenti blocchi di analisi vengono combinate sulla base delle teoria dell'evidenza di Dempster-Shafer. Benchè questo approccio sia interessante presenta alcuni limiti piuttosto evidenti. Il primo è dovuto alla necessità di installare un software esterno, fondamentale per il monitoraggio globale degli utenti. Il secondo limite è nella ridotta capacità di individuare gli attacchi più sofisticati che sono in grado di aggirare le misure di *fingerprinting* [59]. In questi nuovi casi tutte le transazioni risulterebbero comunque provenienti dal dispositivo della vittima risultando perciò in un falso senso di sicurezza.

Nel 2012 Wei et al. hanno proposto i-Alertor, un sistema che combina un insieme di diversi algoritmi per produrre un punteggio complessivo di rischio [103]. In particolare gli autori fanno uso di *decision forest* (un insieme di k alberi decisionali con diversi nodi radice), reti neurali artificiali *cost-sensitive* e di un classificatore basato sulla scoperta dei cosiddetti *Emerging Pattern* (EP), proposti come un metodo per evidenziare comportamenti anomali discriminanti in grado di segnalare un tentativo di frode.

Dato l'utilizzo di tecniche supervisionate, applicate ad un dataset fortemente sbilanciato (un rapporto di 8.000.0000 di transazioni genuine contro 1.500 fraudolente viene riportato nell'articolo) una fase di *pre-processing* precede la fase di modellazione riequilibrando gli esemplari tramite un meccanismo di *undersampling* per la classe maggioritaria ed operando successivamente una selezione degli attributi più significativi e una discretizzazione dei valori numerici per migliorare l'efficienza e il tempo di calcolo nelle fasi successive. Una nota sul sistema di valutazione delle prestazioni: le $top - n$ transazioni segnalate da i-Alertor, ordinate a seconda del punteggio di rischio calcolato, vengono considerate nel calcolo del *detection rate*. A seconda del numero n di

transazioni che si è disposti a considerare per l'investigazione il *detection rate* aumenta contenendo però allo stesso tempo il numero di falsi positivi entro un *range* controllato. Secondo Wei et al. i-Alertor ha ottenuto un *detection rate* superiore rispetto ad un sistema esperto precedentemente in uso nell'ambiente di test e può essere combinato con lo stesso per raggiungere prestazioni complessivamente migliori. Sfortunatamente la disponibilità di esemplari etichettati per entrambe le classi di interesse non è scontata; ciò rende questo approccio non applicabile in svariate realtà.

Karlsen e Killingberg [77] sviluppano un'analisi dei problemi nella sicurezza dei sistemi per l'online banking, soffermandosi su diverse tipologie di minacce e attacchi, considerando tutti i livelli dell'architettura di questi servizi. L'articolo presenta una discussione sulle possibili fonti di dati soffermandosi sulle informazioni ricavabili dai log di *audit* che possono aiutare a comporre un quadro complessivo del comportamento normale degli utenti nell'utilizzo del sistema e definire così una serie di profili per il monitoraggio e la segnalazione degli eventi fraudolenti; l'approccio che viene dunque seguito è quello tipico dell'anomaly detection.

3

Scenario

In questo capitolo viene descritto in maniera approfondita lo scenario con cui il sistema qui proposto andrà a confrontarsi e la sua evoluzione nel recente passato. Non a caso verranno utilizzati termini mutuati dal gergo economico; il mercato "sotterraneo" delle frodi è per molti aspetti simile ai mercati mercati legittimi e sottende a simili regole e dinamiche economiche.

3.1 Infrastrutture criminali

Il mercato *underground* delle frodi si è drasticamente evoluto negli ultimi anni, seguendo logiche tipiche dell'economia legale. Proprio come in qualsiasi mercato libero non sono più diffusi i soggetti che portano a compimento una frode dalle fasi di preparazione tecnica dell'attacco alle fasi finali di monetizzazione dello stesso. I frodatori si sono invece specializzati nella fornitura di specifiche tecnologie o servizi, entrando a far parte di una vera e propria catena di distribuzione. Le attività dei frodatori si possono così dividere in due categorie principali: *harvesting* e *cashing out* [44]. In Figura 3.1 è schematizzata questa divisione in ruoli, supportata dalla presenza di due infrastrutture, una tecnica e un'altra operativa.

L'attività di *harvesting*, traducibile come "raccolta" o "mietitura", consiste in tutte quelle azioni che vengono intraprese dal frodatore con l'intento di accumulare dati sensibili (quali credenziali compromesse, PIN, SSN o altro ancora) con l'intento di rivenderli successivamente ad un terzo soggetto che effettivamente monetizzerà (*cashing out*) questi dati. Per realizzare i loro illeciti scopi gli *harvester* si appoggiano ad

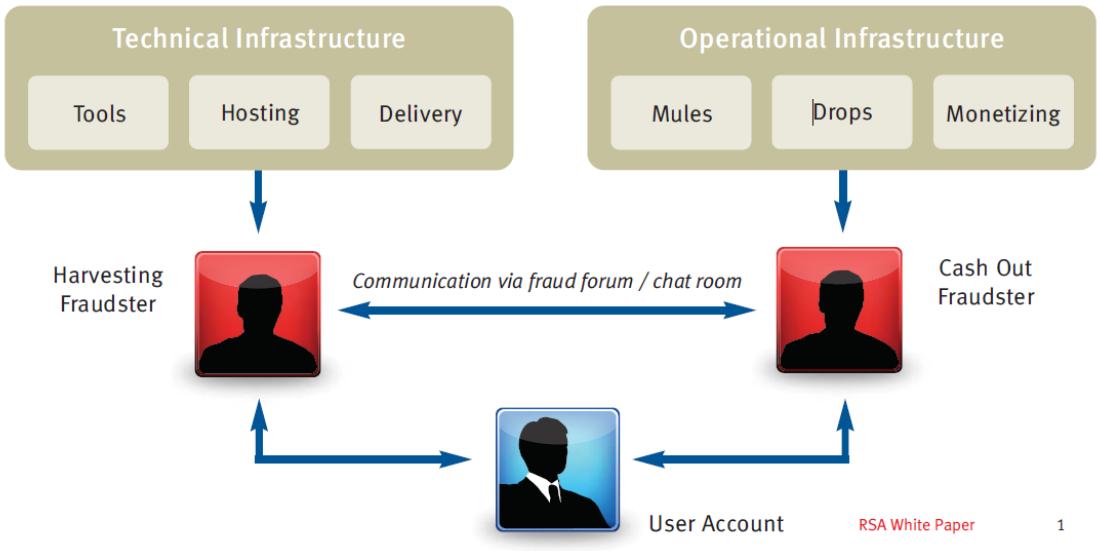

Figura 3.1: Schema dell'organizzazione del mercato *underground* delle frodi. Fonte: RSA [44]

un'infrastruttura tecnologica che comprende tutta una serie di strumenti e servizi di supporto. Quest'infrastruttura è in continua evoluzione, non solo sotto il profilo puramente tecnico, dovendo confrontarsi quotidianamente con i nuovi sforzi messi in campo nella realizzazione di sistemi di sicurezza sempre migliori ma anche sotto la motivazione di una forte spinta innovativa dettata dalle regole della concorrenza, proprio come nel caso delle economie reali.

Quest'ultimo fattore in particolare ha portato ad una riduzione dei costi per mettere in atto un attacco (ad esempio *Zeus*, uno dei trojan più utilizzati, era venduto nel 2009 al costo di $1000 mentre software meno avanzati come *Limbo* potevano essere acquistati nel mercati *underground* per $350 [44]) e ad una semplificazione degli strumenti con la nascita di veri e propri servizi di supporto e customizzazione. L'abbassamento delle due principali barriere all'ingresso, quella tecnica e quella economica, ha consentito così un notevole ampliamento della base dei potenziali frodatori.

Un cybercriminale, per ottenere un elevato numero di credenziali, può contare sulla fornitura di pagine truffaldine, kit automatizzati per il phishing e relativi plugins personalizzati, servizi per l'inoltro di "spam", database di indirizzi email, *junk traffic* e *SEO poisoning*. Veri e propri portali di e-commerce sono nati dove i frodatori possono rivendere i dati di cui sono illecitamente entrati in possesso.

Relativamente ai trojan esiste la possibilità di acquistare una serie di prodotti quali *exploit kits* (veri e propri pacchetti software contenenti programmi appositamente sviluppati per sfruttare vulnerabilità di sicurezza di sistemi operativi o applicazioni) o servizi come l'accesso a *botnets*, codice malevolo personalizzato, servizi per la crittografia delle comunicazini, hosting "bulletproof" (hosting che garantiscono una maggior persistenza dei contenuti caricati dai cybercriminali in quanto non prevedono forti restrizioni sugli stessi), servizi di installazione di software malevolo con pagamento di una quota ad infezione, supporto per la configurazione e altro ancora.

Difficilmente un frodatore prende parte ad un intero ciclo di frode, molto più spesso invece i cybercriminali operano in collaborazione con altri soggetti: proprio come nelle realtà criminali tradizionali non tutti infatti sono disposti a "sporcarsi le mani" allo stesso livello. Esiste allo scopo una vera e propria infrastruttura operativa, complementare all'infrastruttura tecnologica che abbiamo già trattato, costituita da diversi attori che mettono a disposizione, ovviamente dietro compenso, una serie di servizi di supporto, fondamentali per quei frodatori che non dispongono di una propria rete di complici.

In questa infrastruttura particolarmente importante è il ruolo dei *money mules*, persone arruolate allo scopo di prelevare fondi ricevuti nei loro conti personali e trasferirli, tipicamente verso qualche Paese Est-Europeo, attraverso servizi di invio di denaro. Questi soggetti possono essere anche utilizzati come intermediari per ricevere e rispedire beni acquistati attraverso carte di credito rubate o conti correnti compromessi (in questo caso si parla più propriamente di *item-drop mules*). Alcuni frodatori, denominati *mule herders*, si sono specializzati nel reclutamento di questi soggetti, che avviene per lo più tramite email sfruttando uno schema ben noto. Il frodatore infatti invia alla vittima un'offerta di lavoro da casa per una posizione come "agente per il trasferimento di denaro" o "direttore regionale", spesso accompagnando l'email con un collegamento ad un sito di rappresentanza di una compagnia di facciata (per rendere il messaggio più credibile) ed un riferimento al compenso. In Figura 3.2 è riportata una pagina nella quale i frodatori descrivevano l'attività commerciale dell'azienda fittizia come quella di intermediazione nella spedizione di merci, utilizzando un corretto inglese e termini tipici del marketing.

Una volta che la vittima sarà caduta nella rete diventerà inconsapevolmente parte delle operazioni di riciclaggio del denaro proveniente dai conti compromessi trasferendo

Figura 3.2: Un esempio di pagina Web utilizzata per reclutare *money mules*

i fondi ricevuti da questi ultimi verso altra destinazione, possibilimente in contanti. A seconda dell'ammontare di denaro riciclato la vittima potrà essere effettivamente ricompensata con una piccola commissione sul totale trasferito. È interessante notare l'atteggiamento dei frodatori verso questi soggetti che apparentemente vengono trattati come dei veri e propri dipendenti. Il motivo dietro questo interesse è molto semplice: per un frodatore procurarsi delle credenziali compromesse di una carta di credito o di un conto corrente è diventato sempre più semplice ed economico, grazie alla grande disponibilità e al costo per unità notevolmente diminuito di tali "risorse nel mercato nero, mentre i money mules rappresentano sempre più il vero "collo di bottiglia" del processo di frode.

3.2 Il modello Fraud-as-a-Service

Il modello su cui si si basa il mercato delle frodi e che abbiamo delineato nella Sezione precedente è stato etichettato da RSA con la denominazione *Fraud-as-a-Service* (FaaS), sulla falsariga dei nuovi modelli di business nati attorno al concetto di *cloud computing* come ad esempio *Software-as-a-Service* (SaaS) o *Infrastructure-as-a-Service* (IaaS) [43]. Il termine è stato coniato nel 2008 e nel frattempo il mercato delle frodi si è ulteriormente sviluppato seguendo un andamento evolutivo simile a quello dei mercati dell'economia emersa, trasformandosi in un sistema a "catena di distribuzione". In questo arco di tempo non sono cambiati perciò tanto i "beni" venduti quanto piuttusto altri parametri come la scalabilità, la rilevanza dei servizi, l'aumento della disponibilità (con la conseguente diminuzione dei costi), il supporto al cliente e infine le garanzie per l'acquirente.

FaaS è un modello di business, una strategia utilizzata dai cybercriminali per vendere servizi e prodotti esistenti (*fraud commodities*), in maniera facile, veloce ed efficiente senza trascurare la personalizzazione basata sui requisiti dei singoli clienti. La maggior parte di questo processo avviene in forum sepolti nelle aree "sotterranee" della rete Internet, in siti che spesso richiedono l'accesso su invito da parte di membri *senior* o il pagamento di una quota di iscrizione o ancora certificati digitali e URL di accesso personalizzate, misure che i cybercriminali ritengono necessarie sia per rendere più difficoltose le azioni delle forze dell'ordine che monitorano questo mercato illegale sia per limitare le possibilità di analisi da parte dei ricercatori. In questi "luoghi" virtuali le

frodi vengono pianificate, vendute e orchestrate tra i membri criminali che possono, attraverso il web, comunicare anche da Paesi molto distanti tra loro, di modo che spesso il denaro riciclato da una transazione fraudolenta a discapito di un conto corrente statunitense venga trasferito successivamente in Europa rendendo ulteriormente più complicato il lavoro delle forze dell'ordine fattesi necessarie una forte cooperazione e investigazioni coordinate tra le istituzioni internazionali.

Come detto nel recente passato gli sviluppatori di trojan si sono molto concentrati nel migliorare il proprio servizio di supporto, inizialmente relegato solamente a sessioni di chat dal vivo attraverso servizi di messaggistica istantanea (Jabber, ICQ). In questo modo uno sviluppatore poteva sopportare solo un numero limitato di clienti prima che la qualità del servizio stesso degradasse o questo venisse del tutto azzerato. Comprendendo però la rilevanza di un buon servizio di supporto tecnico ai loro clienti gli sviluppatori hanno profuso molte risorse nel trovare nuovi modi di preservare la propria base utenti. Molti dei fornitori nella "catena di distribuzione" hanno introdotto, accanto ai loro prodotti o servizi, nuove forme di garanzia e assistenza: dal cambio di credenziali non più valide fino a guide, configurazioni ad hoc o addirittura delle vere e proprie piattaforme di *Custumer Relationship Management* (CRM) come quella realizzata dal team di sviluppo di *Citadel* [46], uno dei trojan più evoluti attualmente sul mercato. Il team ha predisposto una piattaforma CRM per i propri clienti che fornisce FAQ, servizio di ticketing e consigli per configurare e operare al meglio il trojan da loro acquistato. L'iscrizione a questa piattaforma non è opzionale e prevede una quota mensile per l'accesso. Inoltre il software CRM è provvisto di uno spazio dedicato alla visualizzazione di messaggi pubblicitari, usufruibile dai cybercriminali che desiderino far conoscere le proprie offerte ad un pubblico molto selezionato.

Molto interessante è l'ascesa in questi anni di un nuovo tipo di servizio in supporto alle cosiddette *phone frauds*, le frodi telefoniche. I cybercriminali infatti hanno allargato i propri orizzonti e i canali attraverso cui attuare le frodi, spesso in risposta ad una maggiore comprensione dei temi della sicurezza da parte degli istituti finanziari e alle misure messe in campo dagli stessi per difendere i propri clienti. I "servizi di chiamata" mirano così a colpire le debolezze insite nei call center, spesso trascurati nelle analisi aziendali di rischio. Da anni esistono servizi, anche online [15], che consentono di effettuare chiamate con la possibilità di imitare un qualasiasi *Caller ID*, il numero

telefonico sorgente della comunicazione. Questa tecnica, denominata *Caller ID spoofing*, è stata storicamente utilizzata ache dai frodatori come un mezzo per confermare transazioni processate attraverso servizi di trasferimento di denaro [11]. Un frodatore che avesse voluto confermare una transazione poteva aggirare le barriere dovute alle differenze linguistiche, di genere o geografiche inserendo un messaggio online e cercando un complice in grado di parlare la stessa lingua del proprietario dell'account vittima. Riconoscendo in questo tipo di attività un'opportunità di guadagnare denaro illegalmente e a basso rischio alcuni soggetti hanno focalizzato la propria offerta proprio in questo ambito. I nuovi servizi sono delle vere e proprie piattaforme online attraverso le quali i cybercriminali possono compilare degli appositi moduli di richiesta inserendo tutti i dettagli della comunicazione (il numero da chiamare, il falso *Caller ID* di origine della chiamata, l'orario della stessa e tutte le altre informazioni necessarie) e attendere che del personale "professionale" e multilingua effettui la chiamata. Questi veri e propri call center permettono ai frodatori di comunicare, liberandosi delle barriere di cui sopra, non solo con istituti finanziari ma anche con agenzie di spedizione, commercianti e con i vari money mule reclutati per costi di poco superiori ai 10$ per singola chiamata [11]. In aggiunta alcuni di questi call center criminali offrono anche la registrazione in formato MP3 delle conversazioni che testimoniano la qualità del servizio ai propri clienti.

Analizzando questa evoluzione nel mercato delle frodi osserviamo che i nuovi processi nati attorno al modello classico della catena di distribuzione, così come hanno favorito l'efficientamento delle risorse all'interno di moltissimi settori dell'economia legittima, stanno portando questo tipo di concetti anche nel mondo del cybercrimine. Ricercatori e forze dell'ordine non devono scontrarsi più solamente con un singolo individuo alla ricerca di emozioni e facili guadagni ma con un'organizzazione settoriale, che coinvolge gruppi di criminali dei più assortiti, sempre più agguerriti e interessati solo a trarre il massimo dei profitti dai loro investimenti. La direzione intrapresa, oltre a portare ad un aumento delle vendite di trojan ed ad una operatività semplificata, induce un aumento conseguente del numero di attacchi con l'effetto finale di alimentare i casi di frode perpetrate con successo e le perdite per i consumatori, le banche e gli istituti assicurativi su cui si appoggiano. Le potenzialità di questo nuovo modello economico e la sua redittività per i frodatori sono ben rappresentate dal caso di un gruppo recentemente sgominato. Nel 2010 infatti l'FBI ha smantellato un'organizzazione criminale

internazionale attiva nel settore delle frodi finanziarie a mezzo informatico spiccando ordini di arresto per ben 27 persone, localizzate sia negli Stati Uniti che in Unione Europea (principalmente paesi dell'Est Europa) [61]. Prima dello smantellamento l'organizzazione era riuscita nel complesso a derubare vari istituti finanziari Americani per una cifra superiore ai 70 milioni di Dollari grazie al famoso trojan Zeus.

4

Gli attacchi Man-in-the-Browser

Con attacco *Man-in-the-Browser* (MITB) si intende un qualsiasi attacco ad un sistema informatico che sfrutti la presenza di un trojan caricato nel browser della vittima, o nel suo sistema operativo, per intercettare e manipolare le comunicazioni tra quest'ultima e una qualsiasi applicazione web. In questo capitolo analizzeremo le caratteristiche principali di questi attacchi, come vengono effettivamente perpetrati e con quali strumenti. Esploreremo le tecniche di infezione e forniremo infine una panoramica sui metodi attualmente utilizzati per mitigarli.

4.1 Descrizione dell'attacco

Il termine MITB deriva dal nome di un'altra tipologia di attacchi, molto noti nell'ambito della crittografia e della sicurezza informatica in generale: gli attacchi *Man-in-the-Middle* (MITM). Lo scenario tipico di un attacco MITM è rappresentato schematicamente in Figura 4.1. Una comunicazione tra due parti avviene attraverso un canale non sicuro favorendo un terzo soggetto che può allora essere in grado di intercettare le comunicazioni tra i due soggetti iniziali e impersonare di volta in volta uno di essi. Ciò crea l'illusione di una comunicazione privata ma in realtà l'intero flusso è controllato dall'attaccante il quale può semplicemente "ascoltare" i dati in transito oppure modificarli, alterando il contenuto della comunicazione. In gergo tecnico si dice che MITM è un attacco alla mutua autenticazione (o alla mancanza della stessa) nella comunicazione.

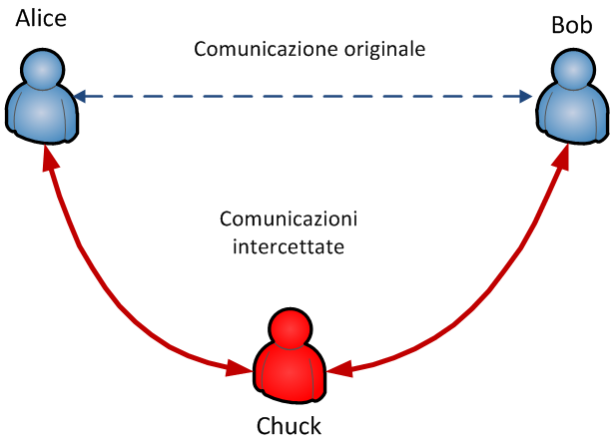

Figura 4.1: Schema concettuale di un attacco Man-in-the-Middle

Concretizzando questo schema astratto nello scenario delle comunicazioni Web possiamo pensare alle due parti originarie della comunicazione come un client e un server che sulla rete Internet si scambiano messaggi, utilizzando un protocollo per lo scambio di ipertesti non sicuro come HTTP. Al contrario il protocollo HTTPS, poggiandosi sulla tecnologia *Secure Socket Layer* (SSL), integra una forma di protezione contro gli attacchi MITM autenticando il lato server della comunicazione grazie all'ausilio di certificati digitali, firmati da una terza parte (detta Autorità di Certificazione) mutuamente riconosciuta, sia dal server che dal client, generalmente rappresentato in questo contesto dal browser dell'utente.

Mentra nel caso di MITM la terza parte che tenta di intromettersi nelle comunicazioni è fisicamente separata dai due soggetti iniziali, nel caso di un attacco MITB l'attaccante opera in maniera più subdola. Per configurare questo attacco il sistema della vittima, il client, viene dapprima infettato con un malware, tecnicamente un *proxy trojan*, sfruttando una combinazione di "ingegneria sociale" e vulnerabilità software presenti nel sistema o negli applicativi della vittima.

Facendo leva sulle comuni tecnologie per ampliare le capacità e funzionalità dei browser come ad esempio i *Browser Helper Objects* (BHO) o le estensioni il malware è in grado non solo di rendersi virtualmente invisibile ai software antivirus ma sopratutto di agire indisturbato all'interno del contesto di esecuzione del browser dell'utente. Quest'ultima proprietà in particolare permette al trojan di intercettare tutte le comunicazioni tra client e server scavalcando i meccanismi di sicurezza del protocollo HTTPS o

l'eventuale autenticazione a più fattori. Il meccanismo di funzionamento degli attacchi MITB è illustrato in Figura 4.2. È possibile notare la presenza di un agente malevolo, installato sul PC del client (Alice): l'agente (Chuck) osserva il traffico e lo manipola fornendo due diverse "visioni" dell'interazione con il sito (Bob) ai due *endpoint* della comunicazione.

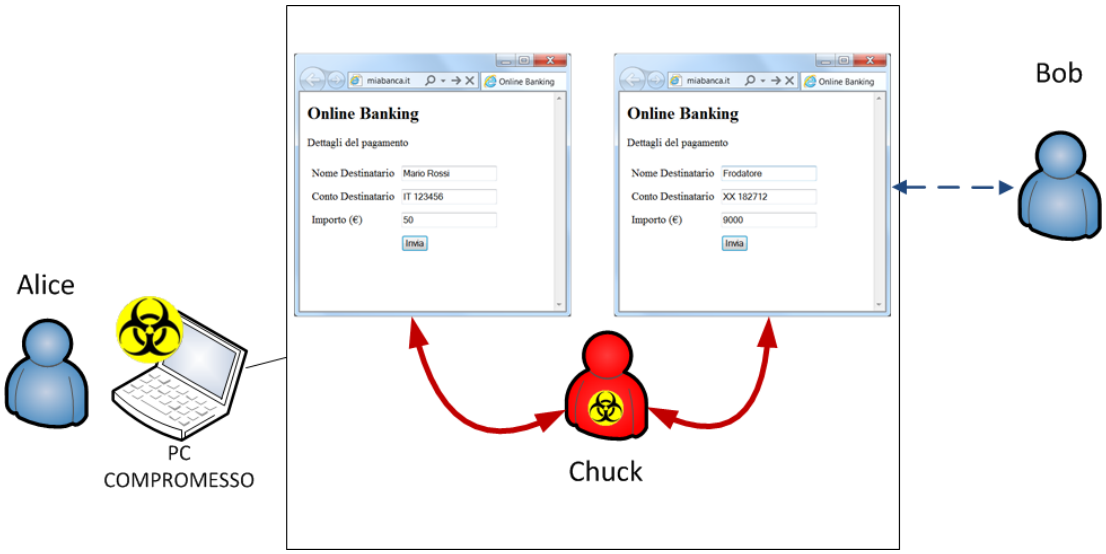

Figura 4.2: Schema concettuale di un attacco Man-in-the-Browser

Attraverso opportune impostazioni il trojan è configurato in modo da agire solo qualora l'utente visiti alcuni specifici siti online, ad esempio un sito di commercio elettronico o il portale di home banking di una certa banca. I bersagli di questi attacchi sono quasi sempre infatti istituti finanziari che permettono di operare e disporre denaro attraverso un semplice sito web.

Vediamo ora brevemente come si articola il flusso di un attacco attraverso l'esempio di un utente che voglia effettuare una transazione finanziaria online:

- Il trojan infetta il computer della vittima, sfruttando una qualche vulnerabilità di sicurezza o l'ingenuità della vittima stessa
- Il trojan installa un'estensione nel browser dell'utente di modo che questa venga caricata alla prossima esecuzione
- Successivamente, in un altro momento, l'utente riavvia il proprio software di navigazione

- Il browser carica l'estensione
- L'estensione registra un gestore di eventi (*event handler*) da eseguire al caricamento di ogni pagina
- Ogni qual volta viene caricata una pagina l'estensione verifica se l'URL della stessa è presente all'interno di una lista di siti bersaglio
- L'utente accede al sito sicuro `https://secure.mybank.com`
- Quando il gestore eventi individua la pagina visitata (ad esempio `https://secure.mybank.com/HomeBanking/do_transaction`) nella propria lista interna registra un ulteriore gestore applicato al click sul pulsante di invio dei dati
- L'utente compila tutti i campi del Form HTML relativi ad una nuova transazione
- Non appena l'utente preme il pulsante per inviare le informazioni relative alla transazione l'estensione blocca temporaneamente la comunicazione, estrae tutti i dati dai campi del modulo attraverso l'interfaccia *Document Object Model* (DOM) del browser e li memorizza
- L'estensione modifica i parametri della transazione tramite l'interfaccia DOM, predisponendo una transazione fraudolenta, e autorizza il browser a procedere con l'invio dei dati
- Il browser invia al server, in forma criptata, i dati compresi delle modifiche effettuate dall'estensione
- Il server riceve i dati sotto forma di una normale richiesta. Il server non è in grado di distinguere tra i dati originali e i dati modificati o di individuare segni di manipolazione
- Il server processa la transazione e genera una risposta
- Il browser riceve la risposta del server relativa alla transazione fraudolenta
- L'estensione individua l'URL `https://secure.mybank.com/HomeBanking/confirmation`, scansiona il contenuto HTML e sostituisce i dati modificati con quelli originali memorizzati precedentemente

- Il browser visualizza il contenuto della risposta con i dati originalmente inseriti dall'utente
- L'utente si accerta, ispezionando il contenuto della pagina, che la transazione da lui autorizzata sia stata completata correttamente

Si consideri questo schema come indicativo della procedura di attacco; molte varianti si possono individuare a seconda del grado di automatizzazione dell'attacco e delle contromisure messe in atto per mitigarlo. Inoltre le estensioni non sono l'unico metodo con cui si può configurare un attacco. In [70] vengono elencate le seguenti possibilità:

- Browser Helper Objects; si tratta di librerie caricate dinamicamente (DLL) da Internet Explorer o Windows Explorer all'avvio. Esse vengono eseguite all'interno di Internet Explorer e hanno accesso completo al DOM. ***Sviluppare BHO è molto semplice***
- Estensioni; simili ai BHO ma utilizzati da altri browser come Mozilla Firefox, Google Chrome o Opera. ***Sviluppare estensioni è semplice***
- UserScripts; scripts che vengono eseguiti dal browser, supportati dalla maggior parte dei browser (ad esempio Firefox grazie all'estensione Greasemonkey o Opera e Chrome nativamente) ***Sviluppare UserScripts è molto semplice***
- API-Hooking; questa tecnica consiste di un attacco MITM tra un eseguibile e le librerie DLL da esso caricate. Un esempio è il caso in cui li motore SSL di un browser venga eseguito da una DLL separata; con questa tecnica sarebbe possibile intercettare e modificare le comunicazioni tra il browser e il motore SSL ***Sviluppare API Hooks è complicato***
- Virtualizzazione; prevede l'esecuzione dell'intero sistema operativo in un ambiente virtualizzato per bypassare facilmente tutti i meccanismi di sicurezza ***Sviluppare attacchi virtualizzati è complicato***

Inizialmente il limite più grande di questa metodologia di attacco era quello di dover sviluppare un trojan personalizzato per ogni nuovo sito vittima, attività che richiede un'attenta analisi del sito stesso (e quindi spesso un accesso autorizzato alla sezione riservata agli utenti o quanto meno ad una sezione dimostrativa pubblica) in

modo da unificare la visualizzazione originale con quella prodotta dal software. Questa unificazione deve essere sia a livello grafico (e quindi immagini, icone, stili, colori e formattazione del testo) sia a livello linguistico in modo da non destare sospetti nell'utente. Se infatti si dovesse notare una discontinuità troppo evidente il pericolo (per il cybercriminale) sarebbe quello di allarmare l'utente il quale potrebbe interrompere la navigazione, chiedere delle verifiche presso il servizio di supporto della proprio banca e infine procedere all'annullamento dell'operazione indesiderata.

Sfortunatamente però, come abbiamo già potuto osservare nel Capitolo 3, negli ultimi anni anche i frodatori si sono evoluti arrivando a realizzare dei malware progettati per essere facilmente configurabili ed adattabili alle varie situazioni, senza perciò richiedere una riscrittura completa: una sorta di *framework* per le frodi sulla falsa riga di quelli nati per l'*assessment* delle vulnerabilità di sicurezza nei software o nei sistemi (Metasploit, Rapid7, Nessus, ecc..). Questi software, veri e propri prodotti commerciali anche se di un mercato *underground*, hanno contribuito a ridurre notevolmente le capacità tecniche necessarie per realizzare concretamente l'attacco ampliando la base di possibili attaccanti e riducendo i tempi di sviluppo per adattarsi ai cambiamenti dei portali. L'evoluzione nel tempo di questi prodotti ha portato alla realizzazione di *toolkit* sempre più sofisticati che sono in grado di automatizzare gran parte del processo di frode, dall'infezione al trasferimento illecito di denaro, riducendo il lavoro dei frodatori ad una semplice "customizzazione" e configurazione.

Dal lato tecnico un software malevolo di questo tipo tipicamente fornisce varie funzionalità tra cui:

- Iniezione di codice HTML (*HTML injection*) e Javascript per visualizzare pagine opportunatamente modificate ed eseguire attacchi di ingegneria sociale (*social engineering*) (ad esempio modificando la pagina di inserimento delle credenziali e richiedendo all'utente di inserire non solo nome utente e password ma anche il PIN della sua tessera ATM)

- Connettività remota (per accedere ai dati delle transazioni come nome del destinatario e il suo IBAN o, nei software più sofisticati, consentire il completo controllo del browser dell'utente tramite un'interfaccia apposita)

Altre funzionalità più avanzate sono la comunicazione criptata tra i vari trojan della *botnet* e il *Command & Control Center* (la postazione da cui il *botmaster* mantiene

il controllo delle sue operazioni fraudolente), registrazione video del contenuto dello schermo dell'utente, cattura dei caratteri digitati dall'utente e delle schermate ad intervalli regolari, furto di credenziali e dati sensibili da file e posta elettronica grazie all'analisi dei contenuti e ancora funzionalità per complicare l'individuazione del trojan agli anti-virus. Fra i più famosi malware di questa famiglia troviamo *Zeus*, *SpyEye*, *Adrenaline*, *Limbo Sinowal*, *Silent Banker* e *Citadel*.

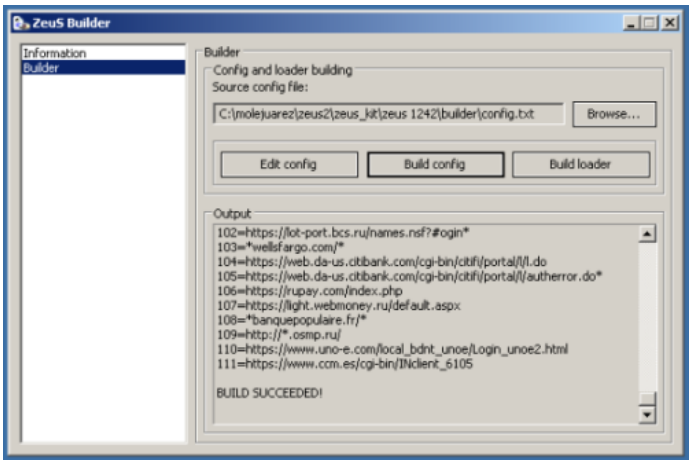

Figura 4.3: Interfaccia per la creazione dell'eseguibile del trojan Zeus

Concludiamo questa sezione introduttiva chiedendoci: quali sono i sistemi di autenticazione che possono essere aggirati da questo tipo di minacce? Sfortunatamente la risposta è che tutti i sistemi che utilizzano il PC come singolo canale per i dati della transazione (vedi Sezione 4.3 per una descrizione di un processo di autenticazione che sfrutti due diversi canali) sono vulnerabili. Alcuni dei più comuni sono:

- Username + Password
- PIN + TAN (Transaction Authentication Number)
- PIN + iTAN (*indexed* TAN)
- Certificati lato client
- Token SecurID o OTP
- Autenticazione biometrica

- Autenticazione tramite SmartCard
- Autenticazione tramte SmartCard e lettore Class3 (dotato di tastierino e display)

La caratteristica comune di tutti questi sistemi è senz'altro il fatto che non tengono in alcun modo in considerazione i dati delle transazioni: il trojan può dunque manipolarle senza per questo inficiare il processo di autenticazione.

4.2 Tecniche di infezione

La prima fase di un attacco MITB è l'infezione del computer della vittima. Diverse tecniche si sono rivelate efficaci, molte delle quali si basano su semplici trucchi di ingegneria sociale, utilizzati per indurre l'utente ad agire in maniera sconsiderata. In questo caso l'utente riceve un'email che suggerisce di visitare un certo sito specificando una motivazione accattivante come una notizia particolarmente interessante, la possibilità di scaricare software gratuito o immagini e video di celebrità. Queste email sono differenti rispetto alle email tradizionali contenenti attacchi di tipo phishing in quanto non vengono inviate con la pretesa di provenire da un'istituto finanziario con l'obiettivo di raccogliere credenziali di accesso; in questo caso l'obiettivo è l'installazione di un malware.

Cliccando sul collegamento nell'email l'utente è rediretto verso un sito malevolo dove viene proposto il download di un software infetto contenente il trojan sotto forma di un codec o di un plugin necessario per la corretta visualizzazione di un video o della pagina oppure ancora come un PDF interessante o un pacchetto software illegalmente distribuito. All'apertura del file scaricato viene eseguita l'installazione del trojan mentre l'utente rimane del tutto inconsapevole.

Questo tipo di attacco richiede una certa interazione con l'utente che può diminuire la percentuale di successo. Una tecnica più sofisticata sfrutta invece una vulnerabilità presente nel browser utilizzato per la navigazione o in altre componenti del sistema per raggiungere lo stesso scopo. Il cybercriminale dapprima ottiene accesso ad un sito legittimo, ad esempio individuando e sfruttando una vulnerabilità nell'applicazione Web, e in seconda battuta inietta nel codice del sito compromesso un iFrame. Quando un utente visita il sito l'iFrame carica silenziosamente una pagina esterna, ospitata in un qualche nodo di una *botnet*, contenente una serie di punti di infezione (forniti dai

cosiddetti *exploits kits*) programmati per scaricare e installare il trojan sulla macchina dell'ignaro utente sfruttando punti deboli nella sicurezza del browser.

Un buon esempio di quest'ultima tipologia di infezione (denominata anche *drive-by download* o *drive-by install*) risale all'Aprile del 2009 quando una pagina realizzata dai fan del celebre musicista Paul McCartney's fu vittima di un attacco *hacker* per ben tre giorni durante i quali i visitatori vennero infettati da una variante di un trojan MITB [31; 45] (vedi Figura 4.4). In quel caso i cybercriminali sfruttarono la coincidenza con un importante concerto per garantire il numero più elevato possibile di infezioni dato l'aumento del traffico verso il sito generato dall'evento.

Figura 4.4: Spesso siti legittimi sono usati per veicolare malware (Fonte RSA [45])

L'enorme popolarità dei nuovi siti di social networking ha ulteriormente contribuito al proliferare di questo tipo di malware dati l'elevato traffico e la raggiungibilità globale di questi siti che li hanno resi un perfetto veicolo per l'infezione. Inoltre l'intrinseca socialità di queste reti fornisce una copertura ai vari tentativi di ingegneria sociale che possono essere perpetrati con un livello inferiore di sospetto da parte della vittima.

L'Anti-Phishing Working Group (APWG) monitora costantemente l'andamento globale delle infezioni di malware attraverso l'analisi dei report provenienti da numerose fonti. Secondo un recente report [69] la percentuale media di computer infetti da malware si attesta intorno al 30%. Tra i vari esemplari classificati l'80% è risultata essere

un trojan. Il report non cita specificatamente la situazione Italiana ma è comunque possibile dedurre dalle cifre un livello di infezione inferiore al 20% nel nostro Paese.

4.3 Misure di protezione

Nel 2006 Gühring [70] ha riassunto diverse idee e soluzioni concettuali per risolvere o quanto meno mitigare il problema degli attacchi MITB. Di seguito proponiamo quelle più interessanti rapportandoci allo scenario attuale. Come vedremo buona parte delle soluzioni effettivamente realizzate soffre di un'eccessiva tendenza al compromesso che inevitabilmente ne inficia le caratteristiche di sicurezza.

Hardened Browser I browser Web si sono evoluti da semplici software per il rendering di singoli documenti a veri e propri ambienti per l'esecuzione di svariati programmi, in maniera molto simile, concettualmente, ad un sistema operativo. L'architettura dei Web Browsers non si è evoluta allo stesso ritmo e non può gestire i requisiti di sicurezza di un ambiente di esecuzione così complesso [95], tanto più che spesso questi requisiti sono in conflitto con l'usabilità e la flessibilità che sono invece alla base delle priorità e della domanda degli utenti.

Alcuni requisiti per rendere più sicura l'implementazione di un browser sono:

- Impossibilità di installare estensioni o plugin (BHO, Active-X, Java, ...)
- Compilato staticamente e senza informazioni di debug (*stripped*)
- Accesso esterno al DOM vietato
- Accoppiamento stretto con il sottosistema crittografico (nessuna modularità tra il *core* del browser e il motore SSL)
- Ulteriori metodi di protezione del codice binario (eseguibile criptato, *packing*, ...)
- Utilizzo di UserScript disabilitato su siti protetti da SSL

Per rafforzare ulteriormente queste misure di sicurezza un client robusto dovrebbe permettere la sola esecuzione di richieste HTTPS (con la funzionalità HTTP pura non più compilata nel browser) e evitare l'utilizzo di sistemi di *caching* su disco. Un passo ulteriore sarebbe l'inclusione di alcuni specifici strumenti per il miglioramento della

sicurezza (ad esempio TrustBar [25], Trusteer Rapport[26], ...). Un browser "rafforzato" di questo tipo potrebbe essere installato sulla macchina dell'utente in parallelo al normale browser insicuro o, meglio ancora installato su un dispositivo di memorizzazione esterna da collegarsi alla macchina solo quando sia necessario visitare pagine che richiedono un alto livello di sicurezza.

Anche se questo approccio è tecnicamente percorribile, un istituto finanziario che volesse adottarlo dovrebbe agire come distributore del software con tutti i problemi che ne derivano. La portabilità del software potrebbe essere un problema per quegli utenti che utilizzano un sistema operativo non supportato. Inoltre il server per le operazioni bancarie non può forzare, in maniera affidabile, l'utilizzo di un tale browser o differenziare correttamente tra accessi sicuri e non.

Ambiente di esecuzione in sola lettura Un'altra metodologia per contrastare la maggior parte delle minacce odierne alla sicurezza delle transazioni si basa sull'utilizzo delle cosiddette "distribuzioni live". Si tratta di ambienti di esecuzione avviabili da supporti di memorizzazione in sola lettura come CDROM o DVD che forniscono un vero e proprio sistema operativo dotato, tra le altre funzionalità, di software per la navigazione. Alcuni esempi sono Knoppix [13], BartPE [2] e FreeBSD [12]. In alternativa la maggior parte delle distribuzioni Linux al giorno d'oggi fornisce un ambiente minimale di esecuzione accompagnato al CD di installazione.

C'è un forte problema di usabilità relativamente a questa soluzione. Infatti interrompere il proprio lavoro, riavviare la propria postazione, perdendo fino a 10 minuti potrebbe essere completamente inaccettabile per una buona parte degli utenti. Inoltre si deve anche considerare la possibilità di dover formare gli utenti all'utilizzo del sistema alternativo (basti pensare ad un utente abituato ad usare un sistema operativo Microsoft). Gühring [70] inoltre pone l'accento sul fatto che questo sistema ipotizza che il BIOS della piattaforma non sia stato compromesso, scenario verso il quale si potrebbe focalizzare la ricerca dei cybercriminali dovesse questo approccio diffondersi sufficientemente. Un altro problema è relativo al necessario aggiornamento della distribuzione nel caso in cui venissero individuate nuove vulnerabilità nel sistema operativo o nel browser installato, facendo così venir meno le caratteristiche di affidabilità dell'ambiente e incrementando i costi della soluzione.

Ambiente Virtualizzato Una soluzione che fornisce un compromesso di usabilità tra le precedenti è costituita dalle *virtual machine*, sistemi di esecuzione simulati che consentono di avviare un secondo sistema operativo alla stregua di una normale applicazione. Prodotti di questo tipo sono ad esempio VMWare Workstation [28], Oracle VirtualBox [18] e QEMU [20]. Questa soluzione, nel breve periodo, consentirebbe di aumentare il costo di un attacco ma, dovesse raggiungere un sufficiente grado di popolarità, potrebbe vedere aumentare i tentativi di compromissione grazie a tecniche di evasione dall'ambiente virtualizzato che sfruttino vulnerabilità del software di virtualizzazione [5; 6; 7; 8] o del sistema *host*. Inoltre è richiesto un certo grado di addestramento dell'utente senza contare la necessaria installazione di nuovo software.

Transaction Verification Con questo termine generico si indicano tutte quelle tecniche di sicurezza utilizzate per verificare che l'effettivo contenuto di una transazione ordinata attraverso Internet non sia stato alterato, caso tipico degli attacchi MITB. Alternativamente si utilizza il termine *Transaction Integrity Verification* (TIV). Non va confuso con *Transaction Authentication* con il quale si indica invece un metodo attraverso cui viene autenticata l'identità dell'utente al livello della transazione, senza però prevedere alcun controllo sull'integrità del contenuto della stessa. La verifica di una transazione avviene tipicamente su un secondo canale di comunicazione (*Out Of Band*, OOB) rispetto a quello utilizzato per l'invio della transazione. L'utente inserisce i dettagli della transazione come di consueto, utilizzando il proprio computer e il browser, sul sito web sicuro da cui intende operare. Una volta ricevuta la transazione il server invia automaticamente all'utente i dettagli della stessa attraverso il canale alternativo. Questo può essere un SMS, una chiamata telefonica automatica, una notifica inviata ad un'applicazione sullo smartphone. L'utente a questo punto verifica i dettagli. Se questi corrispondono a ciò che era nelle sue intenzioni l'utente dovrà inserire un codice (anch'esso ovviamente inviato attraverso il secondo canale) nella pagina di conferma e solo a questo punto la transazione verrà effettivamente processata. La sicurezza di questa procedura sta nel fatto che disaccoppiando il canale di immissione dei dati dal canale di autorizzazione si eleva notevolemente il costo dell'attacco in quanto è più complicato per un cybercriminale avere il controllo di entrambi. Inoltre sfrutta tecnologie già presenti sul mercato e non carica l'utente di un nuovo dispositivo di sicurezza. Ovviamente tutto ciò ha un costo sia operativo (costi degli SMS, telefonici, ...), per

l'azienda che intende utilizzare questo tipo di tecnologie per proteggersi sia in termini di una ridotta usabilità in quanto aggiunge ulteriore frustrazione all'utente e allunga il numero di operazioni e i tempi necessari per completare una transazione.

Recenti sviluppi [62] indicano come i cybercriminali stiano volgendo le loro attenzioni anche verso i nuovi dispositivi mobile. Di particolare interesse in questo campo sono i nuovi malware denominati *Man-in-the-Mobile* (MitMo). Due famosi esempi di questo tipo sono Spitmo [22] e ZeusMitmo [23], entrambi diffusisi nel 2010 e inizialmente sviluppati per cellulari equipaggiati con sistema SymbianOS. Questi software sono creati per funzionare in parallelo con una versione desktop del trojan in modo da aggirare la verifica delle transazioni. Quando l'utente visita il portale home banking tramite il suo browser compromesso viene accolto da un messaggio che spiega come, per misure di sicurezza aggiuntive, sia necessario installare un aggiornamento sul proprio cellulare e di indicare allo scopo il proprio numero telefonico. L'utente riceverà quindi un SMS con un collegamento per scaricare un aggiornamento, ovviamente fittizio. Una volta installatolo, il malware ispezionerà tutti i messaggi in arrivo e, nel caso venga individuato un SMS contenente un codice di autenticazione (detto Mobile Transaction Authentication Number o mTAN) inoltrerà tale SMS verso un numero di telefono controllato dai frodatori, aggirando di fatto il meccanismo di autorizzazione della transazione. Versioni di questi software sono state individuate recentemente anche per sistemi Android [30], Windows Mobile e BlackBerry [16]. Non si conoscono invece ad oggi casi di varianti sviluppate per il sistema iOS che equipaggia i telefonini Apple iPhone; è probabile che questa differenza sia dovuta principalmente alle diverse politiche di questo sistema relativamente all'installazione di applicazioni da fonti non ufficiali.

Un altro pericolo per questa strategia difensiva viene da un nuovo tipo di truffa che è stato escogitato appunto per aggirare proprio tali meccanismi di verifica, o quanto meno quelli che sfruttano le comunicazioni cellulari come canale alternativo: si tratta della cosiddetta *SIM swap fraud*. L'attaccante, venuto in possesso del numero cellulare e di altre informazioni personali della vittima, impersona quest'ultima contattando il servizio clienti di un operatore telefonico mobile e richiedendo il trasferimento del numero telefonico ad un'altra SIM card [14; 21]. Il risultato è la disabilitazione della SIM card originale che rende impossibile per la vittima ricevere ed effettuare chiamate o inviare SMS. Il codice di autorizzazione di un'eventuale transazione sarà così inviato

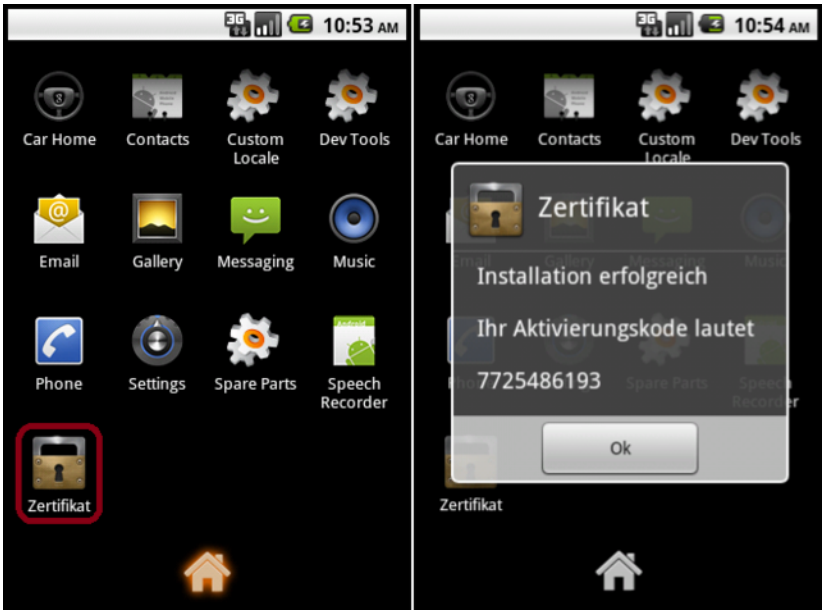

Figura 4.5: La versione Android di Zitmo cerca di mimetizzarsi come un aggiornamento per la sicurezza

alla nuova SIM card, controllata dal frodatore. Questo tipo di frode, diffusasi principalmente in Sud Africa e in altri paesi dove l'autorizzazione delle transazioni tramite SMS è molto utilizzata, è stata finora segnalata in combinazione con attacchi di tipo phishing nei quali l'utente aveva inconsapevolmente fornito le credenziali di accesso al proprio account al frodatore. Non è da escludere però che in futuro possa essere adottata anche nell'organizzazione di attacchi più evoluti come quelli MITB soprattutto se si considera la nascita dei nuovi servizi avanzati di call center all'interno del modello FaaS (vedi Sezione 3.2).

CAP Le SmartCard sono state utilizzate nell'ambito della sicurezza dei sistemi informativi come un metodo di autenticazione a più fattori: l'utente deve sia possedere fisicamente la SmartCard sia conoscere il relativo PIN per autenticarsi. Alcune banche hanno adottato le SmartCard, accoppiate ad un semplice dispositivo di lettura con un piccolo display, come alternativa ai dispositivi SecurID per generare delle password OTP. Come già detto però questo tipo di approccio è insufficiente quando si parla di attacchi MITB in quanto non prevede alcun controllo di integrità delle transazioni. Un tentativo in questa direzione è costituito dal sistema *Chip Authentication*

Program (CAP), sviluppato da MasterCard per autenticare utenti e transazioni tramite le SmartCard EMV [9], utilizzate comunemente anche come carte di debito. Questa implementazione prevede l'utilizzo di un lettore tascabile disconnesso, dotato di un tastierino e di un display (vedi Figura 4.6). L'utente accede al portale di home banking e abilita il lettore inserendo la propria SmartCard e digitando sul tastierino il proprio codice PIN. Successivamente egli utilizza il servizio online per inviare al server i dati della transazione che intende effettuare. Il server risponderà inviando un codice valido soltanto per quella specifica transazione. L'utente dovrà quindi digitare tale codice tramite il tastierino del lettore indicando nuovamente anche altri importanti dettagli della transazione come l'importo e il numero di conto destinatario (o parte di esso). Il dispositivo a questo punto genera un codice OTP tramite un algoritmo crittografico che combina questi dati con una chiave sicura memorizzata nella SmartCard. Inserendo il codice visualizzato sul display all'interno dell'apposito campo nell'interfaccia del portale l'utente conferma infine la transazione.

Figura 4.6: Un dispositivo CAP con una personalizzazione *Barclays*

Questa metodologia permette di combinare l'autenticazione a più fattori con la verifica OOB della transazione in quanto il dispositivo di generazione del codice OTP è scollegato (disaccoppiato) dal canale principale di comunicazione di modo che nessun malware può interferire con esso. Inoltre l'inserimento manuale degli importanti dettagli

della transazione consente di validare l'integrità della stessa di fatto rendendo vani i tentativi di modificare i dati della transazione all'insaputa dell'utente.

Sono state però evidenziate alcune criticità di questo approccio dovute a scarsa attenzione in merito ad alcuni dettagli implementativi. In particolare è rilevante il lavoro di ricerca effettuato dal Computer Laboratory della University of Cambridge [27] che si è concentrato nell'implementazione del sistema CAP offerta da alcuni istituti finanziari nel Regno Unito. In [57] Drimer et al indicano alcune debolezze che rendono i sistemi da loro analizzati deboli ad attacchi di ingegneria sociale dimostrando inoltre un aumento effettivo della superficie di attacco combinando l'autenticazione ai sistemi POS e quella online. Esistono altre implementazioni CAP che fortunatamente risolvono la maggior parte di questi problemi. Tra questi il sistema HDD 1.3 utilizzato da ZKA in Germania [**?**] che però richiede più input da parte dell'utente.

Un altro esempio su questo fronte è costituito dal sistema CrontoSign di Cronto [4] che permette di generare un criptogramma visuale (vedi Figura 4.7(a)), un'immagine contenente, in forma codificata, i dettagli della transazione. L'immagine potrà essere decodificata tramite un opportuno software installato sul telefonino, dotato ovviamente di fotocamera (nuova possibilità di MitMo?), oppure su un dispositivo dedicato (vedi Figura 4.7(b)). Il software mostrerà i dettagli della transazione all'utente e genererà dunque un codice che l'utente dovrà inviare al server per confermare la transazione.

Simile destino è toccato anche ad un'altra tecnologia di protezione, adottata principalmente in Germania; il sistema chipTAN. Per confermare una transazione protetta con questo sistema all'utente viene mostrato nel browser un particolare codice a barre in bianco e nero. Questo codice a barre va decodificato attraverso l'utilizzo di un dispositivo dedicato, disconnesso, nel quale deve essere inserita una SmartCard collegata all'utente tramite un PIN (Figura 4.8). Dopo aver avvicinato il dispositivo allo schermo e averlo correttamente allineato verranno visualizzati sul display del lettore i dettagli della transazione e, una volta confermati, il TAN per completare il processo. Come accennato però anche questa implementazione è stata oggetto di analisi che hanno fatto emergere alcune serie vulnerabilità [68]; inoltre recentemente il trojan *Tatanga* [24] ha aggirato con successo il sistema tramite classici attacchi di ingegneria sociale.

Come abbiamo visto esistono diversi sistemi di questo tipo, ognuno dei quali cerca di implementare una forma di transaction verification che consenta all'utente di verificare l'integrità della transazione. Le varie soluzioni costituiscono un compromesso tra

(a) Un esempio di criptogramma visuale utilizzato nei sistemi CrontoSign

(b) Un dispositivo dedicato dotato di fotocamera per la decodifica del criptogramma

Figura 4.7: Il sistema CrontoSign

Figura 4.8: Un dispositivo ChipTAN durante la scansione dello speciale codice a barre

usabilità, costi e sicurezza reale ottenuta. Alcune (come il CAP nel Regno Unito) favoriscono l'usabilità cercando di costringere l'utente a reintrodurre il minor numero di informazioni ma effettivamente diminuendo la sicurezza complessiva. Altre (è il caso di HDD 1.3) puntano maggiormente su quest'ultimo aspetto ma degradano la qualità dell'esperienza utente aumentando anche la possibilità di errori nella digitazione dei vari dati. Sul fronte dell'usabilità una soluzione come quella proposta da Cronto evita all'utente la digitazione ma necessita ancora dell'installazione di un'apposita applicazione nello smartphone (con tutte le problematiche legate ai malware MitMo che abbiamo discusso) o l'uso di un dispositivo aggiuntivo il cui utilizzo eleva i costi di questo sistema. Quest'ultima considerazione va necessariamente fatta per tutti i sistemi CAP. Concludiamo facendo notare che esistono implementazioni CAP non disconnesse che prevedono il collegamento con il PC dell'utente tramite la comune interfaccia USB, caratteristica che mira a migliorare l'usabilità ma che introduce inevitabilmente un nuovo punto debole. Si veda [38] per un recente lavoro di analisi che ha evidenziato diverse vulnerabilità di un dispositivo di questo tipo utilizzato nei Paesi Bassi.

CAPTCHA Alcuni portali di home banking utilizzano sistemi di protezione, come ad esempio iTANplus (vedi Figura 4.9), che prevedono la visualizzazione dei dettagli della transazione e il codice di verifica all'interno di un CAPTCHA, un'immagine appositamente creata per rendere difficoltosa la decodifica del contenuto da parte di un software ma allo stesso tempo semplice per un essere umano. Questi sistemi si basano sulla premessa che non sia semplice per un software manipolare il testo contenuto all'interno dell'immagine ma questo non è strettamente necessario per condurre un attacco MITB o MITM dove l'immagine può essere semplicemente sostituita lato client dal trojan. Inoltre è eventualmente possibile per il frodatore redirigere l'immagine di modo che venga decodificata usufruendo di un intervento umano. Purtroppo però anche l'ipotesi di difficile manipolazione si è rivelata essere infondata. Esistono numerosi lavori di ricerca che hanno provato come i CAPTCHA non siano efficaci nel differenziare tra umani e computer. In particolare [82] si concentra sulle varie tecniche di protezione che sfruttano i CAPTCHA nei servizi bancari online dimostrando un'efficacia del 100% nei vari attacchi sviluppati. Un altro importante aspetto da valutare è la scarsa ergonomia e usabilità di questi sistemi in quanto spesso i CAPTCHA risultano di difficile lettura da parte di persone con disabilità visive e non solo.

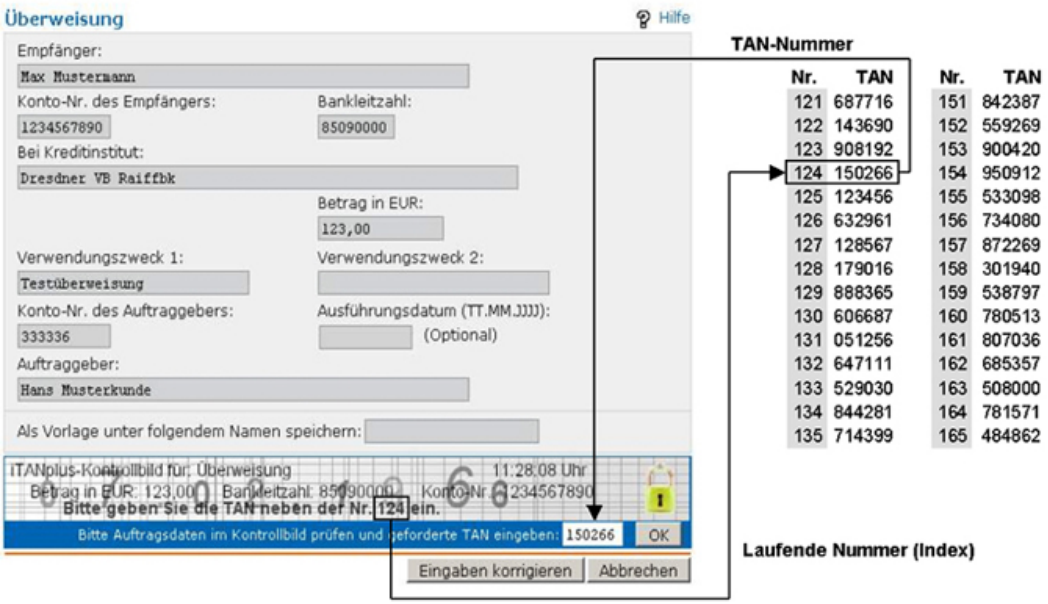

Figura 4.9: Un CAPTCHA iTANplus. È possibile notare la scarsa ergonomia di questo sistema

5

Analisi dei dati

In questo Capitolo discutiamo le caratteristiche, o anche *features*, individuate dai dati del dominio, per la costruzione dei profili descrittivi degli utenti. Di ogni *feature* esploreremo i motivi per i quali è stata selezionata o, eventualmente, scartata come fonte di contributo informativo.

5.1 Due principali sotto-domini

Esaminando le sorgenti di dati nel nostro scenario possiamo individuare due fondamentali sotto-domini:

- dominio dei dati di navigazione
- dominio dei dati transazionali

Nel seguito descriveremo entrambe le tipologie di dati fornendo una valutazione puntuale.

5.1.1 I dati di navigazione

Il primo dei due sotto-domini rappresenta tutto quell'insieme di dati storici e non che sono a disposizione negli archivi e che sono costituiti dai file di log dei web server utilizzati per fornire i servizi di home banking. Tali file sono piuttosto corposi (quasi 4GB al giorno mediamente) pertanto è necessario individuare una strategia che ricavi da questi file soltanto l'informazione strettamente necessaria per catturare il comportamento cosiddetto "navigazionale" degli utenti.

5.1.1.1 Formato dei dati

Come già accennato nella sezione precedente i dati di navigazione sono memorizzati in origine sotto forma di log prodotti dai web server. Il formato di tali log è ovviamente configurabile ma si è optato per il mantenimento di quello già predisposto anteriormente allo sviluppo del sistema antifrode, in quanto considerato sufficiente a registrare una varietà di informazioni adatta per gli scopi previsti. Inoltre mantenere il formato ha permesso di disporre di una corposa mole di dati storici.

Il formato è descritto da un parametro di configurazione del web server e nel nostro caso si presenta come di seguito (si noti che viene qui visualizzato su più righe per ragioni di impaginazione). In Tabella 5.1 è riportato concisamente il significato accanto a ciascuno dei segnaposti contenuti nella riga di configurazione.

```
LogFormat "\"%a\" %l %u %t \"%r\" %s %b \"%{Referer}i\"
\"%{User-agent}i\" \"%{CODICE_UTENTE}i\" %D %V" nome_formato
```

Buona parte dei dati contenuti nei log non è di alcuna utilità pratica per gli scopi del sistema. È necessario perciò analizzare il contributo in termini informativi di ciascuna porzione. Di seguito esponiamo l'analisi compiuta preventivamente per ciascun campo eccezion fatta per i due quelli corrispondenti ai segnaposti %u e %l in quanto non sono mai valorizzati nel nostro contesto applicativo.

Codice utente Particolare attenzione va posta nel campo indicato dal segnaposto "codice utente": si tratta di una stringa che contiene l'identificativo utilizzato dall'utente per effettuare l'accesso al servizio di *home banking*. Tale campo, nella riga di log, viene valorizzato solamente se l'accesso ha avuto un buon esito e consente di semplificare l'attribuzione di ogni singola richiesta ad un particolare utente riducendo la necessità di far ricorso a poco affidabili euristiche. In Sezione 7.3.1 verrano discussi ulteriori dettagli relativi a questo problema implementativo.

Status code L'informazione fornita dallo *status code* della risposta HTTP è anch'essa poco utile in quanto nella maggior parte dei casi il valore indicherà la corretta esecuzione della richiesta anche in caso di frode dato appunto il tentativo del software malevolo di mimare l'esecuzione di un utente reale. Le condizioni di errore si possono verificare in

Segnaposto	Descrizione
`%a`	rappresenta l'indirizzo IP remoto dal quale è originata la richiesta HTTP al web server
`%l`	questo campo non viene mai valorizzato
`%u`	questo campo non viene mai valorizzato
`%t`	data e ora di ricezione della richiesta
`%r`	la richiesta, divisa in metodo, URI e protocollo/versione
`%s`	il codice della risposta del server, come specificato dal protocollo HTTP
`%b`	la grandezza in byte della risposta, non considerando gli header
`%{Referer}i`	il Referer, così come specificato dal protocollo HTTP
`%{User-agent}i`	la stringa utilizzata dal browser dell'utente per identificarsi
`%{CODICE_UTENTE}i`	un codice che identifica univocamente l'utente che ha effettuato la richiesta corrispondente.
`%D`	il tempo, specificato in microsecondi, necessario a servire la richiesta
`%V`	il nome del web server

Tabella 5.1: Descrizione del formato di log del web server

caso di problemi lato server oppure qualora la risorsa richiesta non sia più disponibile, eventualità queste che non ci permettono comunque di avanzare ipotesi di frode o di migliorare significativamente la nostra confidenza sul fatto che una frode sia in atto.

Richiesta Il campo %r contiene la prima riga di ogni richiesta HTTP. Questa riga è composta da 3 sotto-campi: metodo, URL, nome e versione del protocollo utilizzato. Diamo per scontato che il metodo e la specifica del protocollo e della sua versione non possano essere modificati senza inficiare la comunicazione stessa con l'applicazione web. Forti di questa assuzione ci accontentiamo pertanto di estrarre la sola URL che rappresenta la risorsa richiamata dall'utente ed è quindi fortemente legata alla semantica della navigazione.

Referer Il campo *referer* (sic) dell'*header* HTTP, automaticamente impostato dal browser, indica, dal punto di vista di una pagina o di una risorsa, l'indirizzo URL della risorsa a cui è collegata; esaminando il contenuto del *referer* un'applicazione web può individuare il punto d'origine di unaq richiesta. La situazione più emblematica è quella di un utente che clicchi un collegamento ipertestuale all'interno di una pagina. In seguito al click il browser dell'utente invia una richiesta corrispondente all'indirizzo della pagina collegata; il valore del *referer* sarà l'indirizzo della pagina contenente il collegamento. Il controllo di questo campo viene spesso utilizzato nelle applicazioni di reportistica per stilare liste di pagine esterne collegate ad un sito web o per individuare le parole chiave che gli utenti utilizzano per raggiungere un sito attraverso i motori di ricerca.

Ad una prima analisi il contenuto di questo campo può sembrare effettivamente utile per riconoscere un tentativo di frode. Ad esempio l'osservazione di una richiesta con un *referer* corrispondente ad una pagina non precedentemente visitata potrebbe essere percepita come un indicatore della presenza di una manipolazione del traffico, dovuta ad un agente malevolo. Inoltre un'analisi della sessione utente che tenga conto del *referer* potrebbe facilmente individuare "salti" nella navigazione che non sono previsti dal funzionamento dell'applicazione.

Sfortunatamente ci sono importanti ragioni che consigliano prudenza nel trattare questo dato. Innanzitutto non si tratta di un campo obbligatorio del protocollo HTTP [1], anzi la quasi totalità delle applicazioni web non fa alcun uso di questo dato se non per fini statistici, di monitoraggio o di ottimizzazione. Inoltre, dati i rischi per la privacy che sono direttamente correlati con la valorizzazione di questo campo, alcuni prodotti sviluppati per la sicurezza dei sistemi desktop (le cosiddette *Internet-security suites*) inibiscono in maniera predefinita il suo invio modificando il traffico HTTP in uscita ed eliminandolo dall'intestazione. Lo stesso tipo di opzione è disponibile in diversi browser o attivabile tramite l'installazione di componenti esterni come estensioni o *plugins*.

User-Agent Questo campo contiene una sorta di firma del browser che ha effettuato la richiesta. Di seguito un esempio tipico, riportato su più righe per ragioni di spazio:

```
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; .NET CLR 1.1.4322;
.NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
```

[1] si veda `http://www.w3.org/Protocols/HTTP/HTRQ_Headers.html`

In questa stringa troviamo varie informazioni circa il browser utilizzato e la sua specifica versione ma anche dati legati al tipo e alla versione di sistema operativo oltre che di alcuni componenti dello stesso. Si può intuire come un improvviso cambiamento del valore del campo User-Agent all'interno di una singola sessione sia un forte segnale di allarme che dovrebbe contribuire fortemente all'innalzamento del sospetto verso le transazioni avvenute nella sessione stessa, in special modo se si osserva un'alternanza ripetuta tra diverse "firme". In realtà purtroppo questa è un'eventualità piuttosto remota; solo un agente malevolo sviluppato molto ingenuamente effettuerebbe una modifica al valore del campo *user-agent* dato mentre è più probabile che mantengano quello automaticamente inserito dal browser. Pertanto consideriamo quest'ultimo caso come banale e ne prevediamo l'individuazione, per completezza, attraverso un sistema separato il cui output sarà successivamente integrato (cfr. Sezione 5.1.1.3).

Nome del web server Anche il nome del web server non è significativo per quanto concerne la capacità di discriminare le richieste malevole dato che il valore del campo è fissato a priori dalla configurazione dello stesso server; esso risulta però molto utile per filtrare le richieste non pertinenti il servizio di *home banking* che vengono comunque a trovarsi nel log e per partizionare le richieste a seconda del cliente-banca interessato. Ad ognuno di essi infatti corrisponde un preciso nome a dominio[1] che si riflette proprio in questo campo.

Data e ora Su questo campo non ci dilunghiamo; è necessario disporre di questo dato per valutare inizio e fine di una sessione, per individuare sequenze di richieste troppo rapide (spesso segno di attacchi automatizzati). Poter riconoscere eventuali connessioni in orari sospetti rispetto al solito potrebbe essere inoltre un valore aggiunto per il sistema antifrode. Va però ricordato che il successo degli attacchi MITB dipende dalla manipolazione di una sessione legittima della vittima ed essi avvengono perciò in concomitanza.

Indirizzo IP remoto L'indirizzo IP remoto è l'indirizzo, unico sulla rete Internet, da cui ha avuto origine una particolare richiesta. Come abbiamo già detto la "proprietà" di ogni singola richiesta è specificata dal campo "codice utente". Purtroppo tale campo non è sempre valorizzato (Cfr. 7.3.1) e risulta perciò utile tenere traccia dell'indirizzo

[1] nel senso di dominio DNS

IP per garantire una corretta etichettatura di tutte le richieste. Inoltre l'indirizzo IP può essere utilizzato per determinare l'origine geografica di una connessione sebbene con una precisione piuttosto carente ma sufficiente per i nostri scopi. Ad esempio connessioni a distanza ravvicinata da Paesi geograficamente distanti sono spesso un chiaro segno di compromissione di un *account*.

Il risultato di questa analisi ci consente di ridurre il numero di informazioni di interesse ricavabili dai log di un web server riducendo l'attenzione a 4 particolari campi:

- Indirizzo IP remoto
- Data e ora della richiesta
- URL della richiesta
- Codice utente

Si noti che cercare di minimizzare la quantità di informazioni di cui tener conto è utile anche a livello implementativo in quanto semplifica le elaborazioni, riduce la quantità di dati da organizzare e memorizzare e di conseguenza rende la soluzione proposta più adatta ad esecuzioni *real-time* o *near real-time*, obiettivo a cui deve puntare un sistema antifrode del tipo *transaction monitoring*. È consigliato ovviamente conservare tutte le informazioni, come del resto le grosse aziende già fanno, in log appositi e separati dal sistema antifrode, per un periodo sufficientemente lungo (almeno 30 giorni) di modo che sia possibile, in caso di sospetta frode, approfondire manualmente, o con appositi strumenti software, l'analisi dell'*incident*.

5.1.1.2 Rumore nei dati

Abbiamo discusso le informazioni relative alla navigazione che vogliamo utilizzare per modellare il comportamento degli utenti. La descrizione fatta finora non tiene però conto del pesante rumore che è caratteristica frequente in questi log. Infatti una grossa porzione delle richieste che vengono effettuate al web server sono relative ad immagini o ad altre risorse statiche (file JavaScript o Flash, fogli di stile CSS, documenti XML, ...). Tener conto di queste risorse come parte integrante del modello di navigazione di un utente avrebbe conseguenze deleterie sia sulla qualità del modello sia sulla dimensione e quindi gestibilità dello stesso.

In primo luogo le risorse statiche di un sito web sono in numero molto elevato; ciò causa un'esplosione delle richieste da analizzare senza per altro migliorare la capacità del sistema di catturare la semantica della navigazione di un utente, soprattutto all'interno di una tipologia di sito Internet, quale è un portale di home banking, in cui le risorse statiche vengono raramente richieste direttamente dall'utente ma quasi esclusivamente richiamate indirettamente dal browser come risorse necessarie alla corretta visualizzazione della pagina selezionata (ad esempio le immagini di loghi aziendali o i fogli di stile per il rendering dei vari elementi grafici). Data poi la stretta correlazione tra file statici e *layout* di un sito web si ha che anteporre un filtro all'analisi rende più robusto il sistema in caso di cambiamenti grafici dell'interfaccia che non impattano sulla struttura dei collegamenti tra le pagine. Un altro importante problema consegue dal fatto che spesso tali risorse sono conservate nella *cache* (cfr. Sezione 7.3.1.2) del browser dell'utente (o di un *caching proxy server* qualora sia presente) rendendo la mancanza della chiamata ad una risorsa statica non definibile chiaramente come evento sospetto o meno e di fatto abbassando l'affidabilità di un qualsiasi modello che tenesse in considerazione anche di queste ultime.

Un'altra fonte di rumore è rappresentata dal formato degli indirizzi web nel protocollo HTTP, le cosiddette URL. Questo formato è descritto dalla seguente grammatica:

```
protocollo://<username:password@>nomehost<:porta></percorso><?querystring>
```

Per quanto riguarda la stragrande maggioranza delle applicazioni Web le componenti *username*, *password* e *porta* non compaiono. Sono intuibili le ragioni di sicurezza nel caso delle prime due mentre per quanto riguarda il numero della porta di collegamento questo non viene esplicitato qualora venga utilizzato il valore predefinito (porta 80 per HTTP, 443 per HTTPS) ma anche nel caso in cui questo non si verificasse, per precisa scelta degli sviluppatori, il dato rimarrebbe chiaramente invariato in caso di manipolazione delle richieste da parte di un frodatore. Lo stesso argomento vale per la porzione iniziale dell'URL che specifica il protocollo utilizzato dalla richiesta. Possiamo dunque implementare un primo filtro e ridurre il formato iniziale a quello seguente:

```
nomehost</percorso><?querystring>
```

La componente *querystring* rappresenta tutta quella serie di parametri che vengono forniti dall'utente. Questa lista di parametri può essere anche piuttosto lunga e

contenere sia informazioni circa la semantica della pagina richiamata sia parametri aggiuntivi utilizzati dall'applicazione lato server per determinarne la "forma" (ad esempio il linguaggio da utilizzare o la quantità di movimenti del conto corrente da visualizzare). La *querystring* è il meccanismo principale attraverso il quale le applicazioni web dinamiche possono reagire ai diversi input dell'utente permettendo di implementare logiche di funzionamento basate sul valore dei parametri forniti. Proprio per questo motivo moltissimi attacchi a queste applicazioni si basano sulla manipolazione di tali parametri per indurre l'applicazione ad operare in maniera non prevista, basti pensare agli attacchi *SQL Injection* o *reflected XSS*. Ne deriva quindi che gli IDS progettati per garantire la sicurezza dei server web, specialmente quelli di tipo *misuse detection*, dispongono di numerose *signature* che analizzano la presenza di determinati pattern all'interno della *querystring*. Analogamente anche molti sistemi di *anomaly detection* sviluppati per monitorare il traffico HTTP [80; 81] concentrano l'attenzione su questa porzione dell'URL in questo caso però cercando di modellare, attraverso diverse techiche, l'insieme di valori legittimamente attesi.

È necessario dunque tenere conto di questi parametri? La risposta purtroppo non è univoca e dipende da alcune scelte di sviluppo dell'applicazione stessa che vogliamo monitorare. Se il ruolo di questi parametri determina la semantica della richiesta allora è quanto meno necessario effettuare una valutazione dell'applicazione, in collaborazione con gli sviluppatori, per individuare le varie casistiche e produrre una lista delle coppie pagina/parametro che consenta di filtrare solo le informazioni necessarie per descrivere il comportamento dell'utente. Consideriamo a titolo di esempio una sessione composta dalle due seguenti URL:

```
http://www.miabanca.it/azione.do?pagina=Movimenti&quantita=10
http://www.miabanca.it/azione.do?pagina=Saldo
```

Eliminare da questa applicazione l'intera *querystring* a scopo di semplificare il monitoraggio comporterebbe una grave perdita di informazione e porterebbe a considerare equivalenti le due richieste. In questo caso infatti la semantica è totalmente catturata dal parametro "pagina" il cui contenuto dovrebbe essere considerato; al contrario il parametro "quantita" fornisce un'informazione trascurabile.

Abbiamo finora considerato solamente parametri inviati attraverso la *querystring*. Questo tipo di "passaggio" di parametri avviene frequentemente nel caso di richieste

che utilizzano il metodo HTTP GET. Nel caso però in cui parametri significativi vengano spediti anche facendo uso del metodo HTTP POST sarà necessario provvedere la registrazione di questi specifici parametri nei file di log dato che non vengono infatti memorizzati in maniera predefinita o, se questo dovesse risultare complicato e richiedere una configurazione del web server troppo elaborata, prevedere la memorizzazione dell'intero contenuto delle richieste POST, con tutte le problematiche annesse dovute all'incremento delle dimensioni dei dati da archiviare.

L'applicazione monitorata dal nostro sistema non ha richiesto interventi di questo tipo, essendo la semantica di ogni singola richiesta ben definita dal solo nome della pagina acceduta, contenuto nella componente *percorso* di ogni URL. Ci rendiamo però conto che nello sviluppare un sistema più generale sia preferibile posizionare la componente di analisi e scomposizione delle richieste di modo che possa direttamente intercettare tutto il traffico HTTP verso il web server senza richiedere un'analisi a posteriori dei file di log. Questo tipo di implementazione potrebbe anche meglio soddisfare quei requisiti di esecuzione real-time che sono sempre più desiderabili per i sistemi antifrode e avere un impatto nullo relativamente alla configurazione del web server che non dovrebbe mai essere modificata o aggiornata. In Figura 5.1 è possibile osservare una rappresentazione schematica proprio di questa architettura che sfrutta una funzionalità nota come *port mirroring*, di cui sono dotati gli switch di livello *enterprise*, per inoltrare una copia dei pacchetti smistati dallo *switch* stesso verso una porta opportunatamente dedicata, allo scopo di monitoraggio.

5.1.1.3 Integrazione con IDS esistenti

Diverse aziende, per fronteggiare la minaccia costituita dagli attacchi informatici esterni o interni, hanno introdotto negli anni all'interno delle proprie reti dei prodotti IDS per analizzare il traffico o altri aspetti della propria infrastuttura di rete. Nel contesto aziendale in cui si è svolto questo studio ci si è potuti avvalere del supporto di un *Web Application Firewall* (WAF) in grado di intercettare tutte le richieste HTTP provenienti dall'esterno. Questo WAF è stato configurato per filtrare tutte le richieste verso risorse non esistenti, richieste malformate o con parametri specificati in maniera sostanzialmente diversa da quella abituale per quel tipo di risorsa (grazie ad un sistema interno e completamente autonomo basato su tecniche di *machine learning*). Inoltre dispone di una propria lista di *signature* relative a tipologie tipiche di attacchi rivolti ad

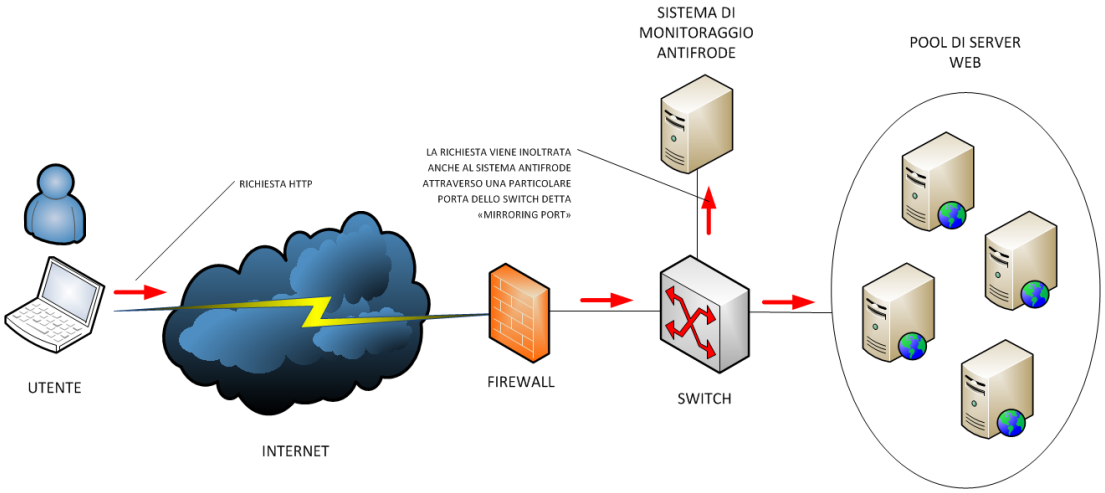

Figura 5.1: Esempio di architettura adatta al monitoraggio anti-frode *real-time*

applicazioni web quali *SQL Injection* o *XSS*. Nonostante non ci aspettiamo questo tipo di eventi nel caso di frode MITB, dove comunque l'attività di navigazione registrata non si discosta eccessivamente da quella abituale e non presenta quindi alterazioni vistose nella forma delle singole richieste (infatti l'obiettivo dell'attaccante è sfruttare le possibilità legittime dell'applicazione per i propri scopi, non forzare l'applicazione verso un comportamento per cui non era stata inizialmente sviluppata), pensiamo comunque di poter fare buon uso anche delle informazioni provenienti da questo sistema. Ad esempio si è potuto configurare il WAF in modo da individuare se l'utente utilizza due differenti *user-agent* all'interno di una singola sessione (cfr. Sezione 5.1.1.1). Benchè questo tipo di informazione fosse ottenibile anche con una opportuna analisi dei log dei web-server, la difficoltà implementativa e la necessità di memorizzare un quantitativo ingente di ulteriori informazioni (spesso peraltro ridondanti) hanno condotto all'esternalizzazione di questa caratteristica, demandandola al sistema già esistente grazie ad una semplice modifica nella sua configurazione.

5.1.1.4 Utilizzi alternativi dei dati

Nelle Sezioni 5.1.1.1 e 5.1.1.2 abbiamo espresso i diversi argomenti che hanno determinato la scelta di escludere dalla creazione di un modello di navigazione alcune informazioni tra quelle disponibili. Vogliamo però qui proporre un utilizzo alternativo per

alcuni di questi dati che possono contribuire ad irrobustire complessivamente il sistema di sicurezza.

Benché le richieste a risorse statiche di un sito web non offrano un contributo significativo nel modellare le attività di un utente esse possono però servire ad un altro tipo di analisi. Esaminare infatti il *referer*, soprattutto per le richieste di immagini e fogli di stile, è un valido strumento di controllo per svelare la presenza o lo sviluppo di attacchi di tipo *phishing* che mirano ad ottenere le credenziali di un utente ricreando una versione modificata ma all'apparenza genuina del sito originale. Questi attacchi necessitano allo scopo di tutte le risorse grafiche utilizzate dal sito web che i cybercriminali intendono mimare; spesso queste vengono semplicemente richiamate dal sito "civetta" (senza la presenza di una copia locale) di modo che il loro accesso verrà registrato sui server legittimi. È perciò possibile implementare un controllo automatizzato del *referer*, inoltrato con tali richieste, che evidenzi tutti quei casi in cui il dominio di provenienza non sia compreso in una *whitelist* preventivamente compilata. In questo modo sarà possibile individuare con anticipo l'esistenza di siti fraudolenti e agire di conseguenza attraverso i canali istituzionali per rapida chiusura, riducendo gli effetti negativi derivanti da una prolungata presenza online, ivi comprese le ricadute sulla sicurezza degli utenti e dei depositi.

5.1.2 I dati transazionali

Finora abbiamo visto una panoramica sui dati relativi al profilo di navigazione di un utente del sito web. Questo è certamente un aspetto importante ma non rappresenta l'unico da considerare. Bisogna infatti tener conto anche dell'ancor più importante aspetto relativo all'operazionalità dell'utente, intesa come tipologie e dettagli delle operazioni bancarie effettuate tramite il sito di home banking.

5.1.2.1 Attributi

Nello sviluppare il nostro sistema l'obiettivo era di individuare le frodi MITB perpetrate tramite trasferimento elettronico di denaro, in particolare tramite bonifico bancario. Un'operazione di questo tipo contiene diverse informazioni:

- Importo
- Identificativo del beneficiario

- Codice IBAN del conto corrente del beneficiario
- Causale del pagamento
- Data e ora dell'inserimento dell'operazione
- Codice utente di colui che effettua l'operazione

Oltre a questi campi il sistema informativo aziendale da noi analizzato prevede anche una serie di campi aggiuntivi utilizzati internamente che non sono però semanticamente correlati e non forniscono informazione aggiuntiva. Degno di nota è solamente il campo che rappresenta lo stato effettivo dell'operazione (se andata a buon fine oppure se in errore e per quale motivazione); d'ora in avanti considereremo nella nostra analisi solo le operazioni inserite con successo nel sistema.

Nel seguito dedichiamo una breve discussione relativa ai dati in nostro possesso. Mentre alcuni campi come l'importo non hanno bisogno di chiarimenti ci soffermeremo su alcuni più interessanti.

IBAN L'*International Bank Account Number* (IBAN), è una stringa alfanumerica, definita dallo standard internazionale **ISO 13616** del 1997 utilizzato per identificare univocamente un'utenza bancaria (un conto corrente). Un IBAN può comporsi di al più 34 caratteri (sono caratteri validi per l'IBAN le cifre numeriche da 0 a 9 e le 26 lettere maiuscole dell'alfabeto latino da A a Z) e si divide in tre blocchi:

- 2 lettere rappresentanti lo Stato (codificato secondo lo standard **ISO 3166-1 alpha-2**)
- 2 caratteri di controllo
- codice BBAN (*Basic Bank Account Number*) specifico per Stato

Il codice BBAN varia da Stato a Stato ma deve mantenere una lunghezza fissa fra i conti correnti della stessa Nazione. In Tabella 5.2 è possibile osservare alcuni esempi di codici IBAN di vari Stati Europei; i caratteri di spaziatura sono stati inseriti solo per migliorare la leggibilità, non fanno parte dei caratteri validi. Si noti come la porzione del codice relativa del BBAN sia piuttosto differente da uno Stato all'altro.

Paese	IBAN
Italia	IT 02 L 12345 12345 123456789012
Romania	RO 02 1234 1234567890 123456
Germania	DE 02 12345678 1234567890

Tabella 5.2: Esempi fittizzi di IBAN da diversi Paesi

Per i nostri scopi non è necessario conoscere la procedura di estrazione delle lettere di controllo in quanto qualunque IBAN che viene da noi analizzato è già passato attraverso un algoritmo di verifica della sua correttezza e pertanto si può considerare sempre valido.

Di tutti i possibili 34 caratteri dell'IBAN quelli più interessanti ai nostri fini sono i primi due, quelli che compongono cioè il codice dello Stato in cui è stato aperto il conto corrente del beneficiario dell'operazione dispositiva. Si deve ricordare che la maggior parte delle truffe del tipo sotto nostro esame avviene tramite un pagamento verso conto corrente estero. Nonostante questo complichi le procedure di verifica degli opportuni organi nazionali e internazionali e renda più difficile bloccare o annullare il pagamento una volta che è stato confermato ci permette però di focalizzare la nostra analisi sul numero, ovviamente ridotto, di pagamenti internazionali senza però trascurare quelli interni.

Codice utente Questo campo è la controparte del "Codice utente" corrispondente alle singole richieste HTTP che abbiamo incontrato in Sezione 5.1.1.1. Ci permette di correlare semplicemente le operazioni dispositive memorizzate all'interno del database di produzione con il contenuto dei log di navigazione.

Beneficiario È un campo testuale in cui l'utente che dispone l'operazione inserisce tipicamente nome e cognome del beneficiario se si tratta di un conto corrente privato oppure la ragione sociale se si tratta di un conto corrente aziendale o della pubblica amministrazione. Si tratta però di un campo il cui contenuto non è automaticamente verificato e costituisce pertanto principalmente un valore mnemonico per l'utente. Non è raro infatti che anche per diverse disposizioni verso lo stesso conto corrente una persona utilizzi identificativi differenti, scambiando ad esempio la posizione di nome e

cognome, modificando eventuali accenti o abbreviando in maniera diversa alcune componenti della ragione sociale in caso di aziende o PA. Dal punto di vista del nostro sistema un campo con queste proprietà non risulta affidabile per essere utilizzato senza sufficiente cautela come valore di discriminazione tra pagamento corretto e potenziale frode. Infatti essendo il suo inserimento completamente sotto il controllo dell'utente (e quindi anche dell'eventuale frodatore) e il suo valore non verificato all'atto della transazione un software MITB abbastanza sofisticato potrebbe approfittare di queste debolezze per associare un nominativo familiare all'utente ad un IBAN completamente diverso verso cui dirottare il pagamento in modo da rendere più difficoltosa la scoperta della frode ad un rapido esame dei movimenti bancari da parte dell'utente stesso. D'altro canto è sempre possibile che uno stesso nominativo sia associato a più un conto corrente (e quindi più di un IBAN) sia a livello nazionale che internazionale, basti pensare ad un'azienda di medie o grandi dimensioni. Il codice IBAN risulta quindi in definitiva l'unico parametro veramente affidabile attraverso cui identificare un'utenza beneficiaria.

Causale L'ultimo campo non banale dell'elenco è la "Causale" dell'operazione. Si tratta di una stringa che descrive in poche parole la natura del pagamento in oggetto come ad esempio "PAGAMENTO AFFITTO OTTOBRE 2012" o "ACQUISTO FERRARI F40"; anche questo campo come già visto per il "Beneficiario" è a completa discrezione dell'utente. Dai dati in nostro possesso possiamo notare come tipicamente i frodatori siano sufficientemente furbi da evitare l'utilizzo di causali in sospette lingue straniere ma preferiscano utilizzare l'italiano, anche se a volte non senza qualche percebile lacuna lessicale o grammaticale. Alcuni esempi reali, utilizzati nei casi di frode accertati da noi esaminati, sono:

- PAGAMENTO PREMIO ASSICURAZIONE
- LOCAZIONE SANCHEZ
- AMMINISTRATORE MESE DI GIUGNO
- DA SILVA REST.2012

Alcuni degli esempi sono linguisticamente ineccepibili, altri risultano meno immediati ma bisogna tenere a mente la natura del campo e la sua lunghezza limitata che

costringe spesso all'utilizzo di abbreviazioni rendendo la comprensione del messaggio non banale da parte di un osservatore esterno anche nel caso di campi perfettamente legittimi. D'altro canto l'uso di valori come "FATTURA" consentono al frodatore di passare praticamente inosservato senza eccessivi sforzi linguistici e in maniera del tutto ripetibile.

Data e ora La data e ora che vengono memorizzate sono quelle corrispondenti al momento dell'inserimento via web dell'operazione dispositiva, fatto che rende poi possibile nell'implementazione del nostro sistema correlare facilmente la disposizione con la sessione che l'ha effettivamente generata.

5.1.2.2 Descrizione statistica

In questa Sezione ci proponiamo di fornire alcuni dati circa la dimensionalità dei record dispositivi. Forniremo inoltre una serie di statistiche che saranno successivamente utili per valutare l'applicazione di certe euristiche al sistema anti-frode. Il dataset utilizzato per queste analisi è costituito da oltre 2.600.000 bonifici bancari effettuati attraverso il solo servizio di online banking nell'arco di un anno da 220.000 utenti.

Il grafico in Figura 5.2 mostra la distribuzione all'interno di una settimana tipo del numero di operazioni ordinate dagli utenti del sistema di home banking. Vediamo che il numero totale, intorno alle 60.000 operazioni, si distribuisce in maniera non uniforme, come era facile supporre, tra i vari giorni della settimana con un deciso calo in corrispondeza dei giorni festivi e prefestivi.

Più interessante ai nostri fini è la visualizzazione della distribuzione degli importi, presentata in Figura 5.3. Si può notare come oltre il 75% delle operazioni abbia un importo inferiore o pari a 1000 € mentre si raggiunge circa il 90% del totale portando la soglia a 2000 €. Sono invece meno del 2% le operazioni che superano il valore di 10000 €. Il volume del transato settimanale si aggira invece attorno ai 60 milioni di Euro.

Il grafico in Figura 5.4 riassume la distribuzione del numero di operazioni effettuate per utente. Per rendere l'istogramma più leggibile abbiamo eliminato dal dataset gli utenti con un numero superiore a 100 disposizioni; il dataset così filtrato è comunque rappresentativo del 99% degli utenti. Possiamo vedere come oltre il 65% degli utenti abbia effettuato un numero massimo di 10 operazioni in totale tramite il portale. In

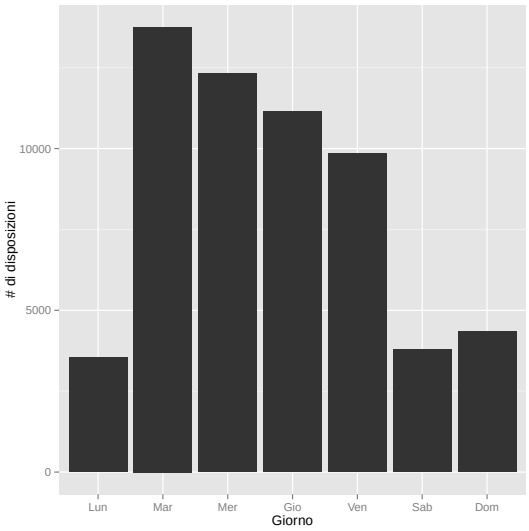

Figura 5.2: Distribuzione giornaliera tipica del numero di operazioni dispositive

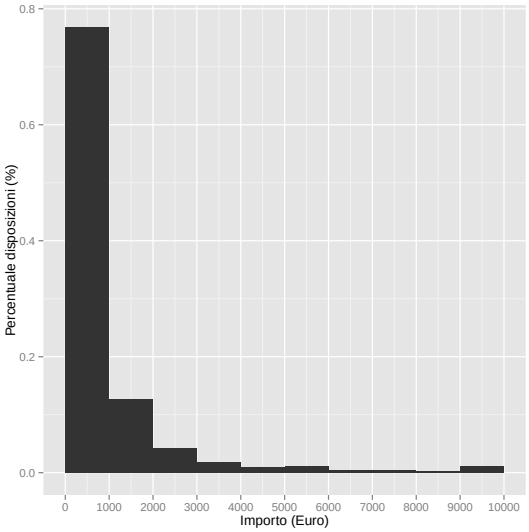

Figura 5.3: Distribuzione delle operazioni per importo

realtà il dato è ulteriormente scomponibile: poco più del 45% degli utenti ha effettuato al più 5 operazioni mentre il 15% addirittura una soltanto. In pratica questo si ripercuote sulle capacità predittive del sistema che non può contare su uno storico oggettivamente significativo per una percentuale molto elevata di utenti.

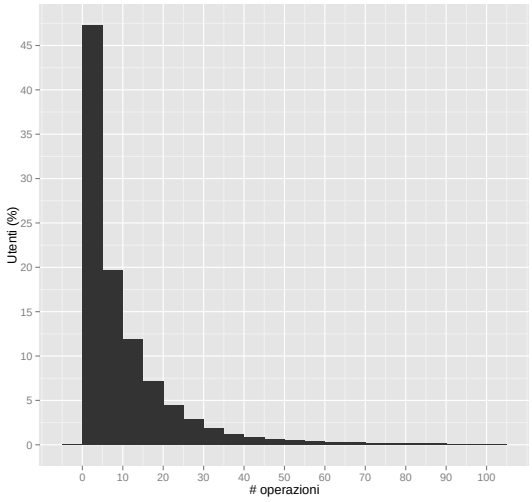

Figura 5.4: Distribuzione del numero di operazioni per utente

Se la percentuale dei trasferimenti operati rimanendo all'interno del territorio nazionale risulta essere del 97% possiamo scomporre il restante 3% nei diversi Paesi interessati. Tra i trasferimenti transfrontalieri il 48% è diretto in Germania, il 26% in Austria e il 5% in Francia. Gli altri Paesi, a scalare, sono la Spagna con il 4%, la Gran Bretagna con il 3%, l'Olanda con poco meno del 2% e così via. Tra questi solo per quanto riguarda la Spagna sono stati segnalati casi di frode in SEC essendo gli altri relativi a trasferimenti operati verso Polonia (0,8% del totale delle operazioni verso estero), Portogallo (0,4%) e Ungheria (0,9%).

6

Modellazione degli utenti

Questo Capitolo descrive formalmente il framework su cui si basa il nostro sistema antifrode. Ampio spazio è dato alla discussione delle varie tecniche utilizzate per modellare le caratteristiche comportamentali degli utenti e per descrivere le strategie di combinazione dell'output dei diversi modelli.

6.1 Il *framework*

Il sistema antifrode da noi sviluppato si inserisce nel filone di ricerca nell'area dell'anomaly detection. Esso si basa su di un *framework* multi-modello la cui formulazione generale si ispira al lavoro di Kruegel et al. [81] benché questo sia stato originalmente elaborato nell'ambito dell'intrusion detection per la protezione di applicazioni web.

Il processo di anomaly detection utilizza una serie di modelli, $\mathfrak{M}_i^{\hat{u}}$, che valutano differenti aspetti del comportamento di uno specifico utente $\hat{u}$. Questi aspetti sono eterogenei e possono essere legati sia al comportamento transazionale sia a quello di navigazione. Si ha

$$\mathfrak{M}^{\hat{u}} = \left\{ \mathfrak{M}_0^{\hat{u}}, \ldots, \mathfrak{M}_{N_m}^{\hat{u}} \right\}$$

dove N_m è il numero di modelli elaborati dal sistema per ogni singolo utente. L'eterogeneità delle caratteristiche monitorate si riflette ovviamente anche sulla tipologia e sul formato dei dati analizzati da ogni singolo modello; ad esempio il calcolo di un modello di spesa richiederà l'elaborazione di record transazionali (i singoli movimenti all'interno di un account) mentre la descrizione comportamentale della navigazione

necessiterà dei dati relativi alle richieste pervenute al web server. Le differenze morfologiche non inficiano però l'architettura generale del *framework*: possiamo visualizzare ogni modello come una *black box* che elabora una singola istanza, un **evento** E^i, e produce un valore di probabilità che rappresenta il livello di concordanza dell'istanza con il modello stesso, quantitativamente espresso come un valore di probabilità. Questo concetto è rappresentato in Figura 6.1. Un evento può essere quindi descritto da un singolo record, una lista di richieste ad un sito web o un'altra tipologia ancora di dato ed ogni modello è progettato per elaborare eventi di una certa classe.

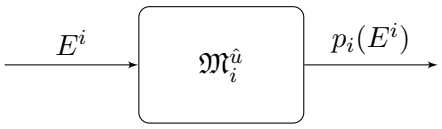

Figura 6.1: Schema a blocchi di un modello

Un modello può operare in due modalità, *training* e *testing*. La fase di *training* è richiesta per permettere al modello di determinare una descrizione compatta, un **profilo**, di una certa caratteristica che vogliamo monitorare allo scopo di individuare scostamenti dalla normalità tali da evidenziare un tentativo di frode. In questa fase il modello viene allenato a partire da un dataset storico di eventi relativi ad un dato utente. Questo può essere costituito dalla lista delle operazioni dispositive effettuate negli ultimi mesi o un estratto dei log del web server corrispodenti ad un diverso intervallo temporale. È importante notare che durante il *training* assumiamo che solo eventi normali vengano elaborati.

Una volta che i vari modelli di un singolo utente avranno "imparato" le caratteristiche degli eventi normali e sarà stato determinato così un profilo complessivo dell'utente il sistema inizierà ad operare in modalità di testing. In questa modalità vengono calcolati i livelli di anomalia e sono riportate le transazioni ritenute sospette.

Il livello di anomalia di una singola transazione è derivato dalla combinazione dei valori di probabilità ritornati dai modelli che sono associati ai diversi eventi. Il valore complessivo è calcolato utilizzando una somma pesata come mostrato in Eq. 6.1. In questa equazione, w_m rappresenta il peso associato al modello m, mentre p_m è il valore di probabilità da esso elaborato. Questa probabilità p_m è sottratta da 1 dato che un

valore prossimo a zero indica la presenza di un evento anomalo che dovrebbe quindi dare luogo ad un elevato livello di anomalia.

$$\text{Anomaly level } = \sum_{m \in \mathfrak{M}^{\hat{u}}} w_m * (1 - p_m) \tag{6.1}$$

Naturalmente, perchè questa equazione abbia un significato, gli eventi analizzati dai singoli modelli devono essere semanticamente collegati. Il nostro sistema ad esempio è in grado di collegare un evento relativo all'inserimento di una nuova transazione all'evento corrispondente alla sequenza di richieste HTTP che ha portato all'esecuzione della stessa operazione.

Il risultato dell'Equazione 6.1 viene utilizzato per determinare le transazioni sospette per mezzo di un valore di soglia th_{abn} configurato a livello di sistema. Tale valore è stato empiricamente calcolato sulla base del carico di lavoro massimo giornaliero a cui possono far fronte gli *auditor* di banca. Le transazioni vengono riportate ad intervalli orari e ordinate per livelli di anomalia decrescenti.

Le sezioni seguenti descrivono gli algoritmi che analizzano le *features* considerate rilevanti per l'individuazione di attività illegittima. Per ogni algoritmo le fasi di *training* e di *testing* vengono discusse.

6.2 Modello del comportamento di navigazione

Quando un utente si interfaccia con un sito Web egli effettua, tramite il suo browser, diverse richieste HTTP al web-server che ospita il sito stesso. La sequenza di queste richieste, dall'ingresso dell'utente nel sito fino alla sua uscita (o logout), forma quindi una traccia di questa attività. Dai file di log del web server è possibile estrarre queste informazioni e utilizzarle per elaborare un modello comportamentale dell'utente.

6.2.1 Il modello DFA

L'idea di utilizzare un automa a stati finiti deterministico per monitorare l'esecuzione di un web server è stata proposta in [75]. In quel caso un DFA (Deterministic Finite Automata) veniva indotto sulla base di un *training set* costituito da un insieme di richieste HTTP comprensive di tutti gli *headers* presenti nell'intestazione delle stesse,

non solo le singole URL. Il DFA forniva una descrizione compatta e formale del linguaggio costituito dalle stringhe corrispondenti alle richieste considerate normali per una specifica applicazione web. Il modello così elaborato veniva poi applicato all'interno di un sistema di *anomaly detection* in grado di identificare richieste non conformi con l'obiettivo di individuare attacchi alle applicazioni ospitate sul web server monitorato. Nonostante le chiare differenze sia di applicazione che di contesto nel seguito descriviamo come è possibile utilizzare una tecnica simile per modellare il comportamento di navigazione dei singoli utenti all'interno di un sito web come un portale di *home banking*.

6.2.2 Definizioni

I concetti di seguito introdotti e definiti saranno alla base della successiva trattazione.

Definizione 1 (Sessione) Una **sessione** è il periodo di tempo che intercorre tra l'accesso e la disconnessione di un utente dal portale di home banking.

Definizione 2 (Richiesta) Sia $R = \{A_1, \ldots, A_M\}$ l'insieme degli attributi di una richiesta, M è il numero di tali attributi. Sia inoltre $U = \{P_1, \ldots, P_N\}$ l'insieme di tutte le possibili URL accessibili da un utente. Una **richiesta** r_t pervenuta al tempo t è allora una tupla:

$$r_t = (P_i, t, u, [a_1, \ldots, a_M]) \tag{6.2}$$

dove P_i rappresenta l'URL acceduta, u è l'utente che ha effettuato la richiesta e a_m è il valore che l'attributo A_m assume per la richiesta.

Esempi di attributi della richiesta possono essere l'indirizzo IP da cui è originata la connessione, la lunghezza dell'intestazione o ancora il codice di stato HTTP ritornato dal server.

Definizione 3 (Sequenza di richieste) Una **sequenza di richieste** $s^{\hat{u}}$ è una collezione ordinata di richieste, pervenute durante una singola sessione dell'utente $\hat{u}$, $\langle r_1, r_2, \ldots, r_N \rangle$ dove N rappresenta il numero di richieste nella sequenza, $r_{i.u} = \hat{u}$ per $1 \leq i \leq N$ e si ha $r_{n.t} < r_{n+1.t}$ ovvero le richieste sono disposte in ordine secondo valori crescenti del tempo di ricezione t. Inoltre se due richieste r_i e r_j, con $r_{j.t} > r_{i.t}$, appartengono alla stessa sequenza allora si ha $r_{j.t} - r_{i.t} < \Delta_h$.

L'ultima clausola della Definizione 3 determina un fattore di raggruppamento temporale delle richieste di una singola sequenza. Lo scopo di questo raggruppamento è quello di partizionare il contenuto di una singola sessione utente in sequenze costituite di richieste temporalmente correlate tra loro. L'idea è che se troppo tempo trascorre tra due richieste allora esse non saranno semanticamente legate. Per ulteriori dettagli in merito a questo criterio e alla scelta del valore del parametro Δ_h si rimanda alla Sezione 7.3.1.4.

Definizione 4 (Database di sequenze) Il **database di sequenze** $\mathbb{D}^{\hat{u}}$ è la collezione delle sequenze di richieste relative ad un particolare utente $\hat{u}$. Si ha perciò $r_{i.u} = \hat{u}$ $\forall r_i \in \mathbb{D}^{\hat{u}}$.

La Tabella 6.1 è un esempio di database di sequenze relativo ad un particolare utente. Per ogni utente del sistema viene memorizzato un tale database.

Sequenza	Pagina richiesta	Timestamp	A_1
s_1	Login	t_1	192.168.1.100
s_1	Saldo	t_2	192.168.1.100
s_1	Pagamento	t_3	192.168.1.100
s_2	Login	t_4	192.168.2.188
s_2	Saldo	t_5	192.168.2.188
s_3	Pagamento	t_6	10.0.0.6
s_3	Logout	t_7	10.0.0.6

Tabella 6.1: Database di sequenze relativo ad un utente

6.2.3 Algoritmo di induzione del DFA

L'algoritmo di induzione dell'automa a stati finiti utilizzato nel nostro sistema è basato sul lavoro di Ingham et al. [75], a sua volta sviluppato a partire dall'algoritmo introdotto da Burge e descritto nello stesso articolo. Si tratta di un algoritmo di complessità $O(nm)$, dove n è il numero di esemplari nel *training set* ed m è il numero medio

di richieste per sequenza. Un'ulteriore importante proprietà dell'algoritmo è che non necessita di esemplari negativi.

Un *training set* di sequenze di richieste viene ricavato dal database di sequenze di un dato utente $\hat{u}$, osservando le richieste pervenute da questo utente in un certo intervallo temporale compreso tra i tempi t_s e t_e, rispettivamente tempo iniziale e finale di valutazione. Ovviamente questi parametri sono variabili; per ulteriori dettagli implementativi si consulti il Capitolo 7. Indicheremo questo specifico database con la notazione $\mathbb{D}^{\hat{u}}_{t_s,t_f}$. Nel caso l'utente $\hat{u}$ sia esplicito dal contesto useremo la forma semplificata $\mathbb{D}_{t_s,t_f}$ ricordando però che in ogni caso un database di sequenze è sempre relativo ad un singolo utente.

Sia $G = (S, A, l)$ un DFA con stati S e transizioni A, $S = \{S_{START}, S_1, S_2, \ldots, S_n, S_{FINISH}\}$. Si ha $A = \{A_{i,j}\}$ dove $A_{i,j}$ è una transizione dallo stato S_i allo stato S_j, con etichetta definita dalla mappa $l(A_{i,j}) \in U$.

Sia $|\mathbb{D}_{t_s,t_f}| = M$ il numero di sequenze di richieste che costituiscono il *training set*. Prima dell'esecuzione dell'algoritmo l'insieme degli stati S del DFA è costituito dal solo stato iniziale S_{START}. L'algoritmo procede poi consumando ordinatamente le sequenze nel *training set* iniziando da quelle meno recenti. Quando una sequenza sequenza s_i viene prelevata dal *training set* da essa vengono estratti, in ordine dalla prima all'ultima richiesta, i percorsi P_{i_j} andando a comporre una stringa W_i in cui ogni percorso rappresenta un singolo simbolo, che chiamiamo *token*, del particolare alfabeto P. A questa stringa viene aggiunto infine un *token* speciale indicato con "END" che indica la terminazione della sequenza di richieste. Si ha

$$W_i = P_{i_1} \| P_{i_2} \| P_{i_3} \| \ldots P_{i_{N_i}} \| \text{“END”} \tag{6.3}$$

dove con $\|$ indichiamo l'operatore di concatenazione tra *token*, $P_{i_j} \in U$ e N_i è il numero di richieste contenute nella sequenza s_i; secondo la definizione di sequenza un singolo percorso P_{i_j} può comparire anche più volte all'interno della stessa stringa W_i. Si noti che $|W_i| = N_i + 1$ per la presenza della richiesta fittizia finale.

Nel nostro modello le pagine visitabili dall'utente diventano quindi i *token* che compongono le varie "stringhe" accettate o meno dall'automa. Queste stringhe a loro volta derivano dalle sequenze di richieste pervenute dall'utente realizzando così un parallelismo tra linguaggio accettato dall'automa e sequenze di richieste normali per

un dato utente nell'utilizzo di una determinata applicazione web, nel nostro caso un portale di home banking.

Una volta costruita la stringa W_i, i cui *token* indichiamo con T_k per $1 \leq k \leq |W_i|$, l'algoritmo procede come seque:

1. Imposta lo stato corrente $C = S_{START}$, $A = \emptyset$
2. Per $k = 1$ a $|W_i|$:
   (a) Se $A_{C,D} \in A$ con $l(A_{C,D}) = T_i$ per un qualche stato $D \in S$ poni $C \leftarrow D$
   (b) Se tale transizione non esiste cerca in S un nodo D tale che è stato destinazione per lo stesso *token* T_i per un qualche nodo sorgente differente da C. Se D esiste (sarà unico in questo caso) allora $A \leftarrow A \cup A_{C,D}$ e $C = D$.
   (c) Se ancora tale nodo D non esiste in S crea un nuovo nodo E e una nuova transizione da C a E. Imposta $C \leftarrow E$.

All'inizio dell'esecuzione dell'algoritmo il DFA è costituito dal solo stato S_{START} che risulta essere lo stato corrente iniziale mentre A, l'insieme delle transizioni, risulta vuoto. Il ciclo principale "consuma" i *token* della sequenza e aggiorna di volta in volta lo stato corrente seguendo la transizione scatenata dal *token* in esame se questa esiste. In caso contrario l'apprendimento prevede due possibilità a seconda che esista o meno un qualche stato D con una transizione in ingresso avente come etichetta proprio T_k: se D esiste viene creata una nuova transizione dallo stato corrente a D stesso altrimenti viene creato un nuovo nodo E e la corrispondente transizione. In Figura 6.2 e 6.3 è illustrato il funzionamento dell'algoritmo.

Tutte le sequenze di richieste nel *training set* $\mathbb{D}_{t_s,t_e}$ vengono processate allo stesso modo reimpostando all'inizio di ogni esecuzione lo stato corrente allo stato S_{START} e aggiungendo nuove transizioni o stati secondo i *token* (le pagine) presenti nelle differenti W_i.

Indicheremo il DFA finale ottenuto dopo aver processato tutte le sequenze di richieste contenuto in $\mathbb{D}_{t_s,t_e}$ con la notazione $G^{\hat{u}}_{t_s,t_e}$. Ancora una volta se l'utente $\hat{u}$ dovesse risultare esplicito dal contesto la notazione semplificata G_{t_s,t_e} sarà utilizzata.

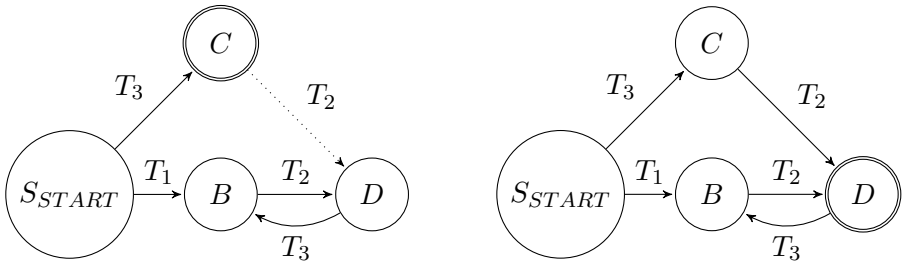

Figura 6.2: Quando il processo di apprendimento consuma i *token* della sequenza a volte devono essere aggiunti una nuova transizione o un nuovo nodo. In (a), C è lo stato corrente, rappresentato dalla doppia circonferenza e il prossimo *token* (T_2) è lo stesso che ha causato la transizione da B a D. L'algoritmo in questo caso produce il nuovo DFA in (b) aggiungendo la transizione da C a D.

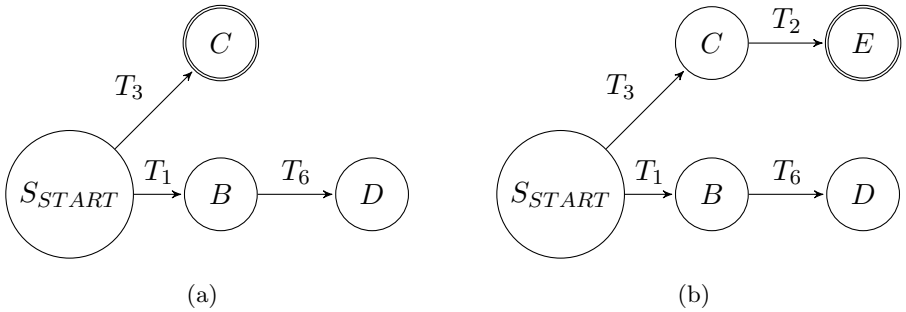

Figura 6.3: In (a), lo stato corrente è C, contrassegnato dalla doppia circonferenza e non esiste nessuna transazioni nell'automa per il *token* T_2. In questo caso quindi il DFA verrà modificato come in (b) con l'aggiunta di un nuovo nodo E e della corrispondente transizione da C ad E.

6.2.4 Testing di una nuova sequenza

Il DFA creato utilizzando la tecnica di induzione rappresenta un vero e proprio modello del comportamento normale di ogni particolare utente del sistema. In presenza di una nuova sessione utente e quindi di una o più nuove sequenze di richieste possiamo utilizzare questo modello per determinare la legittimità di una navigazione e individuare comportamenti anomali. Nel seguito di questa Sezione descriveremo la metodologia adottata per ottenere questo obiettivo all'interno del nostro sistema antifrode.

Un DFA, per sua natura, è semplicemente uno strumento formale in grado di definire l'appartenenza di una stringa di caratteri ad un determinato linguaggio, definito implicitamente appunto dall'automa. Questo meccanismo potrebbe essere utilizzato direttamente anche nell'ambito applicativo proposto ma è lecito attendersi un certo grado di variabilità del "linguaggio" che l'automa inizialmente creato dovrà accettare. La variabilità dipende sia da cambiamenti nel comportamento dell'utente o del software di navigazione utilizzato sia da modifiche al contenuto del sito monitorato.

È necessario perciò definire una misura di distanza per comparare il modello costituito dal DFA con una nuova sequenza di richieste e determinare il livello di anormalità di quest'ultima. Per farlo occorre però prima modificare la classica procedura di *testing* tradizionalmente applicata ad un DFA. Quando un *token* (che ricordiamo rappresenta una singola richiesta effettuata al web server) viene processato e risulta essere illegale rispetto allo stato attuale dell'automa e alle transizioni possibili da tale stato si verifica un evento denominato "token mancante". In questo caso la procedura di *testing* esegue un tentativo di ri-sincronizzazione che consiste nella seguente operazione. Se una transizione corrispondente al prossimo *token* esiste nel DFA ma per un diverso stato sorgente allora la procedura di *testing* effettua una transazione verso lo stato destinazione di tale transizione (vedi Figura 6.4). Nel caso in cui una tale transizione non esista si verifica un secondo evento di "token mancante" e la procedura avanza al *token* successivo nella sequenza tentando una nuovamente una ri-sincronizzazione.

Il numero di "token mancanti" registrati durante l'esecuzione della procedura sopra descritta fornisce una stima del livello di anormalità di una sequenza di richieste rispetto al modello costituito dal DFA. La similarità di una sequenza s rispetto al DFA G viene calcolata dalla formula seguente:

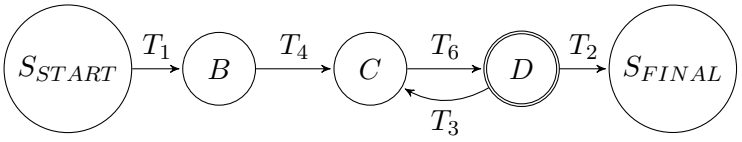

Figura 6.4: D rappresenta lo stato corrente. Se il prossimo *token* nella sequenza risulta essere T_1. Il DFA effettua una transizione verso lo stato B in quanto destinazione dell'arco con etichetta T_1.

$$sim_G(s) = 1 - \frac{\#\text{ di } token \text{ mancanti}}{\#\text{ di } token \text{ nella sequenza}} \in [0, 1]$$

Intuitivamente questa misura riflette il numero di cambiamenti che sarebbe necessarioq introdurre nell'automa a stati finiti (per ogni "token mancante" si dovrebbe introdurre una nuova transizione) perchè questo sia in grado di accettare la sequenza testata come parte del proprio "linguaggio". Si può notare come la misura adottata sia dipendente dal numero di *token*, ovvero di richieste, nella sequenza analizzata. In questo modo sessioni di navigazione più lunghe possono beneficiare di un maggior margine di variazione e rientrare allo stesso tempo all'interno della "soglia di normalità". Al contrario non tener conto della numerosità delle richieste può creare facilmente un elevato numero di falsi positivi nel caso di sessioni particolarmente complesse.

Riferendoci al contesto più astratto del framework antifrode riconosciamo in $sim_G(s)$ il valore di probabilità emesso dal modello in presenza di un nuovo evento s. L'output del modello è quindi quindi

$$p_{DFA}(s) = sim_G(s) \tag{6.4}$$

In Eq 6.4 non viene fatto riferimento esplicito all'utente $\hat{u}$ del quale il DFA G rappresenta il profilo di navigazione. Tale scelta è dettata dalla volontà di non appesantire ulteriormente la notazione. Resta fermo il fatto che un modello si riferisce sempre e soltanto ad un singolo utente del sistema di home banking.

6.2.5 Trattamento dei dati non-stazionari

I siti web cambiano nel tempo; nuove pagine vengono inserite, altre possono essere eliminate o cambiare indirizzo. Allo stesso tempo l'utente stesso modifica il suo modo di interagire con un certo sito, migliorando la propria confidenza con lo strumento,

scoprendo nuove funzionalità o semplicemente imparando nuovi percorsi di navigazione per completare specifiche operazioni. Questo aspetto è noto nella letteratura relativa al campo dell'anomaly detection come *concept drifting*, termine che indica la tendenza di un particolare fenomeno a modificare nel tempo le sue proprietà statistiche, inficiando in questo modo le capacità predittive di un modello che non tenga conto di questa evoluzione; un tale modello vedrà probabilmente diminuire la qualità delle proprie predizioni col passare del tempo. Il fenomeno del *concept drifting* concetto è quindi fortemente legato alla non-stazionarietà dei dati elaborati dal modello. Per limitarne l'effetto il nostro sistema adotta alcuni accorgimenti come di seguito illustrato.

Innanzitutto per seguire il cambiamento nel tempo sia del comportamento dell'utente sia della struttura del portale è stata implementata una funzionalità di auto-aggiornamento del modello. Nel caso in cui venga analizzata una nuova sequenza di richieste che non sia completamente catturata dal DFA corrente ma presenti un elevato grado di similarità con il comportamento dell'utente il sistema provvede all'aggiornamento del modello che verrà quindi modificato per accettare anche la nuova sequenza. La soglia del valore di similarità per questa operazione è ovviamente controllata da un parametro del sistema specificato in fase di configurazione; tendenzialmente è consigliato un valore prossimo a 1 in modo che soltanto sequenze davvero molto simili al DFA vengano introdotte nel modello.

È importamente inoltre monitorare quegli archi nel DFA che corrispondono a sequenze di richieste diventate impossibili (ad esempio nel caso in cui il portale di home banking abbia subito importanti modifiche strutturali) o semplicemente obsolete a fronte di un'evoluzione del comportamento di navigazione di un utente (ad esempio se questi utilizza delle "scorciatoie" o se esplora nuove funzionalità). In [75] questa problematica viene risolta introducendo un contatore per ogni arco che indica il numero di volte che esso è stato attraversato; periodicamente tale conteggio viene monitorato e gli archi con un valore inferiore ad una certa soglia vengono rimossi. Questa soluzione purtroppo risulta complicare notevolmente l'implementazione in quanto la rimozione di un arco può portare ad avere "vicoli ciechi" all'interno del DFA o stati irraggiungibili. Pertanto la soluzione da noi adottata prevede una semplice ricostruzione periodica del DFA.

Con questi due accorgimenti il sistema si adatta velocemente tramite modifiche "addittive" all'introduzione di variazioni nei percorsi di navigazione mentre la rispo-

sta alla diminuzione dell'uso di taluni percorsi risulta condizionata dalla lunghezza dell'intervallo tra una ricostruzione e l'altra del modello.

6.2.6 Resistenza a *mimicry attacks*

I *mimicry attacks* [102] sono una particolare classe di attacchi studiati appositamente per superare le difese costituite dai moderni sistemi IDS basati sulle tecniche di *anomaly detection*. Informalmente possiamo caratterizzare questi attacchi come varianti di attacchi che ottengono lo stesso obiettivo originale ma possono eludere l'individuazione da parte di un IDS. Se nel caso di un IDS *signature-based* realizzare una variante di un attacco è spesso banale ciò non è scontato se il sistema di *intrusion detection* utilizza metodologie appartenenti all'area dell'*anomaly detection*. Tuttavia, come dimostrato in [102], ciò è possibile sebbene richieda un notevole sforzo da parte dell'attaccante. La seguente è una definizione formale di *mimicry attack* adatta agli scopi di questo documento.

Definizione 6 (*Mimicry attack*) Data una sequenza di richieste necessaria ad un attacco, un ***mimicry attack*** $\mathcal{A} \in P^*$ è una sequenza di richieste che può ottenere lo stesso effetto malevolo ma con la proprietà che $p_{DFA}(\mathcal{A}) \geq th_{sim}$.

In realtà la Definizione 6 presenta una semplificazione. Secondo il nostro *framework* antifrode il livello di anomalia di una transizione è costituito da una somma pesata delle probabilità restituite dai vari modelli. Cionondimeno è utile apportare questa semplificazione per comprendere la resistenza a questo tipo di attacchi del modello DFA. Nel caso più generale fornito dal nostro *framework* antifrode l'attività dell'attaccante sarà ulteriormente complicata rispetto a quanto di seguito descritto dal fatto di dover contemporaneamente tenere sotto controllo i valori di anomalia restituiti da tutti i modelli in gioco.

Concretamente, nel nostro contesto applicativo, per attacco intendiamo un qualsiasi tentativo di frode perpetrata attraverso il portale di home banking ai danni di un utente inconsapevole, tipicamente tramite le tecniche MITB descritte nel Capitolo 4.

Un IDS è tanto più resistente a *mimicry attack* tanto più è difficile e costoso per un attaccante individuare sequenze $\mathcal{A}$ opportune. Sfortunatamente l'utilizzo di un modello che tenga conto della non-stazionarietà dei dati introduce una vulnerabilità

che può essere sfruttata per facilitare l'esecuzione con successo di questi attacchi. Il principio è il seguente: se un attaccante può forzare il modello ad "apprendere" la specifiche sequenza $\mathcal{A}$ di modo tale che questa sia ad un certo punto accettata come normale allora potrà successivamente fornire la sequenza in oggetto senza che questa possa essere etichettata come malevola dall'IDS.

Perchè l'attaccante possa essere in grado di raggiungere questo obiettivo egli deve essere a conoscenza dei dettagli del sistema che intende ingannare. Ciò comprende, nel nostro caso, anche il modello DFA dello specifico utente e l'algoritmo di aggiornamento completo dei suoi parametri. Supponendo che questi ultimi due dettagli siano noti un attaccante può costruire un modello utente raccogliendo i dati sulla navigazione dell'utente vittima durante un intervallo di tempo sufficientemente lungo da fornire un modello equivalente (o quasi) a quello memorizzato nel sistema antifrode. Ad esempio il frodatore potrebbe prolungare la sua analisi per un tempo pari all'intervallo di aggiornamento del modello; nel caso di un utente con un'attività di home banking molto intensa il tempo necessario per ottenere una stima sufficiente sarebbe invece molto inferiore.

Se denotiamo con $\mathcal{A}_{orig}$ la sequenza di richieste dell'attacco originale possiamo ottenere uno schema generale per $\mathcal{A}$ che utilizzi richieste "no-ops" intervallate a richieste in $\mathcal{A}_{orig}$. Ad esempio se $\mathcal{A}_{orig} = \langle A_1, A_2, \ldots A_n \rangle$ possiamo costruire la seguente espressione regolare $\mathcal{R}$:

$$\mathcal{R} = \mathcal{N}^* A_1 \mathcal{N}^* A_2 \mathcal{N}^* \ldots \mathcal{N}^* A_n \mathcal{N}^*$$

dove $\mathcal{N} \in P$ rappresenta l'insieme delle richieste che non inficiano il risultato dell'attacco. Tra queste possiamo ad esempio includere le pagine puramente informative ma la maggior parte delle richieste possono essere rese inefficaci semplicemente specificando parametri erronei ricordando che i parametri delle richieste HTTP vengono trascurati nel nostro sistema.

Da queste argomentazioni e ricordando che un'espressione regolare non è altro che una diversa rappresentazione di un automa a stati finiti l'attaccante non deve far altro che determinare il DFA che accetti il linguaggio $\mathcal{R} \cap G$. Come è noto questo è un problema classico della teoria dei linguaggi formali e risulta risolvibile in tempo polinomiale. Ovviamente il linguaggio risultante da questa intersezione può essere vuoto ma questa probabilità diminuisce con l'abilità dell'attaccante di determinare un numero elevato

di pagine "no-ops". Inoltre ricordando che la misura di similarità è dipendente dalla lunghezza della sequenza di richieste si ha che intervallando lunghe sotto-sequenze di pagine "no-ops" a singole richieste dell'attacco originale si è in grado di produrre una sequenza finale con svariate anomalie locali ma che globalmente risulterà accettata dal sistema.

L'attacco, o meglio l'individuazione di una sua corretta sequenza di richieste, come descritto finora risulta completamente passivo. Questo comporta un certo grado di attesa dell'attaccante oltre che di incertezza nella riuscita finale dell'attacco (nel frattempo infatti il trojan occultato nel computer della vittima può essere scoperto da un software antivirus o il disco fisso cancellato). Il frodatore può incrementare le proprie possibilità e velocizzare l'operazione modificando la sua strategia e attivandosi per forzare precisi cambiamenti al modello di navigazione. Rispetto al procedimento descritto sopra questa tecnica richiede all'attaccante di determinare un certo numero $n+1$ di sequenze di richieste $s_1, s_2, \ldots, s_n, s_{n+1} = \mathcal{A}$ tali che

$$\begin{aligned} sim_{G_i}(s_j) < th_{agg} \text{ se } j \leq i, \\ sim_{G_0}(s_1) \geq th_{agg}, \\ sim_{G_1}(s_2) \geq th_{agg}, \\ \ldots, \\ sim_{G_{n-1}}(s_n) \geq th_{agg}, \\ sim_{G_n}(s_{n+1}) = sim_{G_n}(\mathcal{A}) \geq th_{agg} \end{aligned}$$

dove th_{agg} rappresenta la soglia di similarità mimina specificata tramite configurazione di sistema che una nuova sequenza deve superare per scatenare l'aggiornamento del modello, G_0 è il modello iniziale e G_i è il DFA che risulta dall'aggiornamento prodotto in seguito alla sequenza s_i. La prima condizione non è altro che l'ipotesi iniziale del ragionamento ed esprime il concetto che nessuna sequenza può essere accettata dal modello fintanto che le precedenti non vi sono state inserite attraverso l'azione della procedura di aggiornamento.

Sebbene sia possibile individuare una possibile strategia che un attaccante potrebbe adottare nel caso generale ci limitiamo in questa sede a trattare la situazione in cui nel DFA iniziale, G_0, sia presente un lato (o più) per ogni *token* della sequenza di

richieste di attacco $\mathcal{A}$. In questo particolare caso quindi il DFA differisce solo per la mancanza di un certo numero di transizioni che consentirebbero alla sequenza di essere accettata. Supponiamo poi che il grafo del DFA contenga un ciclo di r lati $T_1, T_2, \ldots, T_r = T_1$. È chiaro che nelle condizioni in cui ci siamo posti se il grafo fosse completamente connesso qualsiasi sequenza verrebbe accettata. Per ottenere questo risultato l'attaccante potrebbe sintetizzare una serie di sequenze di questo tipo:

$$\mathcal{S}_{xy} = A(T_1 T_2 \ldots T_{r-1} T_1)^u T_x T_y$$

dove T_x e T_y sono due *token* corrispondenti a due lati non adiacenti nel grafo di G_0 e la sottostringa iniziale A rappresenta il percorso dallo stato iniziale allo stato sorgente del *token* T_1. Questa sequenza verrà accettata fintanto che verrà rispettata la condizione di minima similarità ovvero $\frac{|A|+r^u}{|\mathcal{S}_{xy}|} \geq th_{agg}$. Dato che u può essere arbitrariamente grande è sempre possibile quindi iniettare nel modello una sequenza così prodotta e rendere così adiacenti i *token* T_x e T_y. L'attaccante può utilizzare questa tattica per ogni coppia (T_x, T_y) necessaria per concludere con successo l'attacco sintetizzando un numero polinomiale di sequenze.

Ovviamente il caso generale è più complicato; un ciclo potrebbe non essere presente nel DFA G_0. Anche in questa situazione l'attaccante può però contare sulla dipendenza dalla lunghezza della sequenza del valore di similarità e individuare un percorso nel DFA di lunghezza superiore $l > 2$, costituito ad esempio dai *token* $T_1, T_2, \ldots, T_l$. Le nuove sequenze saranno tutte del tipo:

$$\mathcal{S}'_{xy} = A(T_1 T_2 \ldots T_l)^u T_x T_y$$

Dato che $T_1 T_2 \ldots T_l$ non è un ciclo ma un percorso semplice esso causerà un evento di *token* mancante data la mancanza nel grafo di un collegamento tra T_l e T_1. La formula del caso precedente si modifica quindi in $\frac{|A|+l^u-u}{|\mathcal{S}'_{xy}|} \geq th_{agg}$ che è sempre verificata per valori di u sufficientemente grandi, come nel caso precedente.

Un ulteriore elemento di difficoltà non considerato nella precedente analisi è la necessaria contemporaneità di queste operazioni di manipolazione del modello con l'attività legittima dell'utente. Una soluzione ovvia per l'attaccante sarebbe quella di interrompere tale attività durante l'invio delle sequenze sintetizzate di richieste. L'introduzione

di tempi di attesa imprevisti (l'invio di richieste molto ravvicinate solleva infatti un allarme come descritto in Sezione 7.4.1) potrebbe però insospettire l'utente vittima.

In conclusione possiamo affermare che l'utilizzo di un sistema di aggiornamento adattivo incrementa notevolmente le vulnerabilità del modello DFA ai *mimicry attack*. Al momento non sono noti casi in cui i cybercriminali abbiano sviluppato varianti di trojan specializzate con simili caratteristiche di evasioni ed è probabile, data la natura stessa del sottobosco criminale tecnologico, che c'ho non avverrà fintanto che simili sistemi di protezione non avranno raggiunto una diffusione tale da rendere appetibile un investimento di risorse in questo senso. Ciononostante l'attuale generazione di *banking trojan* implementa già alcune funzionalità di ricerca di informazioni nei computer vittima e il futuro sembra sempre più rivolto verso soluzioni in grado di utilizzare vere e proprie tecniche di *data mining* per estrapolare quanti più dati sensibili, anche non solamente di carattere finanziario, dal sistema compromesso. Per questo è necessario studiare e ricercare nuove tecniche in grado di rendere questi sistemi IDS robusti a questi tentativi di attacco. Alcune tecniche per rendere il nostro sistema più robusto ai *mimicry attack* sono ad esempio l'allungamento dei tempi di risposta rispetto ai nuovi comportamenti di navigazione, il monitoraggio delle richieste web con codice HTTP di errore (per individuare richieste "no-ops"). Una soluzione più drastica richiederebbe l'inibizione della procedura di aggiornamento del modello DFA, a tutto svantaggio però del tasso di falsi positivi.

6.3 Modello di spesa

Il comportamento di navigazione rappresenta certamente un importante aspetto nel quadro del monitoraggio antifrode ma non è sufficiente ad acquisire un'immagine completa del comportamento di un utente. È perciò necessario introdurre nel sistema una componente di analisi delle operazioni dispositive in quanto tali che tenga cioè in considerazione l'aspetto prettamente economico/finanziario del comportamento. Nel seguito di questa Sezione verranno descritte le tecniche utilizzate dal sistema da noi sviluppato per comprendere anche questa caratterizzazione dell'utente.

6.3.1 Definizioni

Prima di trattare nel dettaglio questa componente del sistema sono necessarie alcune definizioni preliminari.

Definizione 5 (Transazione) Una **transazione** τ è una tupla

$$\tau = (a_1, a_2, \ldots, a_L, u, t)$$

composta dai valori di tutti gli L attributi $\{A_1, A_2, \ldots, A_L\}$ selezionati nel database dei dati transazionali bancari. Due ulteriori attributi, u e t, rappresentano rispettivamente l'utente che ha effettuato la transazione e il *timestamp* dell'operazione.

Tutte le transazioni formano un insieme di transazioni T. Per semplificare l'analisi seguente indichiamo con $T^{\hat{u}}$ l'insieme delle transazioni dell'utente $\hat{u}$ e con $T^{\hat{u}}_{t_s,t_f}$ lo stesso insieme considerando però solamente le transazioni tali per cui $t_s \leq \tau.t \leq t_f$, ovvero comprese temporalmente tra l'istante iniziale t_s e quello finale t_e.

6.3.2 Elaborazione del modello

Il nostro sistema antifrode utilizza un modello prettamente statistico per delineare il caratterizzare il comportamento dispositivo di ogni utente. Il modello è necessariamente locale, intendendo con ciò il fatto che esso è specifico e personalizzato per ogni singolo utente. Indichiamo con $M^{\hat{u}}_{\hat{t}}$ quindi il modello dispositivo relativo all'utente $\hat{u}$ calcolato al tempo $\hat{t}$. Esso è una tupla

$$M^{\hat{u}}_{\hat{t}} = (\mu^{\hat{u}}, \sigma^{\hat{u}})$$

dove $\mu^{\hat{u}}$ e $\sigma^{\hat{u}}$ sono rispettivamente la media e la varianza dall'attributo *importo* calcolate su tutte le transazioni $\tau \in T^{\hat{u}}_{t_s,t_f}$ dove t_s rappresenta l'istante corrispondente alla data di attivazione del servizio di home banking per l'utente $\hat{u}$ e $t_f < \hat{t}$ è un istante di tempo definito come $t_f = \hat{t} - a * s_{day}$. Nell'ultima formula s_{day} è il numero di secondi in un giorno e a è a tutti gli effetti un parametro del sistema ($a = 1$ nella nostra configurazione). Si può considerare questo parametro a come una stima dei giorni di attività fraudolenta (o di attività fraudolenta mista ad attività legittima) precedenti alla scoperta di tale attività per un determinato account. Nella nostra esperienza i

cybercriminali hanno agito puntualmente in tutti i casi reali analizzati, vale a dire che l'attività fraudolenta su di un certo account si è svolta ed è terminata con un'unica transazione illegittima. Questo non significa però che le metodologie di attacco non possano in futuro cambiare, anche in virtù di tecniche di fraud detection sempre più efficaci; in tal caso si potrebbe prevedere un valore di a più conservativo (maggiore) di modo da non considerare nel modello transazioni che siano temporalmente troppo vicine a quella in esame.

6.3.3 Testing di una transazione

Il modello sopra descritto viene utilizzato dal nostro sistema per produrre un valore di rischio legato all'importo di una transazione. Tale valore di rischio viene calcolato facendo uso della ben nota disuguaglianza di Chebyshev (Equazione 6.5).

$$\Pr(|x - \mu| > k) < \frac{\sigma^2}{k^2} \tag{6.5}$$

La disuguaglianza esprime un limite superiore alla probabilità che un valore della distribuzione si discosti di almeno un valore k dalla media per una distribuzione arbitraria di media μ e varianza σ^2. In maniera informale 6.5 afferma che in una distribuzione di probabilità quasi tutti i valori tendono a trovarsi "vicino" al valor medio.

Si tratta di una disuguaglianza valida per qualsiasi distribuzione (sconosciuta a priori eccetto che per i valori di media e varianza); benchè questa proprietà sia un ovvio punto di forza essa comporta anche generalmente la capacità di determinare dei *bound* più deboli se comparati con quelli ottenibili nel caso in cui siano note ulteriori informazioni relativamente alla distribuzione analizzata.

Sostituendo in 6.5 il valore di soglia k con la distanza tra l'importo s e la media stimata (ovvero $|s - \mu|$) otteniamo l'Equazione 6.6.

$$\Pr(|x - \mu| > |s - \mu|) < \frac{\sigma^2}{(s - \mu)^2} \tag{6.6}$$

In questo modo siamo in grado di individuare un limite superiore alla probabilità che un importo possa distare maggiormente dalla media della distribuzione rispetto all'istanza corrente. Nel nostro sistema consideriamo come potenzialmente illegittime solamente le transazioni tali per cui $s \geq \mu$, ove cioè l'importo è superiore o al più uguale alla media stimata. L'Equazione 6.7 illustra il calcolo del valore di probabilità $p_{imp}(s)$

prodotto dal modello di spesa (indicato col diminutivo *imp*) durante la fase di test di una nuova transazione τ di importo s all'interno della formalizzazione del *framework* antifrode. Anche in questo caso non si è voluto appesantire la notazione; si tenga però a mente il fatto che i parametri statistici sono relativi all'utente che ha effettuato la transazione τ.

$$p_{imp}(s) = \begin{cases} 1 & \text{se } s \leq \mu, \\ \frac{\sigma^2}{(s-\mu)^2} & \text{altrimenti} \end{cases} \tag{6.7}$$

La disuguaglianza di Chebyshev ci fornisce una metrica efficiente in grado di modellare le probabilità decrescenti per gli importi che superano considerevolmente il valore medio senza per questo ricorrere a soluzioni che prevedono l'utilizzo di schemi ad intervalli fissi (molto spesso applicati nei sistemi antifrode tradizionali). Inoltre l'approccio con Chebyshev ci permette di tener conto anche della variabilità dei dati e ci dà il vantaggio di valori di probabilità che variano gradualmente (in maniera opposta a schemi decisionali "si/no"). D'altro canto valori di limite molto deboli come quelli spesso offerti da questa disuaguaglianza risultano in un grado di tollerenza piuttosto elevato, una proprietà che in talune applicazioni può non essere desiderata. Nel nostro caso questo ci permette di contenere il numero di falsi positivi etichettando come sospetti solamente quei casi evidenti di *outliers*.

7

Implementazione

In questo capito descriviamo in maniera dettagliata l'implementazione del sistema, soffermandoci nella descrizione concreta delle sue componenti fondamentali.

7.1 Contesto

Questo progetto è nato da una collaborazione con una azienda locale con un'esperienza di oltre 40 anni specializzata nella fornitura di servizi informatici in outsourcing, in particolare focalizzata su realtà finanziarie e bancarie sia di livello regionale che nazionale. In particolare lo sviluppo del sistema antifrode, nella sua fase protipale, ha coinvolto 12 degli istituti bancari clienti. Tutti questi clienti usufruiscono di un servizio di online banking incluso in un'offerta più ampia che copre l'intera filiera di processi di una banca. Si tratta di un portale web funzionalmente e graficamente personalizzabile a livello del singolo istituto ma che si basa su una struttura comune, sia per il *front-end* che per il *back-end*.

Dal punto di vista della sicurezza l'accesso al portale prevede l'inserimento del proprio username, di un PIN e di un codice OTP ottenuto tramite un token SecurID. All'atto dell'inserimento di una nuova transazione all'utente viene richiesta la digitazione di un ulteriore codice OTP. Questo codice autentica si l'utente ma non garantisce l'integrità della transazione i cui dati possono essere dunque manipolati da un attacco MITB classico (cfr. Capitolo 4).

7.1.1 Struttura del Sistema Informativo

L'architettura del sistema informativo è molto complessa e non verrà qui riportata; allo scopo del nostro progetto è preferibile adottare una visione da un livello di astrazione superiore che semplifichi la struttura pur mantenendo un quadro sufficientemente realistico. Il risultato di questa analisi è riportato in Figura 7.1.

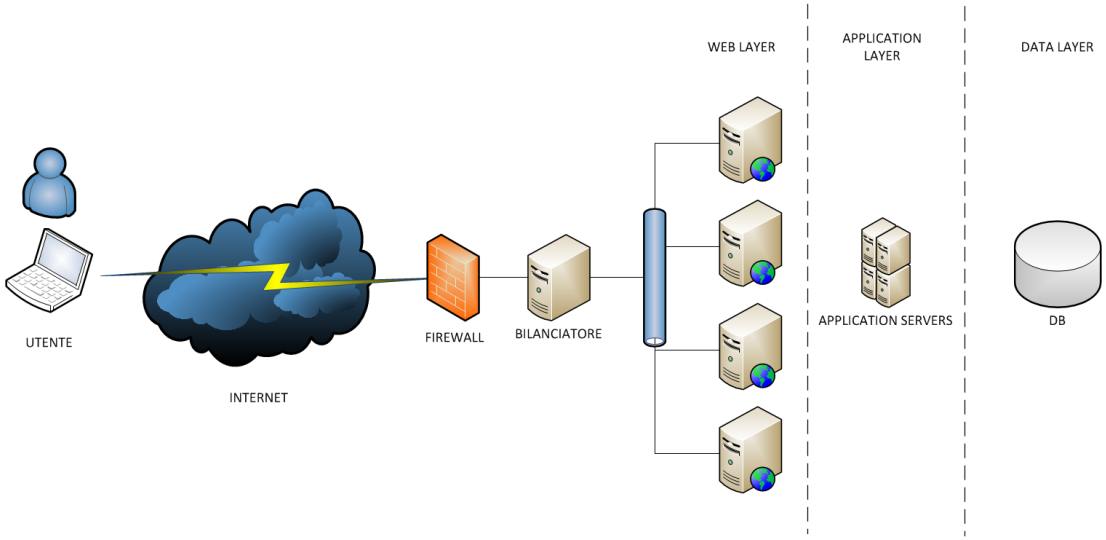

Figura 7.1: Architettura del sistema informativo

Attraverso una connessione Internet sicura (protocollo HTTPS) l'utente si collega ai servizi Web bancari utilizzando un comune browser. L'*endpoint* di connessione è qui rappresentato da un firewall, comprendente sia un dispositivo per l'ispezione del traffico dei livelli inferiori nella pila ISO/OSI che un firewall applicativo WAF. La richiesta dell'utente viene analizzata, verificando la presenza di pattern riconducibili ad attacchi classici. Se nulla di sufficientemente pericoloso viene rilevato la richiesta è inoltrata al livello successivo, un nodo bilanciatore che smista le connessioni in entrata verso 4 differenti HTTP server a seconda del carico delle singole macchine. I server HTTP gestiscono la comunicazione con il livello più propriamente di *business logic*, quello applicativo, attraverso il quale vengono effettivamente eseguite le varie operazioni richieste dall'utente. Il livello applicativo si appoggia ad un ulteriore livello, indicato come *data layer*, costituito da un insieme eterogeneo di sorgenti di dati, tra cui diversi DBMS relazionali (Oracle, DB2, ...). In Figura 7.1 non viene indicato il livello di autenticazione, localizzato subito dopo il bilanciatore e utilizzato per l'autenticazione

utente in tutte le richieste HTTP. Questo livello non è strettamente connesso alla nostra analisi ma ha un ruolo importante nell'implementazione del sistema antifrode in quanto etichetta ogni richiesta HTTP con il codice identificativo dell'utente che l'ha effettuata. I riflessi pratici di questa caratteristica saranno illustrati in Sezione 7.3.1.

7.1.2 Sorgenti dati

Ognuno dei livelli esposti in Sezione 7.1.1 fornisce un qualche meccanismo di *logging*, necessario sia per il *debugging* che per l'analisi in caso di *incident*. Nel seguito presentiamo una descrizione di queste fonti di dati riassumendone le proprietà e il contenuto.

Web Log

Ognuno dei 4 server HTTP in Figura 7.1 produce un log che registra tutte le richieste ad esso pervenute. Globalmente questa funzione di *logging* mantiene traccia dell'interazione di ogni utente con il sistema informativo al suo livello più basso, fornendo un gran numero di dettagli. Come risultato i dati memorizzati presentano una dimensione giornaliera non indifferente che, sommando il contributo dei 4 server, raggiunge i 4GB (circa 10 milioni di record), con oscillazioni anche corpose a seconda del giorno della settimana (ad esempio il traffico è minore traffico se il giorno è festivo). I record sono memorizzati in un semplice file di testo utilizzando un formato standard, denominato *Common Log Format* (CLF)[1] che garantisce perciò una perfetta normalizzazione ed un'alta qualità del dato. In particolare vengono specificate l'origine della richiesta a la risorsa richiamata ma anche altre importanti informazioni. Per un'analisi più completa si faccia riferimento alla Sezione 5.1.1. Un processo schedulato provvede a comprimere i dati storici per ridurre lo spazio necessario alla loro conservazione attraverso il noto algoritmo DEFLATE[2] che, in questo caso, fornisce un rapporto di compressione del 10% circa. Data l'ubiquità di questa metodologia di archiviazione riteniamo lungimirante la scelta di perseguire lo sviluppo di un sistema antifrode basato sui web log benchè l'architettura di memorizzazione distribuita e il formato testuale richiedano un pesante lavoro di preprocessing e riordinamento dei record. La predisposizione di

[1] per una descrizione completa si consulti la documentazione del Web Server Apache all'indirizzo `http://httpd.apache.org/docs/2.2/mod/mod_log_config.html`

[2] implementato dall'utility *gzip*

un sotto-sistema centralizzato di logging, unita ad una indicizzazione automatizzata, consentirebbe di alleviare questa problematica.

Tracking Log

Questo log, anche detto *audit log*, rappresenta una visione ad un livello superiore dell'interazione dell'utente con il sistema di online banking, proprietà che lo rende utile strumento di analisi soprattutto in considerazione della sua facilità di lettura. Rispetto al web log il livello di dettaglio è nettamente inferiore e l'utilizzo del sistema viene descritto come una sequenza temporale delle chiamate alle varie *routine* applicative, la cui memorizzazione è affidata ad una specifica tabella di un database. Nessun tipo di argomento o parametro viene registrato, solamente un'etichetta testuale che descrive univocamente e verbalmente la procedura richiamata accompagnata da alcune informazioni identificative (codice utente) e temporali. Data la natura di questi dati, spesso impiegati nel *debugging*, non tutte le chiamate vengono registrate: questo riduce sì la dimensionalità ma aumenta di conseguenza la granularità del dato riducendo l'affidabilità delle informazioni contenute, specie se si considera l'obiettivo di monitoraggio che ci prefiggiamo di raggiungere.

Date le problematiche riscontrate si è perciò preferito non approfondire l'analisi del contributo informativo offerto da questa tipologia di log. Nonostante ciò crediamo che una standardizzazione della procedura di logging applicativo possa essere una strada da percorrere in quanto ha le potenzialità di semplificare le operazioni di raccolta dei dati, interamente localizzati in una singola base di dati e già normalizzati. Nel seguire questa linea sarà però necessario tener conto dell'esplosione della quantità di record che potrebbe impattare le prestazioni della componente business del sistema se non corroborata da un'appropriata architettura per la memorizzazione.

Chiaramente le informazioni contenute nel tracking log dipendono dalla singola applicazione e sono perciò destinate a variare a seconda del contesto aziendale. Un sistema di monitoraggio progettato esclusivamente sull'analisi di questi dati potrà essere difficilmente generalizzabile e quindi trasferibile in altre realtà.

Database

L'ultima fonte di dati esaminata è costituita dai record memorizzati nelle varie basi di dati utilizzate dal sistema. L'architettura del sistema informativo utilizza sia un

sistema distribuito di accesso ai dati basato sui prodotti Oracle sia un *mainframe* IBM, dotato del DBMS DB2. Data la criticità dei dati memorizzati in questi database l'accesso è fortemente restrittivo e controllato a livello amministrativo. Secondo le *policy* aziendali e visti gli obiettivi del nostro progetto ci è stato fornito un account con privilegi di sola lettura limitato ad alcune specifiche tabelle necessarie per l'ottenimento delle informazioni transazionali e per permettere il collegamento di queste transazioni con gli utenti del sistema di online banking. Inoltre, data la tipologia di servizio fornito, la priorità di esecuzione delle istruzioni SQL è stata fortemente penalizzata per garantire, *in primis*, la continuità e la qualità delle funzionalità di *business logic*.

I record analizzabili, una volta filtrati, forniscono lo stato e la descrizione delle operazioni dispositive effettuate dagli utenti. I campi utilizzati e il formato dei dati sono descritti in maniera più dettagliata in Sezione 5.1.2. Ai fini del monitoraggio anti-frode saranno analizzate solamente le transazioni corrispondenti a bonifici bancari disposti attraverso il canale Internet. Si noti che questa fonte di dati non è propriamente un sistema di logging: si tratta in realtà di informazioni utilizzate direttamente dal livello applicativo per processare le richieste degli utenti. I dati in esso contenuti sono perciò completi (ovviamente solo per quanto concerne le transazioni) e affidabili e quindi di alta qualità. A titolo informativo riportiamo un numero di record variabile tra i 3000 e i 15000 giornalieri, una volta eseguiti gli opportuni filtri e tenendo in considerazione solamente gli istituti coinvolti nel progetto anti-frode.

7.2 Architettura

In questa sezione descriviamo nel dettaglio l'architettura del sistema di monitoraggio anti-frode da noi implementato e la sua integrazione con il sistema informativo preesistente.

Una rappresentazione schematica ad alto livello dell'architettura del sistema anti-frode è osservabile in Figura 7.2. Nella parte laterale sinistra dello schema si trovano le componenti del sistema informativo bancario, costituito dai web server e da un database relazionale che memorizza le transazioni operate dai vari utenti del portale di home banking. Una serie di blocchi di preprocessing assicurano il filtraggio, la normalizzazione e la memorizzazione efficiente sia dei dati di navigazione che quelli transazionali.

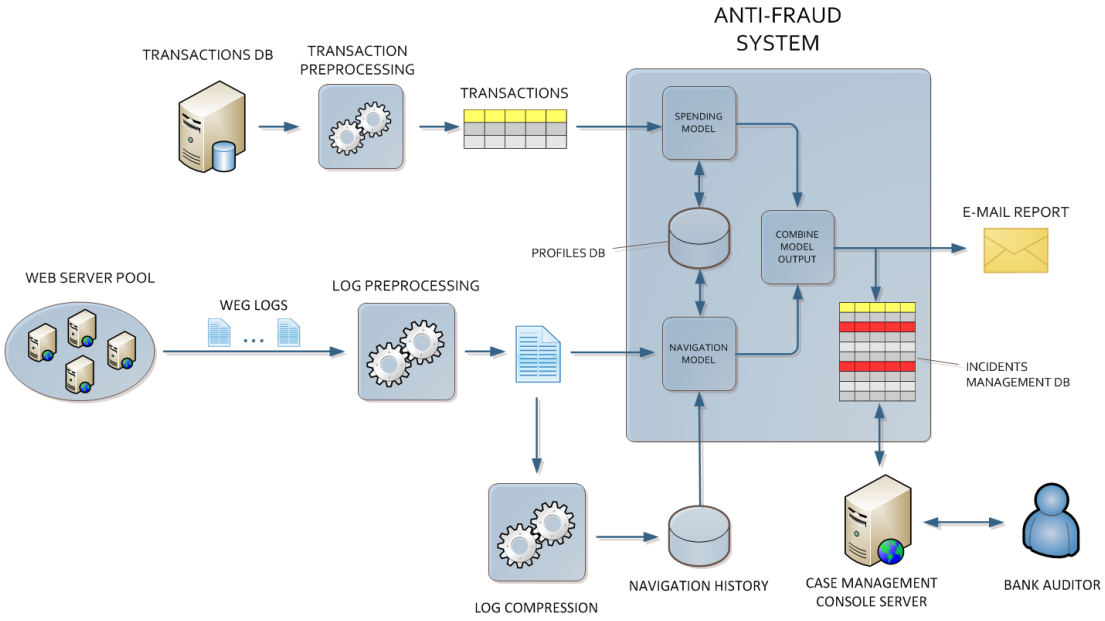

Figura 7.2: Architettura ad alto livello del sistema di monitoraggio anti-frode

Nelle Sezioni successive descriveremo concretamente nel dettaglio le procedure attuate all'interno di questi blocchi.

Il *core* del sistema è costituito da due blocchi di valutazione, rispettivamente dedicati all'analisi della sessione utente e del valore dell'importo di ogni transazione. Un database interno al sistema memorizza i profili descrittivi degli utenti utilizzati per queste analisi comparative mentre un archivio storico delle navigazioni viene interrogato nel caso un modello di navigazione non sia ancora stato costruito per l'utente (o non sia sufficientemente recente). Il sistema combina l'output di queste valutazione con una serie di euristiche, atte a migliorarne le capacità di individuazione e ridurre i falsi positivi, producendo un report da inviare agli auditor di banca. I casi sospetti, in gergo *incidents*, vengono memorizzati in un particolare database che può essere interrogato e manipolato attraverso un'apposita console web, utile per l'investigazione ad opera del personale di banca.

7.3 Preprocessing dei dati

Il sistema da noi implementato utilizza due principali fonti di dati per valutare il rischio di una determinata transazione: i file di log dei web server e il log delle transazioni

memorizzato all'interno di specifiche tabelle di un database relazionale. Entrambe queste sorgenti di dati subiscono una necessaria fase di *preprocessing* per consentirne la trattabilità e ridurre il rumore.

7.3.1 Preprocessing dei dati di navigazione

Il primo ostacolo nella determinazione di un modello di navigazione utente è certamente quello di individuare con esattezza le richieste ad esso appartenenti e di divedere tali richieste in sessioni di navigazione. Nelle applicazioni più comuni un sistema che effettui questo tipo di attività dispone soltanto delle informazioni direttamente ricavabili dai file di log del web server e, a differenza della nostra implementazione, non può servirsi del dato relativo all'account (il codice identificativo) di colui che ha effettuato la richiesta. Questa lacuna complica notevolmente il problema rendendo più difficile asserire con un sufficiente livello di affidabilità se una certa richiesta "appartiene" ad un utente o da un altro. Inoltre, anche basandosi sugli indirizzi IP per ricostruire il percorso di navigazione di un certo utente ci si scontra con un' ulteriore problematica rappresentata da due differenti meccanismi di *caching*, la cache locale del browser dell'utente e l'eventuale presenza di *caching proxy servers*, che possono ridurre notevolmente la tracciabilità di una navigazione a partire dai file di log. Nel seguito di questa discussione analizzeremo le tecniche e le strategie adottate per risolvere o mitigare questi problemi.

7.3.1.1 Identificazione delle richieste

In [48] gli autori delineano alcune procedure euristiche per l'identificazione degli utenti attraverso l'analisi dei web logs. Malgrado fosse possibile implementare le soluzioni proposte in tale studio nel nostro caso ciò non è stato necessario. Nei file di log prodotti dai web server è infatti già presente per la maggior parte delle richieste, un campo identificativo avvalorato da un sotto-sistema di autenticazione. Questa etichetta viene trasmessa al web server tramite un l'inserimento di un nuovo header HTTP e consente una notevole semplificazione dell'operazione di tracciatura degli utenti, anche qualora le loro richieste provengano dallo stesso indirizzo IP di origine. Le poche richieste che non sono direttamente identificabili dal campo sono per lo più richieste a file statici quali immagini o fogli di stile; richieste di questo tipo non influiscono sulla semantica della navigazione e sono per lo più indirettamente inviate dal browser utente in seguito

all'elaborazione di specifici tag HTML. Il nostro sistema utilizza una *whitelist* configurabile di estensioni per dividere l'output dei file di log tra richieste interessanti (che hanno cioè un valore semantico) e quelle indirette che vengono così automaticamente rimosse.

Una volta eliminata questa grossa porzione di richieste non identificate quelle rimanenti sono essenzialmente riconducibili alle pagine di *login* e *logout*, pagine per le quali l'informazione relativamente all'utente connesso non è ancora disponibile o è già stata rimossa in seguito alla disconnessione. In questi casi si è convenuto di applicare un'euristica che sarà descritta in Sezione 7.3.1.3.

7.3.1.2 Effetti dei meccanismi di *caching*

Data la natura delle connessioni criptate con protocollo SSL, utilizzato nell'implementazione delle comunicazioni Internet sicure HTTPS, i browser moderni hanno esteso in maniera predefinita l'adozione di meccanismi di *caching* anche agli oggetti richiesti tramite queste connessioni. Il vantaggio di conservare delle copie locali non criptate di tali oggetti consiste soprattutto nell'evitare un numero eccessivo di operazioni di negoziazione della connessione sicura che possono introdurre rallentamenti percebili. È chiaro che le pagine inviate attraverso trasmissioni crittografate possono contenere informazioni sensibili tali da rendere la loro conservazione un possibile rischio per la privacy; è perciò responsabilità degli sviluppatori delle singole applicazioni Web prevenire la memorizzazione di tutti gli oggetti considerati confidenziali.

In particolare la cache è spesso utilizzata per conservare gli oggetti statici (immagini, fogli di stile, ...) che non subiscono modifiche frequenti nel tempo. Questo tipo di oggetti è completamente ignorato nel nostro sistema in quanto trattasi di contenuti richiesti indirettamente e privi di un valore semantico. Pertanto qualsiasi meccanismo di *caching* applicato soltanto a risorse di questo tipo risulta ininfluente ai nostri fini. Lo stesso si può dire riguardo alle cosiddette richieste condizionate; inserendo specifici *header* nell'intestazione della richiesta HTTP (*If-None-Match* e/o *If-Modified-Since*) il browser dell'utente specifica al server di inviare il contenuto soltanto se una nuo-

va versione dello stesso è presente. In questi casi il browser effettua comunque una connessione di cui il server manterrà traccia nei log di sistema[1].

Nel caso dell'applicazione di home banking da noi monitorata gli sviluppatori hanno adottato infine la combinazione di due ulteriori tecniche per limitare il più possibile la memorizzazione locale dei contenuti nei vari livelli di caching. La prima metodologia consiste nell'applicazione delle specifiche esposte nel documento RFC 2616. Tale documento descrive alcuni meccanismi, interni al protocollo HTTP, per il controllo del livello di cachine che prevedono l'utilizzo di specifici *header* nell'intestazione della risposta[2]. Una seconda tecnica anch'essa molto diffusa nell'ambito dello sviluppo di applicazioni web richiede l'inserimento tra i parametri di ogni collegamento ipertestuale di un valore casuale, ininfluente a livello applicativo, per rendere irripetibile ogni URL e invalidare così l'eventuale contenuto nella cache.

Non deve preoccupare nemmeno l'eventuale azione di "disturbo" dovuta ai *proxy server*. Questo tipo di dispositivi, tipicamente impiegati in contesti organizzativi, non sono impostati in maniera predefinita per effettuare il *caching* di richieste HTTPS, sia per una questione meramente implementativa sia per il danno alla privacy che ne deriverebbe per gli utenti. Infatti non è possibile conservare copia dei contenuti senza procedere ad una decrittazione degli stessi. Esistono tuttavia contesti aziendali che, per ragioni di sicurezza, monitorano anche il traffico HTTPS generato all'interno della propria rete facendo uso di particolari proxy che operano come dei veri e propri Man-in-the-Middle SSL, usufruendo allo scopo di un certificato digitale di fiducia all'interno dell'azienda stessa. Questo livello di sofisticazione viene per lo più utilizzato per operazioni di monitoraggio del traffico e non comporta necessariamente la presenza di alcun meccanismo di caching HTTPS.

7.3.1.3 Divisione in sessioni

Quando tutte le richieste sono state identificate e quindi l'insieme totale è stato partizionato nei diversi utenti l'attività successiva consiste nel distinguere, nelle lunghe

[1] a titolo informativo riportiamo che lo stato della risposta corrisponderà a **304 - Not Modified** in caso il contenuto sia corrispondente a quello memorizzato nella cache mentre **200 - OK** in caso contrario

[2] si veda l'*header* **Cache-Control** descritto nel documento RFC 2616, `http://www.ietf.org/rfc/rfc2616.txt`

sequenze di richieste relative ad un singolo utente, tra blocchi semanticamente legati. Quello che viene fatto tipicamente in questi casi è utilizzare un semplice timeout: 30 minuti sembra un valore predefinito per la maggior parte dei prodotti commerciali mentre in [42] è stata stabilita empiricamente la soglia di 25.5 minuti. Si tratta però di valori che non necessariamente si adattano bene a tutte le situazioni e va eventualmente effettuata un'analisi sito per sito per determinare il valore migliore. Nel nostro sistema questo parametro, che indicheremo con *timeout*, è impostato in maniera predefinita a 20 minuti.

7.3.1.4 Trasformazione dei dati

Abbiamo già menzionato il filtro delle richieste nelle Sezioni precendenti; questa Sezione fornirà invece lo spazio per un approfondimento di questo aspetto e per descrivere le fasi successive della trasformazione dei dati che, in ultima istanza, produce le sessioni utilizzate per la realizzazione dei nostri modelli di navigazione. Innanzitutto le operazioni di filtro vengono eseguite da uno script PERL che processa riga per riga i log accumulati avvalendosi di un'espressione regolare per identificare i campi di interesse discussi precedentemente in Sezione 5.1.1.2. In questa fase vengono rimossi tutti i record riferiti a richieste effettuate da particolari agenti software, denominati anche "sonde", appositamente sviluppati per monitorare il funzionamento dell'applicazione di home banking e che non sono perciò di interesse ai nostri fini. L'identificazione delle navigazioni prodotte da questi software è semplice in quanto il "Codice utente" utilizzato è memorizzato in un'opportuna lista; inoltre sono noti gli indirizzi IP sorgenti di queste comunicazioni.

L'URL di ogni richiesta viene successivamente decomposto nelle sue parti e analizzato procedendo all'eliminazione di tutti i record associati ad oggetti statici attraverso l'ispezione dell'estensione del file richiesto; le componenti non di interesse dell'indirizzo vengono rimosse (querystring, protocollo, porta, dominio, ...), così come precedentemente indicato in Sezione 5.1.1.2. I record vengono quindi memorizzati in un nuovo file adottando il formato indicato in Tabella 7.1.

Si può notare come il *timestamp* temporale sia stato trasformato in un formato totalmente numerico[1] a differenza del formato originale utilizzato comunemente nei

[1] Il formato è denominato *Unix time* e rappresenta il numero di secondi trascorsi dalla mezzanotte (UTC) del 1° Gennaio 1970

Indice	Nome Campo	Esempio Valore
1	IP	95.244.185.115
2	Timestamp (Unix Time)	1338559744
3	URL	/HomeBanking/ListaMovimenti.do
4	Utente	usr12345678

Tabella 7.1: Formato adottato per la memorizzazione temporanea dei dati dopo la prima fase di *pre-processing*

log di tipo CLF che comprende un'indicazione testuale del mese e alcuni caratteri speciali; questa scelta è stata fatta in virtù di una maggiore trattabilità del dato che permette di semplificare (velocizzare) le operazioni di ordinamento temporale delle richieste nella successive fase di trasformazione. Questa infatti prevede l'ordinamento dapprima dei singoli file prodotti dal processo sopra descritto, relativi ai 4 diversi server web, seguito dalla fusione (*merging*) dei record in un singolo file ordinato. Si è inoltre optato per un ordinamento stabile per evitare così l'introduzione di modifiche artificiose nella sequenza delle richieste nel caso di contemporaneità. In Figura 7.3 si può osservare una schematizzazione di questa prima parte di *preprocessing*.

Il file risultante contiene la sequenza giornaliera delle richieste, corredate da un identificatore utente ricavato direttamente dai log. Sfortunatamente, come affermato precedentemente in Sezione 7.3.1.1, non tutte le entrate dei file di log forniscono un codice identificativo, situazione che rende necessario ricorrere ad un'euristica per individuare gli utenti a cui appartengono le rimanenti. Esaminando i log queste vengono assegnate all'utente avente lo stesso indirizzo IP che ha contattato il web server entro un tempo Δ_h dalla richiesta in esame. Questa euristica non è avulsa da errori, basti pensare ad una situazione particolarmente affollata in cui più utenti si colleghino ed effettuino richieste al server web nello stesso periodo di tempo, ad esempio in un contesto aziendale dove è probabile che le connessioni in uscita vengano inoltrate attraverso un unico *endpoint* e siano caratterizzate dunque dallo stesso indirizzo IP sorgente. Riteniamo però una tale situazione poco probabile data la natura del sito monitorato. Un modo per irrobustire questa euristica, qualora lo si ritenesse necessario, potrebbe essere quello di verificare non solo l'indirizzo IP ma anche la stringa *User-Agent* delle

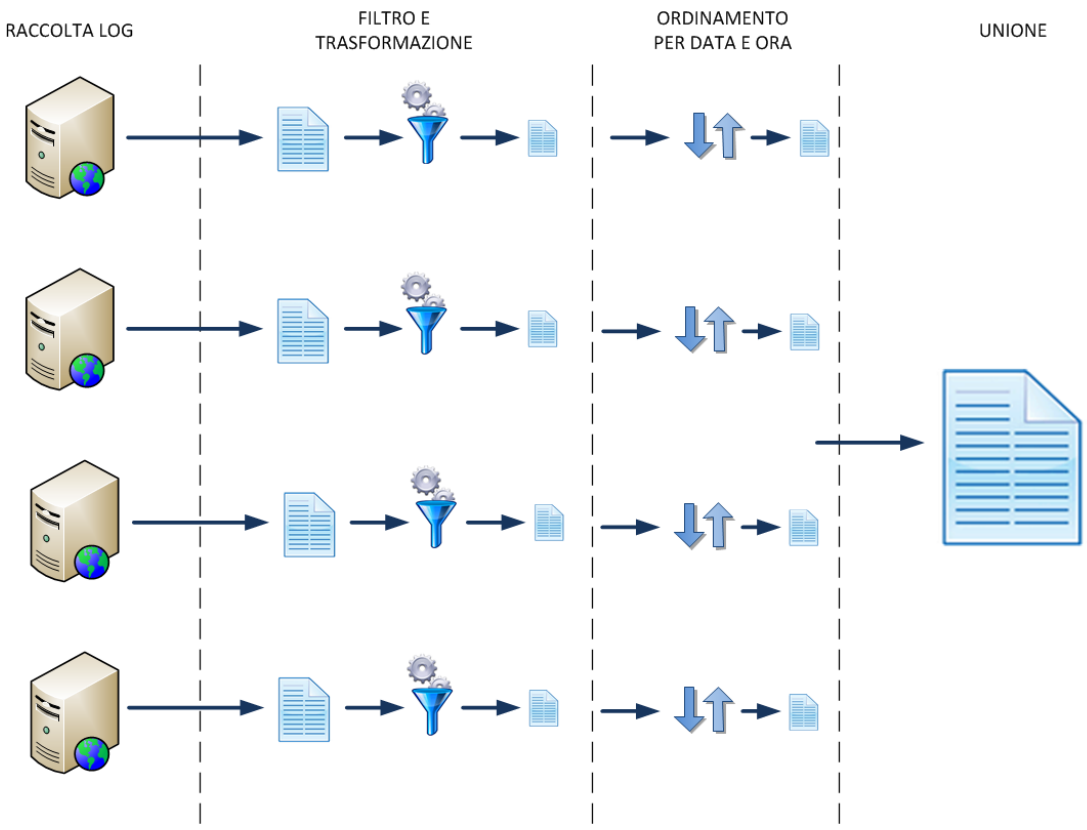

Figura 7.3: Rappresentazione delle varie fasi di *preprocessing*: (1) in prima istanza i log vengono trasferiti dai singoli web-server e (2) separatamente filtrati e processati; (3) i record così dimensionalmente ridotti vengono ordinati per data e ora; (4) infine, sfruttando questo parziale ordinamento, si procede all'unione complessiva

richieste temporalmente vicine a quella esaminata così da avere un ulteriore elemento di confronto a scapito però di un aumento dei tempi di *preprocessing* e della complessità del sistema.

Una volta che tutte le richieste sono state etichettate opportunatamente il sistema procede ad un nuovo ordinamento, stavolta però utilizzando il codice utente come chiave. Sfruttando le proprietà di un ordinamento stabile si ottiene così un unico file finale ordinato per codice utente e, in seconda battuta, per data e ora della richiesta.

Confrontando i dati di navigazione trasformati attraverso questo processo si nota una diminuzione consistente delle dimensioni che vengono ridotte a circa un decimo rispetto a quelle iniziali (cfr. Sezione 7.1.2); applicando l'algoritmo DEFLATE si ottengono file giornalieri di dimensioni non superiori ai 30MB con il livello di traffico attuale memorizzati sotto forma di archivi *gzip*.

Il formato *gzip*, è stato scelto per la sua enorme popolarità e per l'ubiqua disponibilità di strumenti per la sua manipolazione. Si tratta del tipo di compressione più frequentemente utilizzata per la memorizzazione dei *log* dei più famosi web server in ambito UNIX/Linux a cui non fanno eccezione quelli predisposti nel contesto aziendale in cui si pone questo progetto. Sperimentando però con la creazione *on-the-fly* dei modelli di navigazione dei singoli utenti direttamente da questi file compressi è emersa fin da subito una problematica piuttosto chiara: la necessità di un accesso seriale ai dati codificati di fatto riduce sostanzialmente il benificio prestazionale portato dall'ordinamento dei record in base al codice utente.

Infatti, benchè sia possibile decomprimere anche solo parzialmente un file *gzip* risulta comunque necessario decomprimere l'intera porzione del file precedente una tale locazione impendendo di fatto l'utilizzo della strategia di ricerca binaria per velocizzare le operazioni di recupero dei dati necessari alla creazione di un modello o da visionare a scopo investigativo.

Questa limitazione del formato *gzip* ci ha orientato alla ricerca di tecniche di memorizzazione più sofisticate ma che permettessero al contempo di mantenere la flessibilità, la portabilità e la semplicità del metodo inizialmente adottato: in questo senso la soluzione proposta da *dictzip* sembra il miglior compromesso. Si tratta di uno strumento per la compressione e decompressione, compatibile con *gzip*, sviluppato all'interno del progetto DICT[1] del DICT Development Group, un progetto per lo sviluppo di un pro-

[1] *The DICT Development Group*, `http://www.dict.org/bin/Dict`

tocollo client/server (il protocollo DICT[1], appunto) per l'accesso tramite TCP a diverse voci di dizionario.

Nel contesto del progetto DICT *dictzip* è stato sviluppato come metodo efficiente per la memorizzazione delle voci garantendo la possibilità di un accesso pseudo-casuale. Per raggiungere questo obiettivo il programma utilizza particolari campi di intestazione del formato *gzip*[2]; tali campi, essenziali per poter posizionare il cursore di lettura in maniera pseudo-casuale all'interno del file, vengono completamente ignorati dal software *gzip* originale mantenendo in questo modo la compatibilità e la portabilità del formato.

Una descrizione dettagliata del formato *dictzip* è riportata in Appendice A. Alcune informazioni sono però necessarie per la comprensione della sua applicazione in questo progetto. Fondamentalmente il file da comprimere viene diviso in *chunks* la cui struttura viene memorizzata sfruttando un opportuno campo di intestazione specificato dal formato *gzip* e originariamente inteso proprio per lo sviluppo di estensioni del formato stesso. In fase di lettura è possibile attuare una strategia di ricerca binaria determinando il primo *chunk* nella sequenza che contiene i dati di interesse e proseguendo decomprimendo tutti i successivi *chunks* corrispondenti. In questo modo si evita la necessità di decomprimere inutilmente grosse porzioni di file con un miglioramento netto in termini di velocità di lettura riducendo solo minimamente l'efficienza del processo di compressione: l'*overhead* dovuto alla definizione delle struttura risulta infatti marginale per file di dimensioni elevate mentre le dimensioni dei file compressi con *dictzip* risultano essere superiori di circa il 3-4% (vedi Tabella 7.2) rispetto a *gzip*, aumento certamente tollerabile.

File	Dim. originale (byte)	Dim. *gzip* (byte)	Dim. *dictzip* (byte)	Diff. %
File 1	101.014.793	8.252.409	8.627.277	+4,5
File 2	350.591.392	27.733.983	28.679.316	+3,4
File 3	299.907.244	23.821.633	24.626.812	+3,4
File 4	257.500.559	19.885.007	20.538.739	+3,3

Tabella 7.2: Differenti valori di dimensione per la compressione *gzip* e *dictzip* per alcuni file

[1] Il protocollo è descritto nel documento RFC2229, `http://tools.ietf.org/html/rfc2229`
[2] Il formato *gzip* è descritto nel documento RFC1952, `http://tools.ietf.org/html/rfc1952`

Il grafico in Figura 7.4 mostra le differenze prestazionali tra le diverse soluzioni provate[1]. I tempi riportati sono riferiti alla ricerca dei record relativi a 50 utenti casuali (su oltre 60.000 presenti) all'interno di un file contenente i dati di navigazione completi di una giornata elaborati secondo le operazioni di preprocessing sopra descritte.

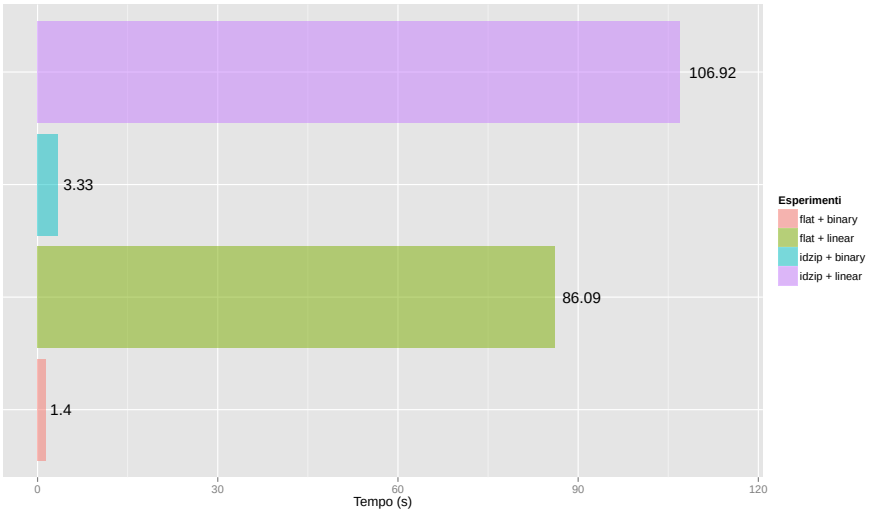

Figura 7.4: Differenze prestazionali percentuali tra i vari metodi di ricerca sperimentati

Sono stati effettuati quattro diversi esperimenti, due per ogni tipo di algoritmo utilizzato (ricerca binaria o lineare) e per ogni tipo di formato del file (compresso con *dictzip* o puramente testuale). Si può notare come la ricerca binaria nel file compresso nel formato *dictzip* risulti solo marginalmente più lenta rispetto alla ricerca su file testuale puro. Molto più lente, come era facile supporre, sono entrambe le soluzioni basate sulla ricerca lineare, risultato che premia la scelta di applicare un ordinamento preventivo dei record. Le buone prestazioni ottenute dalla ricerca all'interno dei file compressi unite al vantaggio della conservazione molto più efficiente in termini dimensionali degli stessi forniscono un argomento sufficiente per l'utilizzo di questa tecnica di memorizzazione.

È necessario precisare che l'implementazione comunemente disponibile di *dictzip* presenta un limite di 4GB per la grandezza dei file da comprimere. Per superare questa limitazione si è pertanto risolto di utilizzare una differente implementazione,

[1] i test sono stati eseguiti su di un computer portatile equipaggiato con CPU Intel Core 2 Duo @ 2.80 GHz e 4GB di RAM

denominata *idzip*[1] che utilizza ulteriori caratteristiche descritte nel formato *gzip* per estendere in maniera virtualmente infinita le dimensioni massime. Ancora una volta rimandiamo il lettore interessato all'Appendice A per informazioni più dettagliate.

7.3.2 Preprocessing dei dati transazionali

I dati transazionali sono memorizzati in una specifica tabella all'interno di un RDBMS utilizzato per le operazioni *business* del sistema informativo. Tale database è gestito in modo da assegnare una quantità limitata di risorse computazionali ad ogni singola utenza connessa in modo da privilegiare l'esecuzione delle applicazioni *mission critical*. Questa restrizione ha posto degli importanti vincoli nella progettazione del nostro sistema che verranno di seguito discussi.

Uno degli obiettivi iniziali di questo progetto richieda al sistema di monitoraggio una modalità di esecuzione offline per alleggerirne l'impatto sul sistema informativo aziendale. Per raggiungere questo obiettivo un componente del sistema antifrode esegue, ad intervalli regolari di un'ora, una query al database di produzione contenente le transazioni inserite dagli utenti. L'utilizzo di un account limitato per l'accesso a tale database riduce il tempo massimo riservato per l'elaborazione della query rendendo di fatto necessario applicare varie tecniche di ottimizzazione per migliorare le prestazioni dell'interrogazione. Una di queste tecniche consiste nell'utilizzare a vantaggio del sistema due indici precostruiti nel database che consentono una ricerca molto veloce delle transazioni in base all'istituto bancario di appartenenza dell'utente e, come indice secondario, in base all'orario. Grazie a questo si è potuta implementare una strategia un po' complicata ma efficacie:

1. viene selezionato il codice identificativo di uno degli istituti; denotiamolo con cod_ist

2. si controlla il timestamp dell'ultima transazione analizzata per tale istituto; denotiamolo con last_timestamp

3. tramite una query al database di produzione si recuperano N transazioni inserite da utenti della banca cod_ist a partire da last_timestamp

[1] *idzip*, ovvero *improved dictzip*, è un'utility scritta in linguaggio Python, `https://code.google.com/p/idzip/`

4. le transazioni vengono memorizzate in un database specifico del sistema antifrode dopo un'operazione di pulizia, filtro e normalizzazione

5. si aggiorna last_timestamp al valore dell'ultima transazioni recuperata

6. si eseguono i punti 3-5 fintanto che l'interrogazione non ritorna più nuove transazioni

7. viene memorizzato il valore di last_timestamp per la successiva esecuzione della procedura

8. si ripetono i punti 1-7 per il successivo cod_ist

Tutte queste operazioni non sono state implementate in codice ma si è preferito utilizzare uno strumento di *Extract Transform Load* (ETL), precisamente il software Pentaho Data Integration[1] (PDI). La scelta è ricaduta su questo tipo di strumento per l'immediato supporto a tecnologie quali JDBC per il collegamento a diverse fonti di dati e per la flessibilità che esso permette. Infatti tramite l'ambiente di sviluppo grafico è possibile inserire o togliere componenti di elaborazione e modificare il flusso delle informazioni con estrema facilità. Meno immediata invece è stata l'implementazione della logica vera e propria dell'algoritmo sopra descritto, in quanto PDI non consente una semplice gestione dei cicli. Una volta completato il lavoro di sviluppo la trasformazione create viene convertita in un file XML che viene eseguito successivamente da uno dei componente di PDI.

La Figura 7.5 mostra l'ambiente di sviluppo fornito da PDI. Al centro dell'area di lavoro è possibile vedere la struttura a blocchi di una delle componenti dell'algoritmo sopra delineato.

7.4 Il modello all'opera

In questa Sezione descriveremo come viene effettivamente implementato l'utilizzo del modello costituito dall'automa a stati finiti generato per ogni singolo utente per valutare la legittimità di un'operazione dispositiva. Verranno inoltre illustrati ulteriori accorgimenti ed euristiche che aiutano a diminuire il carico di lavoro del sistema e il

[1] `http://kettle.pentaho.com/`

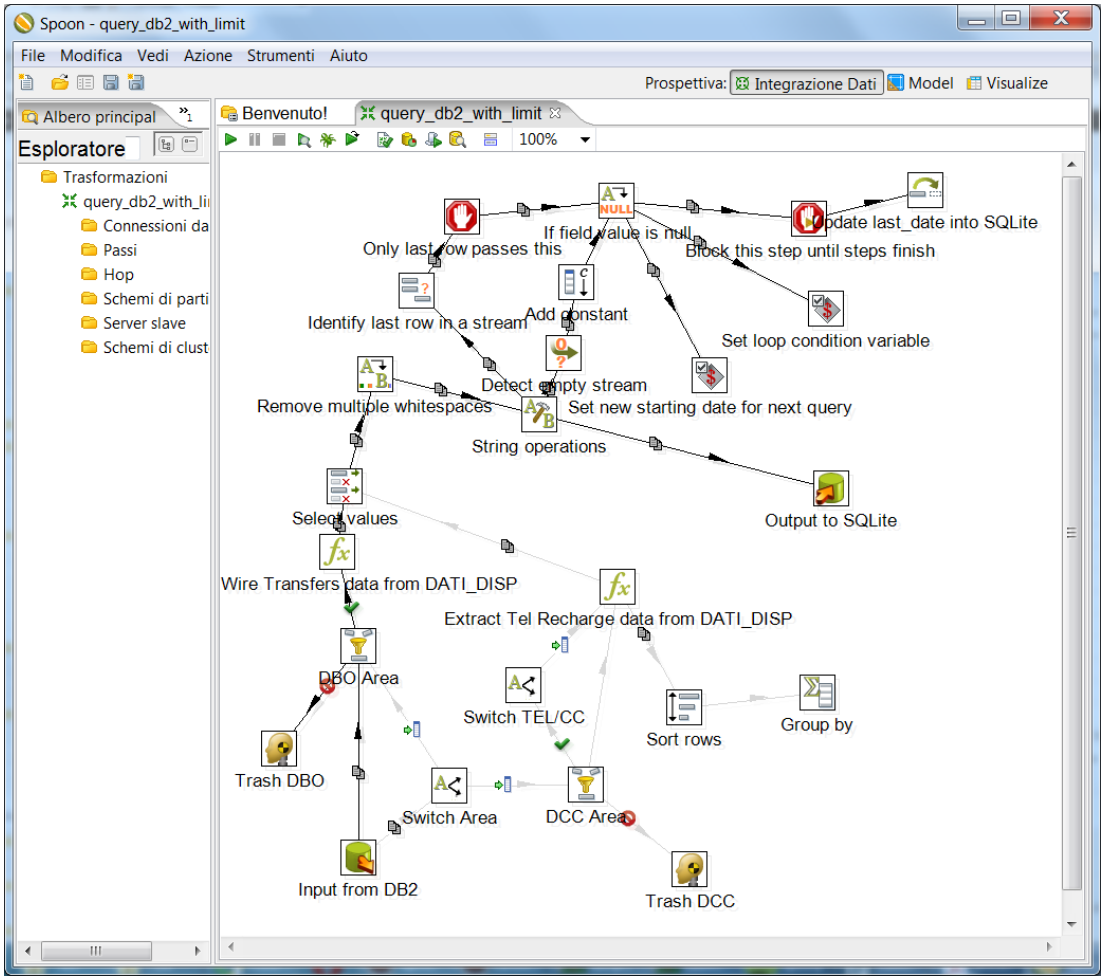

Figura 7.5: L'area di lavoro di Pentaho Data Integration; le trasformazioni possono diventare anche piuttosto complesse

tasso di falsi positivi permettendo agli *auditor* di concentrare la propria attenzione sui casi considerati più rischiosi oltreché dannosi.

7.4.1 Euristiche

Facendo leva sulla conoscenza di alcune particolarità del dominio è stato possibile sviluppare alcune semplici euristiche per migliorare sia l'efficienza che l'efficacia del sistema antifrode. Queste euristiche sono in grado di aumentare o diminuire il livello di rischio di un'operazione dispositiva prodotto come output dal sistema favorendo così l'analisi dei casi considerati più pericolosi. In questo modo si cerca di minimizzare lo spreco di risorse e di focalizzare maggiormente l'attività degli *auditor* di banca.

Blacklist Analizzando i dati in nostro possesso relativi a frodi avvenute nel recente periodo a danno degli istituti monitorati all'interno del progetto antifrode sono stati ricavati circa 20 IBAN precedentemente utilizzati in casi reali di frode; uno di questi è stato utilizzato in addirittura 2 diverse occasioni. Come abbiamo discusso nel Capitolo 3 questi account sono riconducibili ai *money mules* che di volta in volta si sono prestati, più o meno inconsapevolmente all'attività criminale. Benché quindi non sia possibile affermare con certezza assoluta che un trasferimento di denaro verso uno di questi IBAN sia necessariamente un atto fraudolento è necessario dare quanto meno un forte peso qualora questo accadesse innalzando il livello di rischio dell'operazione.

Il nostro sistema gestisce un database relazione che mantiene la *blacklist* contenente account con la possibilità per gli *auditor* di inserire ulteriori IBAN nel caso di nuovi episodi fraudolenti. L'efficacia di questa tecnica è più alta in realtà proprio nel periodo immediatamente successivo ad un nuovo caso di frode in quanto non ci aspettiamo, a distanza di molti giorni o addirittura mesi, l'utilizzo da parte dei frodatori di uno stesso account già compromesso; sarebbe infatti una scelta poco lungimirante per i criminali che vedrebbero diminuire drasticamente le possibilità di riuscita dei propri intenti criminosi.

Account noto Il nostro sistema non analizza i pagamenti verso conti bancari per cui sono già stati registrati trasferimenti da parte dello stesso utente; consideriamo infatti tali account degni di fiducia. Per rendere quest'euristica più robusta e resistente a tentativi di manipolazione da parte di un attaccante si dovrebbero considerare solamente i

bonifici più vecchi di almeno qualche giorno, onde evitare che un cybercriminale possa condurre una singola operazione, di piccolo volume, per elevare la fiducia di un account mulo poco prima di un trasferimento più ingente.

A titolo informativo in Tabella 7.3 riportiamo la percentuale di operazioni verso nuovi account disposte dagli utenti in due periodi di lunghezza rispettivamente pari a 1 e 4 settimane. L'insieme degli IBAN noti è stato invece calcolato tenendo conto di tutte le disposizioni effettuate dai singoli utenti nell'anno precedente. Come si vede dalla tabella, ma era facile prevederlo, la percentuale di operazioni verso account inediti per un dato utente aumenta con l'aumentare dell'intervallo temporale considerato. Il dato mostra chiaramente come almeno il 65% delle nuove operazioni possa essere facilmente trascurato sulla base della storia personale di un cliente (60% se consideriamo le sole operazioni sopra i 2000€ che rappresentano il 12% del totale).

Periodo	% op. verso nuovi account	% op. verso nuovi account ($\geq$ 2.000 €)
1 settimana	31,95	36,28
4 settimane	36,05	40,04

Tabella 7.3: Percentuale di operazioni verso nuovi account nell'arco di differenti periodi

Limite inferiore della somma Idealmente gli *auditor* dovrebbero analizzare ogni caso riportato dal sistema antifrode. In realtà ciò non è sempre possibile per mancanza di risorse. Il sistema antifrode consente quindi di fissare una soglia sotto la quale inibire l'analisi di una data operazione e relativa sessione di navigazione. In questo modo viene sensibilmente snellito il carico di lavoro del sistema al costo di sorvolare però su alcuni possibili casi di frode. L'esperienza dei casi reali di cui disponiamo ci supporta in questa scelta evidenziando come i frodatori preferiscano trasferire somme ingenti di denaro per lo più per ragioni di opportunità ma anche economiche (i casi di successo sono pochi, è necessario massimizzare l' "investimento"). Al momento il sistema prevede due differenti soglie, una per i pagamenti verso conti Italiani e una per quelli verso l'estero. Se necessario ulteriori stratificazioni possono essere facilmente inserite nel codice del sistema una volta completata la fase di prototipizzazione.

In Tabella 7.4 riportiamo alcuni dati relativi alla distribuzione delle operazioni al variare dell'importo. I valori sono stati calcolati considerando un periodo di un

mese. Sulla base di questi numeri una strategia possibile per alleviare il carico di lavoro del sistema e degli operatori addetti all'investigazione potrebbe prevedere il monitoraggio delle sole operazioni al di sopra dei 2.000 €. Una soglia più specifica si potrebbe calcolare in funzione di una precisa struttura di costo definita a livello aziendale che tenga conto delle ore-uomo necessarie per l'investigazione, del tasso medio di falsi positivi riportati dal sistema oltre che delle perdite per la banca dovute al numero di falsi negativi, ovvero i casi di frode trascurati in seguito all'applicazione della soglia.

Importo minimo (€)	% op. mensili
500	43,50
1.000	25,15
1.500	16,07
2.000	12,09
2.500	9,19
3.000	7,74

Tabella 7.4: Distribuzione delle operazioni in funzione dell'importo minimo

Velocità della navigazione Un semplice criterio per individuare l'azione di un *robot* o di un trojan è quello di esaminare la sequenza temporale delle richieste ed individuare intervalli di tempo troppo ridotti tra due pagine susseguenti nella sequenza o tra due particolari pagine di riferimento. Un software sofisticato potrebbe facilmente emulare il comportamento umano evitando di inviare richieste HTTP ad una velocità improbabile per un utente reale; cionondimeno la conoscenza di questa eventualità risulta utile agli *auditor* durante la fase di indagine.

Abbiamo registrato casi di falsi positivi il cui livello di rischio era stato innalzato proprio a causa di questa euristica. Questo fatto ha reso evidente la necessità di migliorarne la valutazione inserendo un confronto non più in termini assoluti sui tempi di navigazione ma piuttosto cercando di modellare questa caratteristica relativamente al singolo utente. Ciò risulta necessario in quanto il comportamento degli utenti è vario; mentre ci sono utenti abituati ad usare il portale di home banking per i propri pagamenti (ad esempio imprenditori o amministratori) la cui operatività all'interno del

sito è decisamente più rapida sono presenti anche nuovi utenti o comunque utenti con minor esperienza o confidenza con lo strumento per i quali è opportuna una differente taratura dell'euristica. Tale caratteristica dovrà essere pianificata nella versione finale del sistema antifrode.

Account affidabili Non tutti gli account destinazione di trasferimenti di denaro hanno lo stesso livello di rischio. Nella fase di analisi di una nuova operazione dispositiva è possibile far leva sull'affidabilità dell'IBAN verso cui il denaro viene trasferito per ridurre il numero di falsi positivi. Il nostro sistema antifrode calcola periodicamente questo grado di affidabilità contando il numero di singoli differenti account per i quali si è in passato registrato un trasferimento verso un dato IBAN. Questo IBAN sarà considerato affidabile se collegato ad almeno min_users[1] altri account. Il sistema prevede poi diverse soglie basate sul valore di affidabilità per la diminuizione del livello di rischio di un'operazione.

Questa euristica, benché molto utile ed effacacie, va utilizzata però con prudenza. Non è opportuno infatti impostare un valore troppo basso per min_users. In tal caso si aumenterebbero le possibilità di successo per un frodatore che volesse sfruttare l'euristica stessa per far passare inosservata un'operazione manipolando il calcolo dell'affidabilità nel periodo immediatamente precedente costringendo diversi *money mules* ad effettuare bonifici verso il conto destinazione finale della frode; in mancanza di complici il cybercriminale potrebbe utilizzare un *banking trojan* per raggiungere lo stesso scopo realizzando diverse piccole transizioni illegittime ma con poco visibilità. È però improbabile che un simile livello di organizzazione possa essere messo in campo per un singolo episodio criminale.

La Tabella 7.5 espone il numero di IBAN confidenti individuati in un dataset comprensivo di oltre 2.600.000 disposizioni, operate attraverso il sistema di home banking nell'arco di un anno da 220.000 utenti, in funzione di diversi gradi di affidabilità minima. Una terza e quarta colonna evidenziano la percentuale di nuove operazioni che verrebbero considerate a basso rischio in base al grado di affidabilità specificato e all'importo. Queste operazioni appartengono ad un testing set che racchiude circa 240.000 operazioni eseguite nel mese immediatamente seguente al termine del periodo considerato per la determinazione degli account confidenti.

[1] attualmente nel sistema prototipo $min\_users = 7$

Grado di conf.	# Account conf.	% op. affidabili	% op. affidabili ($\geq$ 2000 €)
5	20.269	26,12	11,10
6	16.479	24,77	10,28
7	13.722	23,54	9,48
8	11.723	22,52	8,89

Tabella 7.5: Risultati della sperimentazione con diversi gradi di affidabilità

Come si può notare benchè il numero di account confidenti individuati a livello di affidabilità minimo pari a 5 sia il doppio rispetto a quanto calcolato per il valore 8 la percentuale di operazioni considerate non a rischio si discosta di soli 4 punti rimanendo superiore al 22%. Se invece consideriamo solamente le nuove operazioni con importo superiore o uguale a 2.000 € questa percentuale si abbatte portandosi, al livello di affidabilità massimo, addirittura al di sotto del 9%. Ulteriori riduzioni sono previste all'aumentare della soglia minima di importo.

Trasferimenti di denaro all'estero Seppur basata su un ristretto numero di casi di frode reali questa euristica pone un'attenzione superiore su tutti quei trasferimenti di denaro verso conti ospitati presso banche straniere. In effetti dei 20 casi analizzati soltanto 3 utilizzavano *money mules* basati in Italia. Questa situazione è spiegabile data la maggior facilità per le autorità locali nel recuperare il denaro piuttosto che in un contesto internazionale; i cybercriminali devono quindi orientarsi verso conti residenti all'estero rendendo praticabile questa euristica. Data la natura poi dello scenario del cybercrimine globale è più probabile che l'arruolamento di *money mules* avvenga in Nazioni con maggiori difficoltà economiche dove la percentuale di persone disposte a questo tipo di incarichi è potenzialmente più elevata. Oltre a questo dato però i frodatori devono tener conto di un altro fattore ovvero il tempo necessario per l'accredito di una somma successivamente ad un bonifico. Nell'area SEPA questi tempi si stanno progessivamente assottigliando arrivando anche a sole 24 ore. Tempi minori implicano una minor possibilità di veder cancellato un trasferimento illegittimo ed è per questo che tutti casi di frode pervenuti indicano pagamenti all'interno dell'Unione Europea, in particolare verso Spagna, Portogallo, Ungheria e Polonia. È perciò opportuno analiz-

zare con più attenzione i trasferimenti di un utente verso questi Stati, in special modo qualora non vi siano dati storici che confermino precedenti scambi internazionali.

Primo trasferimento di denaro Nel caso di un nuovo utente al primo utilizzo con il portale di home banking il sistema, non avendo sufficienti dati per inferire alcuna ipotesi valida, assume un atteggiamento conservativo e segnala ogni caso (tenendo sempre conto della soglia minima di importo descritta più sopra). La situazione descritta è tecnicamente denominata *undertraining* denotando il fatto che il modello non risulta in possesso di abbastanza dati per produrre un output affidabile. L'atteggiamento conservativo del sistema è tanto più valido nel caso in cui la prima operazione sia proprio verso uno Stato estero tra quelli considerati più a rischio. Nonostante questo è però chiaro che in presenza di un elevato numero di nuovi utenti giornalieri o comunque di utenti che effettuano il loro primo trasferimento si renderà necessario adottare un approccio meno prono ai falsi positivi. Una possibilità potrebbe essere quella di costruire un modello di navigazione *ad hoc* utilizzando per la sua realizzazione i dati provenienti dalle sequenze di navigazione dei soli utenti con un profilo di registrazione simile (utenze domestica piuttosto che utenza *business*) oppure con un simile profilo di utilizzo del servizio, ad esempio considerando soltanto le prime sessioni di navigazione di un certo numero di utenti per produrre un modello ridotto. Abbiamo sperimentato in particolare con quest'ultima possibilità, la cui validità è già stata studiata in [94] evidenziando buone prospettive seppur in un contesto differente (*anomaly detection* per IDS HTTP). Sebbene i risultati siano stati incoraggianti ulteriore lavoro è necessario per stabilire i criteri più opportuni di raggruppamento per la realizzazione di uno o più modelli globali, aggiornati periodicamente, da sfruttare in caso di *undertraining* del modello locale (del singolo utente).

Geolocalizzazione Avendo a disposizione l'indirizzo IP di ogni richiesta pervenuta il sistema è in grado di utilizzare tale data per eseguire un'analisi a livello geografico delle varie richieste. Al di là dei *velocity checking* e *collision checking*, già visti in numerosi sistemi antifrode specializzati per il settore delle telecomunicazioni, che abbiamo implementato per irrobustire il nostro sistema è possibile utilizzare questa informazione per individuare un IP di connessione anomalo rispetto a quelli utilizzati comunemente da un dato utente. In questo caso per anomalo intendiamo un indirizzo associato ad una differente area geografica (meglio ancora una differente Nazione) in quanto spesso

l'indirizzo può variare anche all'interno di una stessa sessione se ad esempio il *modem* dell'utente è stato riavviato o una perdita di tensione ha richiesto il riavvio della macchina.

Integrazione output WAF In Sezione 5.1.1.3 abbiamo descritto come il sistema antifrode possa essere integrato con sistemi già in uso per la protezione delle applicazioni web. Nel contesto reale in cui è stato sviluppato questo prototipo l'azienda utilizza un WAF per individuare gli attacchi più comuni. Il prodotto in questione è stato da noi configurato in modo da monitorare il caso in cui un utente, all'interno di una stessa sessione di navigazione, risulti utilizzare due o più browser differenti osservando il campo User-Agent nell'intestazione delle richieste. Inoltre il WAF monitora anche la presenza di eventuali richieste con parametri i cui valori corrispondono ad una serie di *pattern* noti, di fatto funzionando come un IDS di tipo *misuse detection* sviluppato specificatamente per le applicazioni web e il protocollo HTTP in generale. Il sistema antifrode esamina quindi il log prodotto dal WAF assieme a tutti gli altri dettagli di ogni sessione per fornire elementi di analisi agli *auditor* ed elevare il livello di rischio in presenza delle casistiche sopra descritte.

7.4.2 Ulteriori informazioni per gli *auditor*

Account in osservazione Sebbene non ci siano stati registrati casi di questo tipo tra le frodi reali osservate è possibile che un certo account possa essere oggetto di più tentativi di frode nel breve periodo ad esempio se il computer della vittima risulta ancora compromesso. Questa informazione indica agli *auditor* di prestare maggiore attenzione nell'investigazione dei casi sospetti legati a questi account.

Attività sospette dell'account Ci sono alcune attività che i frodatori eseguono all'interno di una navigazione automatizzata dal *banking trojan* che permettono normalmente ai cybercriminali di innalzare le probabilità di successo di un tentativo di frode. In particolare nel corso dell'analisi di alcuni casi reali abbiamo potuto notare come un primo obiettivo del frodatore, una volta preso il controllo della sessione utente, era quello di disattivare eventuali notifiche per SMS delle varie operazione dispositiva o di modificare il numero di cellulare a cui inviare tali notifiche. In questo modo il cybercriminale cercava di garantirsi un tempo superiore prima dell'individuazione della frode,

sufficiente a consentirgli di entrare effettivamente in possesso del mal tolto. Questo tipo di attività ha però un vantaggio per chi controlla sulla legittimità delle operazioni: è infatti molto semplice da individuare. Attraverso il controllo di tutte le pagine visitate da un utente è semplice per il sistema verificare se durante la sessione sono state portate a termine questa o altre operazioni simili atte a ridurre il livello di sicurezza dell'utente e a minare l'affidabilità del processo di autenticazione dei trasferimenti. Si potrebbe affermare che queste eventualità dovrebbero essere già individuabili attraverso l'analisi del profilo di navigazione; sebbene ciò sia vero risulta utile segnalarle puntualmente agli *auditor*. Il modello di navigazione infatti non è in grado di descrivere in un linguaggio naturale il motivo per cui una certa navigazione è da considersi sospetta ma si limita a fornire un valore di probabilità. Così facendo invece uniamo a quell'analisi un valore semantico che facilità l'operatività del personale predisposto all'investigazione dei casi e migliora la comunicabilità di una diagnosi, aspetto che non è da trascurare in ambito aziendale.

7.4.3 Monitoraggio delle transazioni

L'obiettivo primario del nostro sistema antifrode è quello di monitorare le operazioni che coinvolgono trasferimenti di denaro disposte da un utente individuando eventuali tentativi di frode. Questo obiettivo viene raggiunto attraverso i seguenti punti:

1. Ad intervallo orario il sistema programmaticamente provvede come descritto a recuperare dal database centrale di produzione i dati relativi alle ultime transazioni inserite

2. Il sistema effettua una pre-elaborazione:

   - Vengono scartate tutte quelle operazioni giudicate "non interessanti" secondo le euristiche precedentemente descritte
   - Eventuali operazioni destinate ad account già presenti nella *blacklist* del sistema vengono immediatamente segnalate

3. Di ogni transazione rimasta viene determinato l'utente che l'ha effettuata e recuperata dai file di log dei web server la sessione di navigazione nel portale nel corso della quale è stata prodotta la richiesta di elaborazione della transazione stessa

4. Dal database dei profili caricato il modello di navigazione dell'utente che ha ordinato l'operazione
    - Se il modello di navigazione non esiste o risulta troppo datato (più di un mese) viene (ri)creato *on-the-fly* e memorizzato
    - I dati considerati corrispondono alle sessioni di navigazione comprese in un periodo che va da 5 mesi prima a un mese prima della transazione verificata
    - Nel caso non sia possibile trovare un numero di sessioni superiore o pari a 10 il modello non viene generato
5. La sessione utente viene testata utilizzando il modello di navigazione producendo come output un valore in $[0, 1]$ che rappresenta il livello di rischio della transazione secondo questo modello
6. Il sistema genera il modello di spesa dell'utente a partire dalle precedenti transazioni
    - Se l'utente non ha effettuato un numero minimo di operazioni in passato (5 nell'implementazione) il modello non viene generato
7. L'importo della nuova transazione viene testato attraverso il modello di spesa producendo un valore in $[0, 1]$ che ne rappresenta il livello di rischio secondo tale modello
8. L'output dei due modelli viene combinato producendo una valutazione di rischio finale
    - Tale livello di rischio è calcolato con la seguente formula (cfr. Sezione 6.1):
    - Risk level = Anomaly score $*$ *importo*
9. Il sistema genera un report contenente un numero massimo di T transazioni ordinate secondo il livello di rischio calcolato
10. Il report viene inviato tramite e-mail ad un gruppo di *auditor*

È doveroso a questo punto dare alcune precisazioni. Innanzitutto, data la rilevanza di tale tipologia di operazioni tra quelle maggiormente sfruttate dai frodatori, il sistema, nel suo stato di prototipo, monitora solamente i trasferimenti di denaro tramite

bonifico bancario. Proprio per l'incidenza dei bonifici, ad oggi il mezzo prediletto dai cybercriminali per perpetrare le frodi attraverso i sistemi di home banking, non consideriamo tale scelta come limitante. Con questa motivazione bene in mente è comunque utile affermare come il monitoraggio di operazioni di diversa tipologia (come ricariche telefoniche o trasferimenti su carta di credito) possa essere implementato in futuro; l'inserimento di una tale funzionalità dovrebbe prevedere una fase di studio preliminare per valutare l'introduzione di modelli dispositivi separati o per sviluppare un modello globale per tutti i tipi di operazione.

Infine è importante sottolineare come il sistema, per sua costruzione, non possa analizzare quelle transazioni ordinate da un utente per i quali non siano presenti nel sistema informativo un numero sufficiente di dati storici per la creazione di uno o più profili descrittivi. Queste transazioni devono perciò essere analizzate diversamente; la soluzione utilizzata nella nostra implementazione prevede l'esecuzione delle euristiche specificate in Sezione 7.4.1 mentre un filtro sulla base dell'importo può essere specificato per eliminare dall'analisi le transazioni al di sotto di una soglia di importo configurata.

7.5 Console investigativa e di amministrazione

Il nostro sistema software comprende una console di amministrazione con interfaccia Web per la gestione degli *incident*, ovvero le segnalazioni prodotte dal sistema. La scelta di realizzare un'interfaccia di questo tipo è stata dettata in primo luogo dalla semplicità di sviluppo data dalle moderne tecnologie Web che ci ha consentito di creare un prototipo funzionante della console in tempi ridotti. Il secondo vantaggio è dato dalla facilità di accesso di una soluzione *browser-based* che consente l'utilizzo della console da più postazioni senza la necessità di installare software aggiuntivo.

La console è stata sviluppata utilizzando il linguaggio di programmazione PHP sulla base del *framework* MVC[1] CakePHP[2]. L'utilizzo di PHP ha semplificato la fase di *deploy* dell'applicazione essendo già disponibile nella macchina di test un server Web con supporto a tale tecnologia[3]. Date le sue caratteristiche di semplicità di sviluppo e l'ottima documentazione è inoltre un linguaggio particolarmente adatto per la prototipizzazione in ambito Web. L'interfaccia si collega al database SQLite utilizzato dal

[1] *Model View Controller*, una particolare architettura molto utilizzata nello sviluppo Web
[2] CakePHP - The rapid development php framework `http://cakephp.org/`
[3] Apache Server 2.x con modulo PHP

sistema antifrode; per migliorare le prestazioni di ricerca all'interno del gran numero di record sono stati creati alcuni indici. Le prestazioni di questo strumento sono un aspetto molto importante che ne impattano l'efficacia nel suo complesso; tempi di risposta troppo lunghi infatti diminuirebbero la qualità dell'analisi manuale e in ultima istanza renderebbero di fatto il sistema poco pratico per l'utilizzo quotidiano.

Le funzionalità offerte dal pannello di controllo sono le seguenti:

- ricerca delle transazioni per account, istituto e data
- visualizzazione, modifica e inserimento di record nella *blacklist*
- segnalazione manuale degli *incident*
- visualizzazione delle informazioni relative alla sessione di navigazione di una determinata transazione
- configurazione dei parametri del sistema
- visualizzazione di alcune informazioni statistiche come numero casi di frode per istituto e mensili (Figura 7.6)

Oltre a servire come punto di raccolta e ispezione dei vari casi di frode (accertati o in fase di accertamento) la console funge quindi anche da strumento di investigazione consentendo ad un *auditor* di ispezionare manualmente e in maniera comprensiva un evento potenzialmente fraudolento fornendo allo scopo tutte le informazioni necessarie come lo storico delle transazioni di un determinato account, il profilo dell'account stesso e i log di navigazione processati dal nostro sistema per una maggiore leggibilità. Per quanto riguarda in particolare il profilo dell'accout vengono fornite alcune informazioni personali del cliente, l'istituto e la data di attivazione. Un ulteriore utile elemento da inserire è il bilancio dell'account, in correlazione alla necessità dei frodatori di guadagnare il più possibile da una singola frode: un'ipotesi di frode può essere meglio supportata in caso di rapida estinzione del bilancio. Tale funzionalità potrà essere incorporata in una versione successiva del software.

Particolare attenzione è stata posta nel tentativo di provvedere alla realizzazione di un sistema di rappresentazione che fosse in grado di suggerire visivamente agli *auditor* il livello di anomalia riscontrato all'interno della navigazione senza appesantire

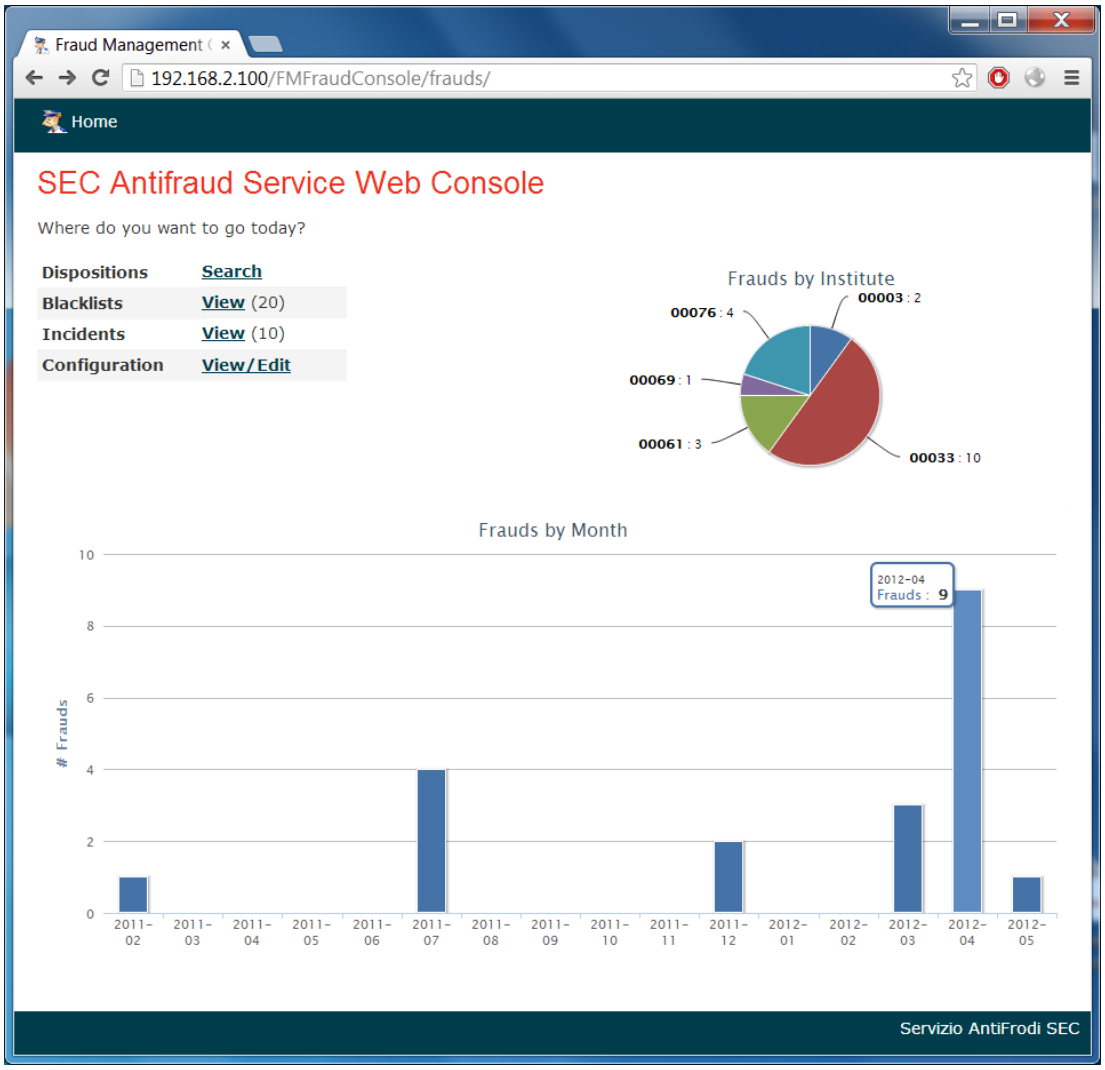

Figura 7.6: La schermata principale del pannello di controllo mostra alcune importanti statistiche

lo strumento (a livello della quantità di informazioni) o diminuirne l'usabilità. Accanto alla lista di tutte le transazioni effettuate da un particolare utente un *link* porta l'*auditor* in una schermata (vedi Figura 7.7) che riassume il log di navigazione dell'utente. Inoltre l'utilizzo della colorazione per lo sfondo delle righe facilita l'individuazione in maniera semplice e immediata dei punti maggiormente discordanti dal comportamento modellato dell'utente; gradazioni sempre più forti di rosso denotano i punti più critici.

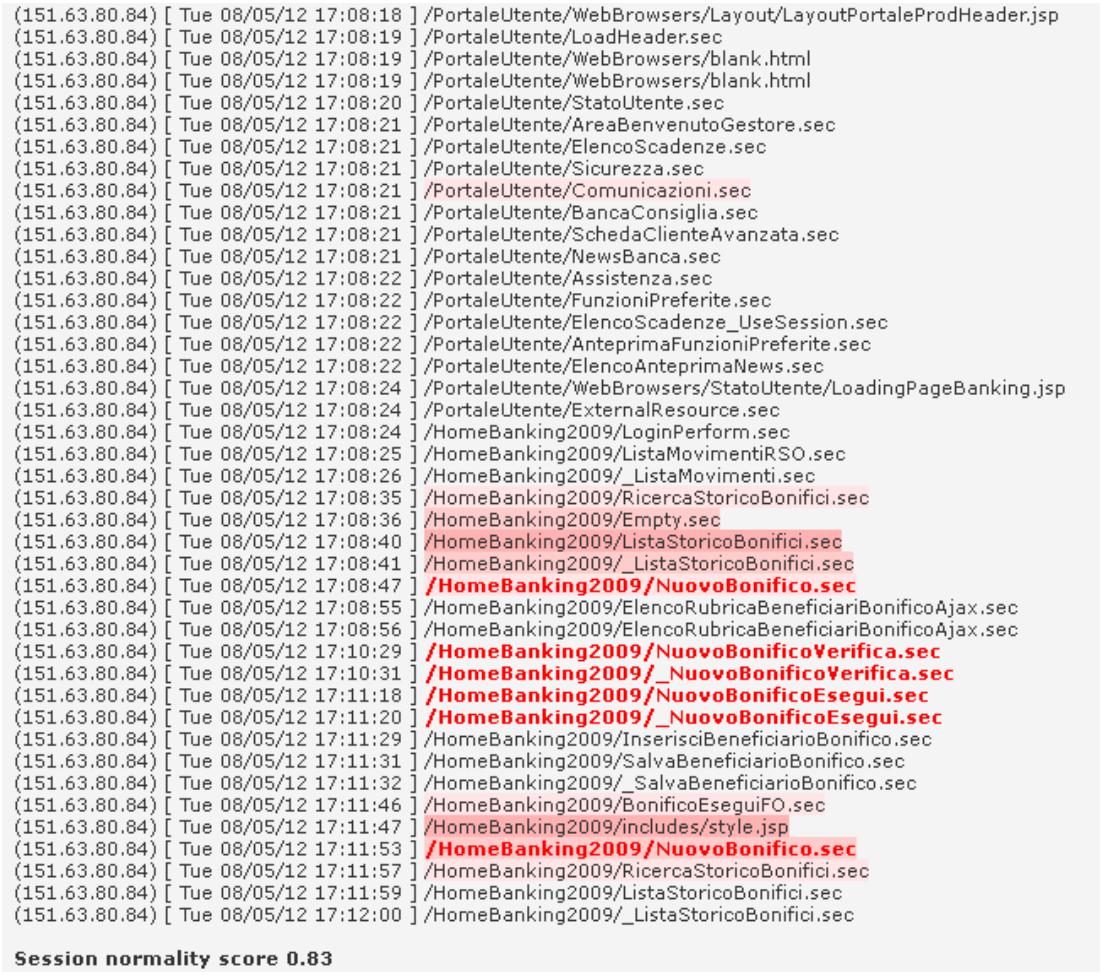

Figura 7.7: Una sequenza di navigazione di esempio. Le scritte in grassetto rosso indicano le pagine attenenenti alle richieste esecutive delle transazioni. L'utilizzo di diverse sfumature di rosso denota i punti più o meno critici della navigazione in relazione al modello del particolare utente memorizzato nel sistema

Trattandosi di un prototipo non sono state implementate alcune funzionalità certamente necessarie in questo tipo di strumenti come l'accesso riservato soltanto agli

auditor o l'integrazione con altri strumenti per la segnalazione automatica di una certa transazione fraudolenta in modo da avviare una procedura di blocco dell'account coinvolto o altro tipo di politica. Nonostante questo il pannello di controllo è stato utilizzato con soddisfazione anche dagli stessi operatori che lo hanno potuto testare in quanto in grado di fornire un'interfaccia comoda per analizzare in pochi passi i casi sospetti riportati, cosa che prima non era possibile non esistendo in azienda uno strumento che aggregava i dati transazionali con quelli di navigazione.

8

Risultati

In questo Capitolo presenteremo gli esperimenti svolti e i risultati ottenuti ottenuti dal nostro sistema di monitoraggio.

8.1 Validazione

In questa fase di validazione consideriamo il sistema ridotto ai suoi minimi termini. Per quanto riguarda il *core* valuteremo l'output di entrambi i modelli mentre abbiamo ritenuto opportuno, in questa prima fase di sperimentazione, non inserire nell'analisi l'apporto di alcune delle euristiche delineate precedemente; la motivazione dietro questa scelta è riportata in Sezione 8.2.

In fase di test il sistema è stato configurato in questo modo:

- **account noto**: una transazione verso un account noto viene sempre considerata legittima
- **account affidabile**: una transazione verso un accout affidabile viene sempre considerata legittima

In particolare per account noto intendiamo un account al quale l'utente abbia già, in passato, trasferito del denaro. Per tutelare il sistema dall'abuso di questa euristica la transazione precedente non deve essere più recente di un mese (cfr. 7.4.1). Infine per il calcolo degli accout affidabili abbiamo impostato $min\_users = 7$ e considerato solamente le transazioni memorizzate fino ad un mese prima del periodo di test (cfr. Sezione 7.4.1 per una giustificazione di questa scelta).

Sfortunatamente non abbiamo a disposizione un numero elevato di casi di frode per effettuare una taratura precisa del sistema. Un primo obiettivo dei nostri esperimenti sarà quindi di determinare delle soglie di anomalia opportune a seconda dei risultati ottenuti nell'applicazione dei modelli comportamentali a questi casi di frode. Ricordiamo che il nostro sistema utilizza i dati storici degli utenti per determinare il livello di anomalia. Il numero quindi di casi di frode per cui disponiamo di sufficienti dati storici per entrambi i modelli è 8.

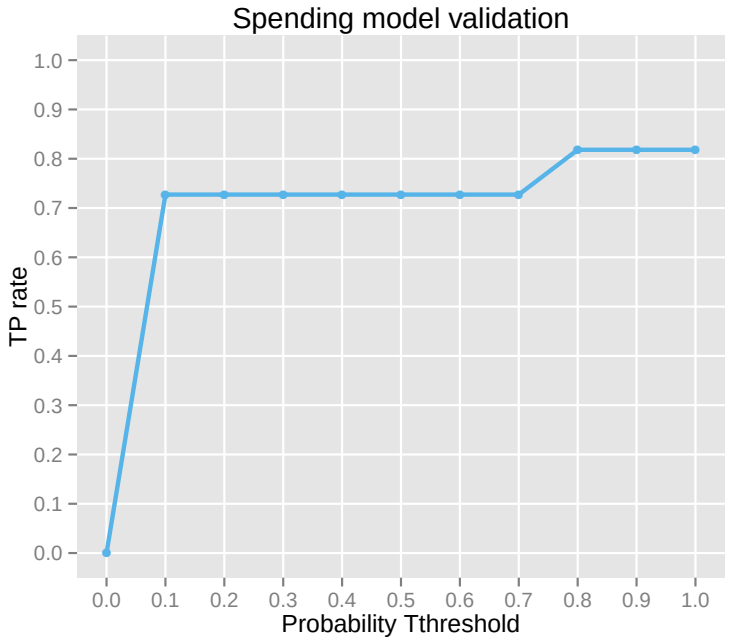

Figura 8.1: Andamento del TP rate del modello di spesa al variare della soglia di probabilità

In Figura 8.2 è riportato un grafico che mostra l'andamento del *true positive rate* applicato al solo modello di spesa. L'analisi del grafico, pur essendo basato su un ristretto numero di campioni, mostra come il modello sia efficace nell'individuare l'insorgenza di frode. In particolare si vede come la sensitività del modello sia alta per la maggior parte della scala di valori di probabilità minima, indicando una tendenza delle frodi ad avere un livello di probabilità prossimo allo 0 secondo il modello di spesa. Purtroppo, qualunque soglia si applichi, il modello non risulta in grado di individuare un 20% circa delle frodi da noi analizzate (2 campioni). In un caso l'importo della

frode era molto basso (1.100 €circa) e inferiore alla media mentre nell'altro l'importo rientrava nel modello dell'utente.

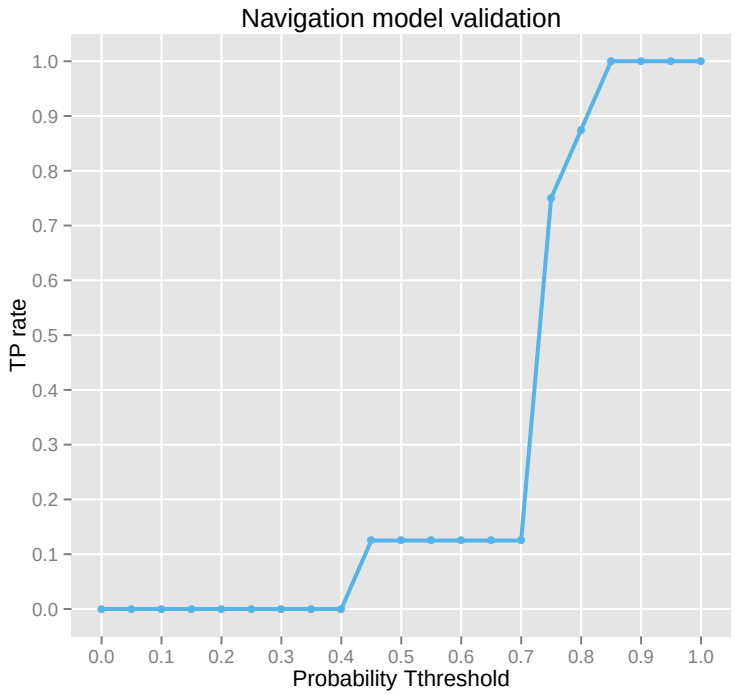

Figura 8.2: Andamento del TP rate del modello di navigazione al variare della soglia di probabilità

Possiamo produrre un simile grafico anche per il secondo modello, quello di navigazione. Si nota come questo modello possegga una sensitività inferiore rispetto al modello di spesa. Infatti il *true positive rate* decresce molto rapidamente se si imposta una soglia di probabilità inferiore a 0.75.

Pur avendo a disposizione pochi dati e avendo quindi potuto effettuare una sperimentazione limitata possiamo comunque fare una prima riflessione. Gli esperimenti sembrano indicare una maggior efficacia del modello di spesa nell'individuare i casi di frode. Una ragione plausibile per differenza di prestazioni con il modello di navigazione è che, nei casi da noi analizzati, l'attività del banking trojan sia stata ridotta ai minimi termini andando ad incidere meno sulla sequenza di pagine richieste piuttosto che sul livello della cifra rispetto al comportamento normale della vittima. Nonostante questo i

nostri esperimenti dimostrano la necessità di combinare entrambi i modelli per ottenere un *true positive rate* del 100%.

Ovviamente non è sufficiente considerare il tasso di veri positivi per giustificare la bontà di un sistema di fraud detection basato sul machine learning. Il successivo passo per la validazione del sistema consiste nel valutare il *false positive rate*, ovvero il tasso di esemplari legittimi classificati erroneamente. Nel valutare questo numero abbiamo impostato, secondo i risultati riportati dagli esperimenti precedenti, la soglia minima di probabilità calcolata dal modello di spesa a 0.9 e a 0.75 per il modello di navigazione. Questa configurazione consente di ottenere un 100% di frodi individuate tra quelle esaminate ed è quindi interessante esaminare il numero di falsi positivi che si ottengono impostando questi valori.

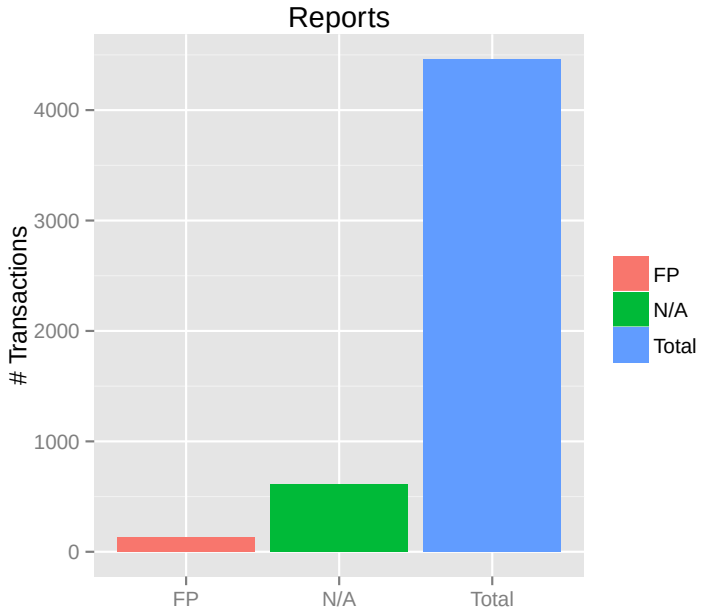

Figura 8.3: Falsi positivi ottenuti dal sistema in un giorno di operatività. Con l'etichetta "N/A" indichiamo quelle transazioni per cui non è stato possibile eseguire un'analisi data la mancanza di dati storici

L'esperimento è stato condotto analizzando i risultati dell'esecuzione del sistema nell'arco di un'intera giornata per un totale di 4464 transazioni. Con la configurazione sopracitata si ottiene un tasso di falsi positivi del 2,97% che si traduce in un numero

assoluto pari a 133. Se consideriamo il numero di istituti coinvolti nel progetto, 12, otteniamo una media di circa 11 casi da analizzare per istituto in una giornata, la cui investigazione può essere agevolmente eseguita da un singolo operatore per banca. I risultati sono esposti nel grafico in Figura 8.3.

Per focalizzare l'investigazione sulle transazioni più costose è possibile definire una valutazione di rischio, uno *score*, e ottenere da questa un ranking. Quella da noi proposta prevede di pesare la somma algebrica dei livelli di anormalità ottenuti dai due modelli con l'importo della transazione, come indicato in Eq. 8.1 dove p_m è il livello di probabilità calcolato dal modello m. Impostando $w_m = 1$ per ogni modello si ottiene l'effetto di considerare allo stesso modo il risultato dei due modelli per il calcolo del punteggio. Una specifica implementazione potrebbe però prevedere pesi differenti.

$$score = importo \cdot \sum_{m \in M} w_m(1 - p_m) \tag{8.1}$$

Il singolo istituto potrà poi definire un livello minimo di *score* o, alternativamente, indicare il numero massimo k di casi da investigare giornalmente, ricavando i top-k dall'ordinamento proposto. Quest'ultima opzione risulta valida soprattutto nel caso l'istituto intenda assegnare specifiche risorse all'investigazione quotidiana delle transazioni. Un approccio simile è stato proposto in [103].

Dedichiamo quest'ultima parte della rassegna dei risultati ai limiti del sistema. Purtroppo al momento un buon numero di transazioni giornaliere, circa il 13% (indicate con l'etichetta "N/A" nel grafico di Figura 8.3), non possono essere correttamente analizzate dal sistema così come descritto. La motivazione sta nelle ridotte informazioni a disposizione per il calcolo dei profili di alcuni utenti. Infatti non è rara, come si può vedere dai dati numerici, la situazione in cui non si disponga di sufficienti informazioni storiche per calcolare valori statisticamente significativi in merito alle abitudini di spesa. Al momento il sistema infatti calcola un profilo di spesa solamente per gli utenti che abbiano inserito almeno 5 transazioni. Il dataset da noi utilizzato per le analisi consta delle navigazioni registrate nell'arco di un anno. Sfortunatamente anche considerando un periodo così lungo il sistema non dispone di un numero di sessioni valide memorizzate per costruire un modello di navigazione di ogni utente (attualmente, in produzione, il sistema è impostato per richiedere un numero minimo di 10 sessioni). Questa mancanza rende impossibile valutare con affidabilità alcune delle nuove sessioni

limitando le capacità di analisi del sistema. Ovviamente non è pensabile trascurare queste transazioni. La ricerca futura dovrà quindi affrontare il problema di individuare una tecnica valida per la valutazione di questi casi in condizione di *undertraining*[94].

8.2 Ulteriori valutazioni

Abbiamo escluso in questi esperimenti l'utilizzo di molte delle euristiche delineate nel Capitolo 7. La motivazione riguarda essenzialmente la mancanza di dati, e quindi evidenze sperimentali, per poterne valutare l'efficacia. Cionondimeno consideramo ragionevole il loro utilizzo in un sistema di monitoraggio in quanto esse derivano da uno studio della letteratura e dall'analisi dello scenario attuale delle frodi nel settore dell'online banking. Il rilevamento di una velocità di navigazione anomala, nonostante sia stata segnalata in un solo episodio di frode tra quelli analizzati, è però supportata dalla conoscenza della modalità di funzionamento di alcuni banking trojan. Proprio per la contemporaneità tra attacco e utilizzo legittimo del portale di home banking sembra invece meno probabile l'eventualità di richieste HTTP ravvicinate provenienti da aree geografiche molto distanti tra loro. L'euristica di geolocalizzazione delle connessioni, cui compete proprio questa analisi, è stata indicata anche in [32]. Dal 2006, anno di pubblicazione dell'articolo, lo scenario delle frodi è molto cambiato e lo stato dell'arte delle attuali tecniche utilizzate dai cybercriminali è in grado di eludere questo tipo di controllo. Nonostante questo consideriamo comunque interessante l'implementazione dell'euristica nel sistema in quanto in grado di individuare con molta facilità i tentativi di frode meno sofisticati. Lo stesso argomento vale anche per un'altra delle euristiche sviluppate, vale a dire il controllo della presenza di richieste dello stesso utente effettuate con browser diversi nell'arco di una singola sessione di navigazione.

Sicuramente opportuno è invece l'utilizzo nel sistema finale di una blacklist che contenga gli IBAN degli account dei cosiddetti *money mules*. Si tratta di uno strumento che non complica l'implementazione del sistema ma che è in grado, con un semplice controllo, di individuare una nuova frode con trasferimento di denaro verso un account precedentemente utilizzato dai frodatori come intermediario.

Abbiamo ignorato finora un aspetto molto importante riguardo alle prestazioni del sistema, vale a dire i tempi di risposta. Tra gli obiettivi della tesi si richiedeva l'implementazione di un sistema in grado di rispondere in tempi rapidi ad un tentativo

di frode. I risultati in questo senso da noi ottenuti sono soddisfacenti. L'esecuzione oraria determina un tempo massimo di risposta che è appunto di un'ora a cui si deve sommare però il tempo di analisi delle nuove transazioni. Questo tempo, considerando sia il trasferimento dei log sia le query al database del sistema informativo SEC, non supera i 10 minuti. La segnalazione della frode quindi avviene al più in 70 minuti, un tempo più che ragionevole considerate le procedure di elaborazione attuali delle transazioni. A questi tempi va aggiunto l'eventuale tempo di investigazione da parte di un operatore. La nostra console di amministrazione ha tra i suoi compiti quello di aiutare un *auditor* nel suo compito, velocizzando i tempi di analisi di ogni caso. Un test di usabilità potrà essere attuato per verificare la validità dello strumento ed eventualmente evidenziare i punti di miglioramento.

9

Conclusioni

La crescente sofisticazione delle frodi perpetrate attraverso i moderni portali di home banking impone agli istituti finanziari l'introduzione di un ulteriore strato di protezione, da affiancare agli attuali sistemi di fraud-prevention.

Questo lavoro di ricerca, per la realizzazione di un sistema di monitoraggio anti-frode, nasceva dalla volontà di mettere a frutto la grande quantità di dati storici a disposizione delle banche per facilitare l'individuazione di nuove transazioni fraudolente. Ad alto livello ci si proponeva cioè di determinare, a partire da tali dati, una descrizione formale del comportamento di ogni utente e di segnalare, all'atto dell'analisi di una nuova transazione, scostamenti significativi da tale formalizzazione, imputabili a possibili tentativi di frode. Alla luce di queste premesse l'utilizzo di un'architettura basata su tecniche mutuate dall'anomaly detection è sembrata quindi una scelta naturale nel determinare il *core* del nostro sistema di monitoraggio. La scarsa disponibilità di dati relativi ad episodi di frode reali, e il conseguente problema dello sbilanciamento delle classi, di fatto compromettevano in partenza l'efficacia dell'utilizzo di tecniche di apprendimento supervisionato, spesso trattate nella letteratura relativa alle applicazioni del machine learning alla fraud detection, in special modo nel settore delle carte di credito.

L'efficacia del sistema sviluppato è dimostrata dai due casi di frode che ha consentito di individuare, per un valore complessivo di oltre 10.000 €. Sfortunatamente (o, meglio, fortunatamente) il numero così ridotto di episodi fraudolenti reali a disposizione non ci permette del tutto di trarre conclusioni generali relativamente alla bontà dello strumento. Il passo successivo per arrivare a quest'obiettivo di valutazione dovrà

prevedere il recupero di un maggior quantitativo di dati relativi a episodi di frode, tale da garantire una sperimentazione più dettagliata. Purtroppo tali dati non sono facilmente reperibili nel contesto applicativo, sia perchè in SEC Servizi i casi di frode bancaria online non sono stati molti (circa una ventina quelli documentati) sia perchè l'utilizzo di dati, soprattutto relativi alle sessioni di navigazione, provenienti dai log di portali di home banking alternativi dovrebbe prevedere un'analisi iniziale che ne valuti l'applicabilità all'interno del nostro modello DFA. Un fattore determinante è la presenza o meno di un'etichettatura delle richieste contenute nei log senza la quale il partizionamento delle richieste in base all'utente risulterebbe un'operazione priva delle necessaria affidabilità. Questo limite deriva anche dal fatto che il sistema di monitoraggio è stato sviluppato per operare in un preciso contesto, ovvero internamente al sistema informativo di SEC Servizi. Lo sviluppo di un sistema più generale è uno dei possibili temi per un'ulteriore lavoro di ricerca.

Per quanto riguarda la profilazione i risultati ottenuti dalla nostra analisi evidenziano una maggior sensitività del modello di spesa rispetto a quello di navigazione benchè nessuno dei due sia in grado, individualmente, di ottenere una percentuale nulla di falsi negativi. Ciò testimonia la validità di un approccio multi-modello. La ricerca futura dovrà concentrare nuovi sforzi nel determinare modelli ancora più precisi. Ad esempio potrebbe essere interessante analizzare le caratteristiche di un approccio basato su reti neurali, in particolare la variante auto-associativa [33]. Il framework predisposto è ovviamente estendibile con nuovi modelli. L'osservazione di ulteriore dati potrebbe rendere evidenti nuove *features* da modellare (e.g. testo nella causale del pagamento, *digital analysis* [58], ...).

Benchè in termini percentuali il tasso di falsi positivi generato dal sistema non sia particolarmente alto (inferiore al 3%) in termini assoluti, considerato che in una giornata il sistema informativo può processare anche 15.000 transazioni, ciò si traduce in un numero di 450 potenziali casi da investigare quotidianamente. Tali casi sono distribuiti però tra 12 diversi istituti bancari con una media dunque inferiore ai 40 giornalieri circa per ognuno. In questi termini lo sforzo di analisi può essere quindi ridotto e assegnato ad un singolo auditor incaricato allo scopo all'interno del personale di ogni istituto. Cionondimeno è auspicabile l'individuazione di una strategia per ridurre ulteriormente il tasso di falsi positivi. Il reperimento di nuove informazioni circa episodi

fraudolenti potrebbe aiutare a stabilire delle soglie di anomalia più precise e a individuare una configurazione del sistema con il compromesso desiderato tra true positive rate e false positive rate. Come detto però la disponibilità di queste informazioni non è affatto scontata e potrebbe rivelarsi preferibile perseguire la strada di determinare una funzione di costo che, data un numero fisso di N casi giornalieri riportati, consenta di garantire la presenza di una transazione fraudolenta tra queste N presentate con sufficiente probabilità.

Un altro aspetto critico del sistema è la riduzione drastica, nei termini definita dal framework multi-modello, della capacità di analisi di tutte quelle transazioni che non sono supportate da una sufficiente quantità di dati storici, siano essi vecchie transazioni dell'utente o tracce della sua attività online. Questa eventualità si presenta spesso; circa il 10% delle transazioni nell'arco di una giornata sfugge quindi all'analisi comportamentale. È doverosa una precisazione: il nostro sistema prototipo ha accesso solamente ad una piccola porzione dei dati storici conservati all'interno dei database di produzione di SEC Servizi. Siamo convinti che un'integrazione più stretta tra il sistema di monitoraggio e il sistema informativo esistente garantirebbe la fruibilità di tutte le informazioni in realtà disponibili senza incorrere per questo in un abbattimento delle prestazioni. Grazie all'accesso a tali dati la generazione di profili di spesa attendibili potrebbe essere applicata ad un numero superiore utenti. Per quanto riguarda invece i dati di navigazione non è consigliabile procedere all'esame di dati storici in quanto il comportamento stesso dell'utente nell'utilizzo del portale dipende dalla struttura del sito che è stata spesso modificata negli ultimi anni rendendo i dati archiviati di fatto inutilizzabili.

Si pone comunque il problema di analizzare in maniera efficace le transazioni inoltrate da utenti principianti, ovvero utenti che utilizzano per la prima volta il portale di home banking o che per la prima volta effettuano un pagamento online. Ovviamente per questi utenti non è possibile generare alcun profilo descrittivo data la totale mancanza di dati. La soluzione adottata nell'implementazione del nostro sistema prevede l'applicazione di una serie di euristiche che determinano il livello di rischio della transazione tenendo conto di fattori quali il Paese verso cui viene trasferito il denaro, l'ammontare della transazione o l'eccessiva velocità di inserimento della transazione tramite il portale, possibile segno dell'intervento di un trojan automatizzato. Questo tipo di approccio non è però soddisfacente e va contro alcuni degli obiettivi principali

soddisfatti grazie all'utilizzo di tecniche di anomaly detection, vale a dire la possibilità di individuare casi di frode diversi da quelli registrati e l'alienazione da qualsiasi attività di scrittura manuale di regole dettate dall'esperienza nel dominio. Un possibile input ad una nuova fase di ricerca proprio su questo tema potrebbe venire da [94]. In questo articolo viene affrontato il problema dell'*undertraining* (così si definisce in gergo questa situazione) in un diverso dominio, l'intrusion detection per le applicazioni web. Non è chiaro però quanto un approccio come quello presentato possa essere portato nel nostro contesto con sufficiente affidabilità.

Tra gli obiettivi della tesi vi era infine lo sviluppo di un'interfaccia grafica atta a favorire l'investigazione dei casi di frode riportati, in maniera usabile ed efficace. Il sistema da noi proposto è costituito anche da un tale componente, rappresentato da una GUI web-based semplice ma funzionale. Il pannello di controllo mostra alcuni dati di sommario, consente di popolare una blacklist dei casi di frode e di ispezionare i dati relativi alle varie transazioni processate dal sistema di monitoraggio. L'auditor può usufruire di questi dati con tempi di risposta molto rapidi che favoriscono l'analisi e, nel tempo, non compromettono l'utilizzo effettivo del pannello. Una sezione dedicata all'analisi visuale delle anomalie nella navigazione consente di semplificare il lavoro dell'auditor aumentandone la produttività. Riteniamo che una maggiore integrazione dell'intero sistema di monitoraggio con il sistema inforamtivo di SEC possa favorire lo sviluppo di ulteriori funzionalità per quanto riguarda questa interfaccia grafica, prima fra tutti la possibilità di inibire manualmente l'elaborazione di una transazione considerata sospetta, rispettando ovviamente le policy aziendali.

Appendice A

Il formato *gzip* e l'estensione *dictzip*

Questa appendice descrive il formato *dictzip* e la sua applicazione nel progetto.

A.1 Alcuni cenni riguardo DEFLATE

I file *gzip* rappresentano un formato per la memorizzazione di dati compressi particolarmente utilizzato nel contesto UNIX/Linux, molto efficace nella compressione di file testuali come ad esempio i file di log prodotti da alcuni noti Web server. Il *payload* contenuto all'interno di questi file è generato applicando al file di input l'algoritmo DEFLATE, sostanzialmente una combinazione di LZ77 e della classica codifica di Huffman, che produce una serie di blocchi autoterminanti ognuno dei quali è preceduto da un *header* che ne specifica la codifica utilizzata. La compressione di un blocco prevede due fasi separate. Nella prima fase i dati vengono processati attraverso una procedura basata su LZ77. Non è questa la sede per illustrare il funzionamento di tale algoritmo di compressione dati[1] ma è necessario quanto meno fornire una spiegazione illustrativa. In LZ77 la compressione avviene sostituendo stringhe di bit di con delle coppie (*puntatore*, *lunghezza*) dove la lunghezza rappresenta il numero di bit sostituiti e il puntatore fa riferimento ad una posizione precedente nel buffer di input dove l'identica stringa di bit è stata identificata. Questo meccanismo di ricerca all'indietro, benchè possa

[1] rimandiamo il lettore curioso all'articolo *A Universal Algorithm for Sequential Data Compression* degli autori Lempel-Ziv

spaziare tra diversi blocchi, è limitato ad una finestra di 32 KB in DEFLATE. Una volta completata questa prima fase lo *stream* risultante viene ulteriormente compresso attraverso un'opportuna codifica di Huffman.

A.2 Il campo Extra Field di *gzip* e il suo utilizzo in *dictzip*

In riferimento al nostro progetto è di interesse analizzare una parte dell'intestazione di un file *gzip*. In Figura A.1 è illustrata la struttura del campo *Extra Field*, previsto dal formato come descritto nel documento RFC 1952. Per indicare la presenza del campo extra il *flag* FLG.FEXTRA deve essere impostato nell'*header* mentre la sua lunghezza viene specificata nel campo XLEN, localizzato anch'esso precedentemente nell'intestazione.

Figura A.1: Descrizione del campo *Extra Field* del formato *gzip*

Nel caso di un file *dictzip* si ha SI1 = 'R' e SI2 = 'A', ad indicare "Random Access". Dopo il campo LEN i dati sono organizzati come in Figura A.2.

Figura A.2: Struttura delle informazioni nel sotto-campo dati in un file *dictzip*

Dato che il campo XLEN è costituito da 2 soli byte la lunghezza massimo del campo è di 0xffff byte, 2 dei quali sono dedicati al sotto-campo ID (SI1 e SI2) mentre altri 2 sono riservati al campo LEN: questo consente un totale di 0xfffb byte di *payload*.

Durante la compressione il file originale è diviso in "chunks" - porzioni - ognuno dei quali è di dimensioni inferiori a 64KB e può ovviamente essere compresso in un'area che è a sua volta minore di 64KB (considerando anche il caso limite di dati incomprimibili; in generale la dimensione dei dati compressi risulta molto inferiore a quella originale). Il campo CHLEN specifica la lunghezza di un "chunk" mentre il campo CHCNT specifica il numero di "chunk" presenti. Le successive CHCNT parole di dati indicano infine la lunghezza di ciascun "chunk" dopo l'operazione di compressione.

Nel caso del software *dictzip* per effettuare l'accesso casuale (o, per meglio dire, pseudo-casuale) ai dati un valore di *offset* e un valore di lunghezza vengono forniti

a opportune procudere. Attraverso le informazioni ricavate dal contenuto dell'*Extra Field* queste ultime determinano il "chunk" in cui è localizzata la posizione iniziale dei dati richiesti e decomprimono tale porzione del file. Se necessario porzioni consecutive possono essere decompresse a loro volta. Questo meccanismo non potrebbe ovviamente funzionare se non fosse possibile "spezzare" opportunatamente l'output ottenuto dal processo di compressione. Fortunatamente *zlib*, la libreria su cui si basa effettivamente *gzip* per tutte le operazioni di de/compressione implementa un sistema di questo tipo attraverso un particolare comando denominato "Full Flush"[1] di cui presentiamo una breve descrizione. L'effetto principale di tale comando è quello di azzerare il contenuto della finestra utilizzata da LZ77 come dizionario, reimpostando l'algoritmo al suo stato iniziale; a partire dal punto dello *stream* in cui avviene il "flush" l'algoritmo di compressione non potrà dunque più utilizzare dati localizzati precedentemente nello *stream* stesso. Utilizzando opportunatamente questi punti di "flush" in corrispondenza con la terminazione di un "chunk" e attraverso la memorizzazione della loro struttura nel campo extra dell'intestazione è possibile sviluppare una strategia di accesso pseudo-casuale allo *stream* di dati compressi.

Come precedentemente accennato questa strategia è limitata dalla lunghezza massima del campo *Extra Field* e dalla seguente definizione dei campi CHLEN e CHCNT, entrambi di 2 byte. Da un semplice calcolo si ottiene che la dimensione massima di un file comprimibile da *dictzip* garantendo l'accesso pseudo-casuale ai dati è di 4GB. Per superare questo vincolo è possibile tenere conto di un'ulteriore proprietà del formato *gzip*: un file di questo tipo è in fatti diviso al più alto livello nei cosiddetti *members*. Dato che un singolo file può essere costituito da un numero indefinito di questi *members* è possibile ottenere una dimensione massima virtualmente infinita. Il software *idzip* da noi utilizzato per la compressione dei *web log*, filtrati e trasformati, utilizza proprio quest'ultima caratteristica per superare i limiti di *dictzip*; benchè la dimensione massima di 4GB fosse un vincolo tutt'altro che stringente nel nostro progetto si è preferito comunque eliminare tale barriera per facilitare ulteriori estensioni. Inoltre il software *idzip* è stato sviluppato utilizzando il linguaggio di programmazione Python permettendo un'alquanto semplice personalizzazione.

[1] Si può trovare una descrizione dei vari meccanismi di *flush* all'indirizzo `http://www.bolet.org/~pornin/deflate-flush.html`

Bibliografia

[1] **AV-TEST Institute**. http://www.av-test.org/en/statistics/malware/. 7

[2] **BartPE**. http://www.nu2.nu/pebuilder/. 49

[3] **Common vulnerabilities and exposures, 2003**. http://www.cve.mitre.org/.

[4] **Cronto: Products datasheet**. http://www.cronto.com/download/Cronto_Products_Datasheet.pdf. 54

[5] **CVE-2009-1244**. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-1244. 50

[6] **CVE-2009-1564**. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-1564. 50

[7] **CVE-2010-1142**. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-1142. 50

[8] **CVE-2012-3288**. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-3288. 50

[9] **EMV Specifications**. http://www.emvco.com/specifications.aspx. 53

[10] **Fiserv Survey Shows Online Banking Growing, Now Used by Four of Five Online Households**. http://investors.fiserv.com/releasedetail.cfm?ReleaseID=396336. Ultimo accesso: 12/02/2013. 1

[11] **Fraudster-Operated Call Centers Emerge in the Underground Economy to Facilitate Phone Fraud**. http://blogs.rsa.com/rsafarl/

fraudster-operated-call-centers-emerge-in-the-underground-economy-to-facilitate-phone-
37

[12] **FreeBSD LiveCD**. http://livecd.sourceforge.net/. 49

[13] **Knoppix**. http://www.knoppix.org/. 49

[14] **MTN Moves to Prevent SIM Card Swap Fraud in South Africa**. http://www.balancingact-africa.com/news/en/issue-no-386/telecoms/mtn-moves-to-prevent/en. 51

[15] **New Caller I.D. spoofing site opens**. http://www.securityfocus.com/news/9822. 36

[16] **New ZitMo for Android and Blackberry**. https://www.securelist.com/en/blog/208193760/New_ZitMo_for_Android_and_Blackberry. 51

[17] **The Open Web Application Security Project**. http://www.owasp.org.

[18] **Oracle VM VirtualBox**. https://www.virtualbox.org/. 50

[19] **Popularity of online banking explodes, ABA survey says**. http://www.thefreelibrary.com/Popularityofonlinebankingexplodes,ABAsurveysays.-a0272609241. Ultimo accesso: 12/02/2013. 1

[20] **QEMU**. http://www.qemu.org. 50

[21] **SIM swaps - a growing problem with a SIMple solution**. http://www.finextra.com/community/fullblog.aspx?blogid=6358. 51

[22] **SymbOS.Spitmo**. http://www.symantec.com/security_response/writeup.jsp?docid=2011-040610-5334-99. 51

[23] **SymbOS.Zeusmitmo**. http://www.symantec.com/security_response/writeup.jsp?docid=2010-093000-5257-99. 51

[24] **Tatanga Attack Exposes chipTAN Weaknesses**. http://www.trusteer.com/blog/tatanga-attack-exposes-chiptan-weaknesses. 54

[25] **TrustBar Firefox Extension**. http://trustbar.mozdev.org/. 49

[26] **Trusteer Rapport**. `http://www.trusteer.com/Products/trusteer-rapport-pc-and-mac-security`. 49

[27] **University of Cambridge Computer Laboratory - Security Group**. `http://www.cl.cam.ac.uk/research/security/banking/`. 54

[28] **VMWare Workstation**. `http://www.vmware.com/products/workstation/overview.html`. 50

[29] **ZeuS-in-the-Mobile Facts and Theories**. `http://www.securelist.com/en/analysis/204792194/ZeuS_in_the_Mobile_Facts_and_Theories`.

[30] **Zitmo hits Android**. `http://blog.fortinet.com/zitmo-hits-android/`. 51

[31] **McCartney site hacked**. *Computer Fraud Security*, **2009**(4):4, 2009. 47

[32] V. Aggelis. **Offline Internet banking fraud detection**. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, pages 2–pp. IEEE, 2006. 28, 134

[33] E. Aleskerov, B. Freisleben, and B. Rao. **Cardwatch: A neural network based database mining system for credit card fraud detection**. In *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997*, pages 220–226. IEEE, 1997. 18, 138

[34] James P Anderson. **Computer security threat monitoring and surveillance**. Technical report, Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, 1980. 23

[35] E.L. Barse, H. Kvarnstrom, and E. Jonsson. **Synthesizing test data for fraud detection systems**. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 384–394. IEEE, 2003.

[36] R.A. Becker, C. Volinsky, and A.R. Wilks. **Fraud detection in telecommunications: History and lessons learned**. *Technometrics*, **52**(1):20–33, 2010.

[37] K.B. Bignell. **Authentication in an Internet Banking Environment; Towards Developing a Strategy for Fraud Detection**. In *Internet Surveillance and Protection, 2006. ICISP'06. International Conference on*, pages 23–23. IEEE, 2006. 6, 28

[38] A. Blom, G. de Koning Gans, E. Poll, J. de Ruiter, and R. Verdult. **Designed to fail: A USB-connected reader for online banking**. In *17th Nordic Conference on Secure IT Systems (NordSec 2012)*, **7617**, 2012. 56

[39] R.J. Bolton and D.J. Hand. **Statistical fraud detection: A review**. *Statistical Science*, pages 235–249, 2002.

[40] R.J. Bolton, D.J. Hand, et al. **Unsupervised profiling methods for fraud detection**. *Credit Scoring and Credit Control VII*, pages 235–255, 2001.

[41] R. Brause, T. Langsdorf, and M. Hepp. **Neural data mining for credit card fraud detection**. In *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on*, pages 103–106. IEEE, 1999.

[42] L.D. Catledge and J.E. Pitkow. **Characterizing browsing strategies in the World-Wide Web**. *Computer Networks and ISDN systems*, **27**(6):1065–1073, 1995. 106

[43] Anti-Fraud Command Center. **RSA Online Fraud Report**. Technical report, RSA, Novembre 2008. 35

[44] Anti-Fraud Command Center. **Business Success in a Dark Market: An Inside Look at How the Fraud Underground Operates**. Technical report, RSA, Settembre 2009. vii, 31, 32

[45] Anti-Fraud Command Center. **RSA Online Fraud Report**. Technical report, RSA, Ottobre 2009. vii, 5, 47

[46] Anti-Fraud Command Center. **Fraud-As-A-Service: A Look the Fraud Business in 2012**. Technical report, RSA, Agosto 2012. 36

[47] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. **SMOTE: Synthetic Minority Over-sampling Technique**. *Journal of Artificial Intelligence Research*, **16**:321–357, 2002. 19

[48] Robert Cooley, Bamshad Mobasher, Jaideep Srivastava, et al. **Data preparation for mining world wide web browsing patterns**. *Knowledge and information systems*, **1**(1):5–32, 1999. 103

[49] I. Corona, D. Ariu, and G. Giacinto. **HMM-web: a framework for the detection of attacks against web applications**. In *Communications, 2009. ICC'09. IEEE International Conference on*, pages 1–6. IEEE, 2009. 28

[50] C. Cortes and D. Pregibon. **Signature-based methods for data streams**. *Data Mining and Knowledge Discovery*, **5**(3):167–182, 2001.

[51] C. Cortes, D. Pregibon, and C. Volinsky. **Communities of interest**. *Advances in Intelligent Data Analysis*, pages 105–114, 2001.

[52] F.F.I.E. Council. **Authentication in an Internet Banking Environment**, 10 2005. 4

[53] F.F.I.E. Council. **Supplement to Authentication in an Internet Banking Environment**, 6 2011. 4

[54] K.C. Cox, S.G. Eick, G.J. Wills, and R.J. Brachman. **Brief application description; visual data mining: Recognizing telephone calling fraud**. *Data Mining and Knowledge Discovery*, **1**(2):225–231, 1997.

[55] O. Dandash, Y. Wang, P.D. Leand, and B. Srinivasan. **Fraudulent Internet Banking Payments Prevention using Dynamic Key**. *Journal of Networks*, **3**(1):25–34, 2008. 28

[56] J.R. Dorronsoro, F. Ginel, C. Sgnchez, and CS Cruz. **Neural fraud detection in credit card operations**. *Neural Networks, IEEE Transactions on*, **8**(4):827–834, 1997. 17

[57] S. Drimer, S. Murdoch, and R. Anderson. **Optimised to fail: Card readers for online banking**. *Financial Cryptography and Data Security*, pages 184–200, 2009. 54

[58] Cindy Durtschi, William Hillison, and Carl Pacini. **The effective use of Benfords law to assist in detecting fraud in accounting data**. *Journal of forensic accounting*, **5**(1):17–34, 2004. 138

[59] O. Eisen. **Catching the fraudulent Man-in-the-Middle and Man-in-the-Browser**. *Network Security*, **2010**(4):11–12, 2010. 29

[60] T. Fawcett and F. Provost. **Adaptive fraud detection**. *Data mining and knowledge discovery*, **1**(3):291–316, 1997.

[61] FBI. **Cyber Banking Fraud: Global Partnerships Lead to Major Arrests**. `http://www.fbi.gov/news/stories/2010/october/cyber-banking-fraud`, Ottobre 2010. 5, 38

[62] A.P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner. **A survey of mobile malware in the wild**. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 3–14. ACM, 2011. 51

[63] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. **Hypertext transfer protocol–HTTP/1.1**, 1999.

[64] S. Forrest, S. Hofmeyr, and A. Somayaji. **The evolution of system-call monitoring**. In *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pages 418–430. IEEE, 2008. 27

[65] S. Forrest, S.A. Hofmeyr, A. Somayaji, and T.A. Longstaff. **A sense of self for unix processes**. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 120–128. IEEE, 1996. 27

[66] S. Fox and J. Beier. *Online banking 2002*. Pew Internet & American Life Project, 2002. 3

[67] S. Ghosh and D.L. Reilly. **Credit card fraud detection with a neural-network**. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, **3**, pages 621–630. IEEE, 1994.

[68] RedTeam Pentesting GmbH. **Man-in-the-Middle Attacks against the chipTAN comfort Online Banking System**. `http://www.redteam-pentesting.de/en/publications/MitM-chipTAN-comfort/-man-in-the-middle-attacks-against-the-chiptan-comfort-online-banking-system`, 2009. 54

[69] Anti-Phishing Working Group. **Phishing Activity Trends Report**. *Anti-Phishing Working Group, http://www.antiphishing.org/phishReportsArchive.html* , Settembre 2012. 47

[70] P. Gühring. **Concepts against man-in-the-browser attacks**. `http://www2.futureware.at/svn/sourcerer/CAcert/SecureClient.pdf`, 2006. 4, 43, 48, 49

[71] D. Hand. **Deception and dishonesty with data: fraud in science**. *Significance*, **4**(1):22–25, 2007. 13

[72] D.J. Hand. **Fraud detection in telecommunications and banking: Discussion of Becker, Volinsky, and Wilks (2010) and Sudjianto et al.(2010)**. *Technometrics*, **52**(1):34–38, 2010. 22

[73] DJ Hand, C. Whitrow, NM Adams, P. Juszczak, and D. Weston. **Performance criteria for plastic card fraud detection tools**. *Journal of the Operational Research Society*, **59**(7):956–962, 2007.

[74] K. Ingham and H. Inoue. **Comparing anomaly detection techniques for http**. In *Recent Advances in Intrusion Detection*, pages 42–62. Springer, 2007.

[75] K.L. Ingham, A. Somayaji, J. Burge, and S. Forrest. **Learning DFA representations of HTTP for protecting web applications**. *Computer Networks*, **51**(5):1239–1255, 2007. 28, 79, 81, 87

[76] N. Japkowicz and S. Stephen. **The class imbalance problem: A systematic study**. *Intell. Data Anal.*, **6**(5):429–449, October 2002. 19

[77] K.N. Karlsen and T. Killingberg. *Profile based intrusion detection for Internet banking systems*. PhD thesis, Norwegian University of Science and Technology, 2008. 30

[78] J. Kirk. **UK hails first cybercrime cooperation with banks**. *ITWORLD*, 2009. 5

[79] S. Kovach and W.V. Ruggiero. **Online banking fraud detection based on local and global behavior**. In *ICDS 2011, The Fifth International Conference on Digital Society*, pages 166–171, 2011. 28

[80] C. Kruegel and G. Vigna. **Anomaly detection of web-based attacks**. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 251–261. ACM, 2003. 27, 66

[81] C. Kruegel, G. Vigna, and W. Robertson. **A multi-model approach to the detection of web-based attacks**. *Computer Networks*, **48**(5):717–738, 2005. 27, 66, 77

[82] S. Li, S. Shah, M. Khan, S.A. Khayam, A.R. Sadeghi, and R. Schmitz. **Breaking e-banking CAPTCHAs**. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 171–180. ACM, 2010. 56

[83] P.J.G. Lisboa, B. Edisbury, and A. Vellido. *Business applications of neural networks: the state-of-the-art of real-world applications*. World Scientific Publishing Company Incorporated, 2000.

[84] W.R. Mebane Jr. **Election Forensics: Statistical Interventions in Election Controversies**. In *Annual Meeting of the American Political Science Association*, 2007. 13

[85] T. Meyer, A. Stobbe, and S. Kaiser. **Online banking and research: The state of play in 2010**. 2010. 1

[86] T. Meyer, A. Stobbe, and S. Kaiser. **Update on online and mobile banking: 47% of Germans will use online banking in 2012**. 2011. 2

[87] T. Meyer, A. Stobbe, and S. Kaiser. **Growing need for security in online banking**. 2012.

[88] Y. Moreau, B. Preneel, P. Burge, J. Shawe-Taylor, C. Stoermann, and C. Cooke. **Novel techniques for fraud detection in mobile telecommunication networks**. In *ACTS mobile summit*, 1997.

[89] R. Oppliger, R. Rytz, and T. Holderegger. **Internet banking: Client-side attacks and protection mechanisms**. *Computer*, **42**(6):27–33, 2009. 4

[90] A. Paes de Barros. **O futuro dos backdoors, o prior dos mundos**. `http://www.paesdebarros.com.br/backdoors.pdf`, 2005.

[91] C. Phua, D. Alahakoon, and V. Lee. **Minority report in fraud detection: classification of skewed data**. *ACM SIGKDD Explorations Newsletter*, **6**(1):50–59, 2004. 18

[92] C. Phua, V. Lee, K. Smith, and R. Gayler. **A comprehensive survey of data mining-based fraud detection research**. *Artificial Intelligence Review (submitted for publication)*, 2005. 14, 17, 19

[93] A.L. Prodromidis and S. Stolfo. **Agent-based distributed learning applied to fraud detection**. 1999.

[94] W. Robertson, F. Maggi, C. Kruegel, and G. Vigna. **Effective anomaly detection with scarce training data**. In *Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, CA*, 2010. 120, 134, 140

[95] Corrado Ronchi. **Web Web Browser Hardening Browser Hardening for Secure Internet Transactions**. EISST, 2009. 48

[96] A. Srivastava, A. Kundu, S. Sural, and A.K. Majumdar. **Credit card fraud detection using hidden Markov model**. *Dependable and Secure Computing, IEEE Transactions on*, **5**(1):37–48, 2008.

[97] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan. **Credit card fraud detection using meta-learning: Issues and initial results**. In *AAAI-97 Workshop on Fraud Detection and Risk Management*, 1997.

[98] S.J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P.K. Chan. **Cost-based modeling for fraud and intrusion detection: Results from the JAM project**. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, **2**, pages 130–144. IEEE, 2000. 23

[99] M. Syeda, Y.Q. Zhang, and Y. Pan. **Parallel granular neural networks for fast credit card fraud detection**. In *Fuzzy Systems, 2002. FUZZ-IEEE'02.*

Proceedings of the 2002 IEEE International Conference on, **1**, pages 572–577. IEEE, 2002.

[100] F.F.A. UK. **Fraud the Facts (2012)**. *Financial Fraud Action UK: disponibile all'URL: http://www.financialfraudaction.org.uk/download.asp?file=2696*, 2012.

[101] G. Vigna, F. Valeur, D. Balzarotti, W. Robertson, C. Kruegel, and E. Kirda. **Reducing errors in the anomaly-based detection of web-based attacks through the combined analysis of web requests and SQL queries**. *Journal of Computer Security*, **17**(3):305–329, 2009.

[102] D. Wagner and P. Soto. **Mimicry attacks on host-based intrusion detection systems**. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 255–264. ACM, 2002. 88

[103] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen. **Effective detection of sophisticated online banking fraud on extremely imbalanced data**. *World Wide Web*, pages 1–27, 2012. 29, 133

[104] T. Weigold and A. Hiltgen. **Secure confirmation of sensitive transaction data in modern Internet banking services**. In *Internet Security (WorldCIS), 2011 World Congress on*, pages 125–132. IEEE, 2011.

[105] Business Wire. **America's First Bank on the Internet, Security First Network Bank, goes on-line Oct. 18**. *Business Wire, October*, 1995. 1

[106] Business Wire. **Stanford federal credit union pioneers online financial services**. *Business Wire, June*, 1995. 1