Malware

Worm

L'altra tipologia più diffusa di malware sono i worm. La differenza con i virus sta nel fatto che i worm si replicano in maniera indipendente, non richiedono cioè di essere attaccati a un altro programma per la loro propagazione.

I worm portano con loro un **payload** (carico malevolo) che consente di effettuare dei task nascosti:

1. Aprire delle porte nascoste (backdoors)
2. Agire come dei trojan horse
3. Agire da agenti per fare dello spam (spam relays)
4. Agire da agenti per attacchi DDoS (DDoS agents)
5. Per fare mining di criptovalute

Le fasi che si hanno per i worm sono quelle standard:

1. **Probing**: individuare le possibili vie di infezione. Il worm cerca di selezionare una vittima per l'infezione.
2. **Exploitation**: ricevuta una risposta positiva dalla vittima il worm sfrutta la specifica vulnerabilità.
3. **Replication**: il worm si diffonde costruendo diverse copie di sé stesso.
4. **Payload**: compie l'azione cattiva.

Esempi di Worm attacks

Il **Morris worm** è storicamente (1988) uno dei primi esempi di worm che ha avuto parecchi effetti negativi e che si è diffuso abbastanza rapidamente: ha infettato circa 6mila macchine (il 10% di computer connessi ad Internet a quel tempo) con circa 10 mln di dollari di danni a causa del tempo di inattività e di riparazione delle macchine.

Un altro esempio storico (2001) è **Code Red** nella versione I e II: 500mila macchine danneggiate e 2.6 miliardi di dollari di danno.

Morris worm - 1988

Era composto da due parti:

1. Un programma per diffondersi: non faceva altro che cercare potenziali macchine vittime per infiltrarsi.
2. Un programma vettore: molto piccolo (solo 99 righe di codice C!), veniva compilato ed eseguito sulle macchine infette per continuare l'infezione.

Uno dei motivi del suo successo era la possibilità di sfruttare più vulnerabilità:

1. Buffer overflow su **fingerd**
2. Sfruttare una versione di **sendmail** eseguita con permessi di root

che avevano entrambi, come effetto collaterale, la possibilità di eseguire una shell con i permessi di root.

Vettore di infezione 1: la feature Debug di sendmail

Sendmail era a quel tempo il programma di default per la spedizione delle mail su sistemi Unix ed è in ascolto sulla porta 25 (quella del protocollo SMTP). Quando sendmail veniva compilato con l'opzione DEBUG accettava tra i comandi del protocollo SMTP un comando addizionale DEBUG, ottimo per il testing ma utilizzabile anche per altri fini.

In particolare, se si inviava tale comando alla macchina ospite (vittima) si otteneva la capacità di aprire una shell

su di essa e di eseguire, con i permessi di root, una lista di comandi specificati dall'attaccante.

Vettore di infezione 2: fingerd

Il comando fingerd veniva utilizzato per capire chi fosse un utente attivo su una certa macchina host, cioè con quali utenti connessi si poteva entrare in contatto.

Lanciando il comando *finger utentex* su una macchina vengono reperiti dei dettagli riguardo all'utente *utentex* come:

1. L'ultima connessione effettuata al terminale
2. La directory home
3. L'ultima mail ricevuta

Il demone fingerd si occupa di fornire queste informazioni rimanendo in ascolto sulla porta 79. Aveva però un problema di buffer overflow: il Morris worm scrivendo una stringa più grande di quella permessa dal buffer interno di fingerd (512 bytes) causava la riscrittura dell'indirizzo di ritorno di una chiamata a funzione, in modo che al ritorno della chiamata si potesse fare un jump direttamente nel codice che eseguiva la shell con i permessi di root.

Nota: in generale gli attacchi di buffer overflow servono per eseguire del codice remoto su una macchina vittima e sfruttano l'assenza di controlli abbastanza forti in modo da riscrivere la memoria e passare il controllo dell'esecuzione a pezzi di codice arbitrario.

Vettore di infezione 3: Sfruttare la fiducia nel Remote Login

Si poteva sfruttare anche la possibilità di connettersi in remoto alle macchine UNIX amiche mediante i comandi **rlogin** (remote login) o **rsh** (remote shell). Alcune macchine infatti hanno delle relazioni di fiducia prestabilite e contenute in alcuni file di sistema come:

1. /etc/host.equiv: elenca tutti gli host trusted che possono collegarsi in remoto all'host locale mediante rsh senza l'inserimento di una password. Accessibile da tutti gli utenti dell'host locale (system wide)

2. /.rhosts e ~/.rhosts: contiene una lista di combinazioni *host-user*, invece che solo di host come accade per host.equiv. Lo specifico *user* della macchina *host* ha il permesso per collegarsi in remoto mediante rsh all'host locale senza fornire una password. Vi è un file /.rhosts per ogni utente dell'host locale, accessibile solo dall'utente specifico.

Esempio: se l'host *aaa.xyz.com* contiene nel suo file */etc/host.equiv* l'host *bbb.xyz.com*, nel momento in cui riceve un rsh dalla macchina *bbb.xyz.com* da parte dell'utente Alice, essa risulta immediatamente collegata alla macchina ospite *aaa.xyz.com*.

Differenza importante tra rsh e rlogin: nel caso di rsh l'utente **già loggato** di un host trusted può richiedere l'esecuzione di comandi sull'host locale senza autenticazione, nel caso di rlogin (come anche rexec) l'utente **già loggato** di un host trusted deve anche inserire username e password dell'host locale.

Per sfruttare il meccanismo di fiducia il worm deve quindi indovinare la password dell'utente sull'host trusted in cui sta girando (con uno dei meccanismi del prossimo paragrafo) e poi può richiedere una rsh sul prossimo host. Nel caso di rlogin oltre a indovinare la password dell'utente sull'host trusted in cui sta girando deve anche bypassare il meccanismo di autenticazione di rlogin sul prossimo host. (da verificare!)

Phases of the Morris worm

Prima fase
Sulla macchina su cui gira il worm viene costruita una lista degli indirizzi IP delle macchine disponibili sulla sottorete locale (cioè delle macchine direttamente connesse alla macchina vittima) utilizzando il comando **netstat**.

Negli anni ’80 non si era sempre in connessione con altri, anzi la rete aveva una piccola banda ed era una risorsa da gestire con parsimonia. Al giorno d’oggi i worms semplicemente provano tutti i possibili indirizzi IP anziché solamente quelli delle macchine nella stessa sottorete.

Seconda fase Il worm cercava di infettare utilizzando i vettori di infezione visti prima: rsh/rlogin, finger e sendmail.

Nel caso di rsh/rlogin è prevista una fase di cracking delle password degli utenti presenti sull’host trusted per propagarsi ulteriormente.

Nota: in un sistema Unix ogni utente possiede una riga nel file /etc/passwd che, tra le tante informazioni, contiene la password cifrata (con algoritmo DES a quel tempo) e salata con il sale indicato nella stessa riga.

In particolare il Morris worm tentava i seguenti meccanismi di guessing delle password:

1. Controllare se l’account di un utente è senza password (compare come *null* nella riga di /etc/passwd)
2. Controllare che l’utente abbia scelto delle password semplici: il proprio username, il proprio username al contrario, il proprio username concatenato con sé stesso, ecc…
3. Attacco dizionario con 432 parole incluse nel worm e attacco dizionario più esteso comprendente parole in lingua inglese contenute in un file di sistema UNIX: per ogni parola dei due dizionari si usava la routine di cifratura DES con l’aggiunta del sale dell’utente da violare, osservando se il risultato finale avesse la stessa cifratura della password specificata in /etc/passwd.

Se il cracking aveva successo si cercavano altri utenti e altri host dai file rhosts e instaurando delle connessioni per propagarsi.

Nota importantissima: il worm, leggendo da un file rhosts, ottiene gli host trusted che possono eseguire comandi da remoto sull’host locale in cui il worm stesso sta girando e non il viceversa. Pertanto, il worm non necessariamente dall’host locale può eseguire dei comandi verso l’host trusted senza autenticazione. La propagazione, per avere successo, deve avere come assunzione la **reciproca fiducia**, cioè il fatto che se un host *x* è trusted per l’host locale *y* allora anche l’host locale *y* è trusted per *x* con alta probabilità.

Morris fanout attacks

Il morris worm si nascondeva abbastanza bene da un utente medio perché anche se si eseguiva un *ps*, cioè ottenere la lista dei processi attivi in quel momento sulla macchina, veniva listato come *sh*, cioè come una normale shell in esecuzione.

Quando il worm riusciva a connettersi ad una macchina mediante uno dei meccanismi di diffusione il suo processo si forkava (si sdoppiava) in modo che potesse continuare ad infettare la macchina vittima e, allo stesso tempo, continuare a ricercare ulteriori vittime sulla macchina precedente.

Altri esempi di worm attacks

Sono esistiti altri worm nel corso della storia, oltre a quello di Morris. Alcuni arrivano anche ai giorni nostri.

Nimda Worm (2001): usa cinque diversi modi per propagarsi.

SQL Slammer (2003): utilizzava un exploit sull’SQL server della Microsoft.

Conficker (2008, 2009): quello arrivato fino ai giorni nostri. Ha avuto un sacco di varianti ma continua a evolversi e a diffondersi. Ha infettato tra i 9 e i 15 milioni di computer. Ha come payload anche delle macchine che fanno spam (**spambot**) o terrorismo (**scareware**).

Nimda family

Questa famiglia è stata molto attiva e tuttora lo è.

Per propagarsi:

1. Sfruttava una certa vulnerabilità del sistema Microsoft (MS01-020)
2. Cercava di replicarsi nelle directory locali, nelle directory di rete e utilizzando delle backdoor per infettare anche i computer remoti.

Nimda worm

La vulnerabilità principale è stata pubblicata nel 2001 dalla Microsoft e riguardava un bug nella costruzione delle pagine HTML in IE. Un avversario poteva così scrivere del codice HTML per spedire dei messaggi di posta elettronica. L'esecuzione del codice, poi, partiva senza alcun controllo da parte dell'utente, nel senso che l'unica cosa che doveva fare era visualizzare quella particolare pagina web con IE.

Nimda worm (September 18, 2001)

I vettori di infezione che Nimda aveva erano:

Messaggi di posta elettronica come attachment
Ogni 10 giorni avviene la fase di spedizione di sé stesso utilizzando la rubrica di posta elettronica presente sul computer infettato. Il codice del worm viene spedito al destinatario come allegato *README.EXE*. Esso poteva diffondersi grazie alla vulnerabilità MS01-020: l'utente, visualizzando la mail sia in modalità *preview* sia nella sua interezza con Microsoft Outlook, causava l'esecuzione diretta del codice. Quindi, affinché il worm avesse effetto, non era necessario cliccare volontariamente sull'allegato.

Replicazione di sé stesso su drives condivisi in rete
Il worm effettua una ricerca nella rete LAN e si replica come un nuovo file *riched20.dll* in ogni directory di ogni macchina della rete che contenga file con estensione .doc o .eml. Quando altri utenti aprono un file .doc o .eml di queste directories mediante gli applicativi Microsoft Word, Wordpad o Outlook essi richiedono automaticamente il caricamento della libreria di sistema *riched20.dll*. Il worm infetterà così la macchina vittima su cui è stato aperto uno degli applicativi precedentemente citati.

Possibilità di creare particolari URL
Il worm sfrutta delle vulnerabilità del server web IIS di Microsoft con cui venivano corredate di default le macchine con piattaforma Windows. Con una semplice riga HTTP venivano automaticamente eseguiti dei comandi di shell.

Backdoors lasciate aperte da worm precedenti

Inserimento di codice JS in pagine web
Nimda inizia a scandire la rete Internet alla ricerca di web servers. Il worm cerca di infettare un web server utilizzando diversi buchi di sicurezza noti e, se l'operazione ha successo, modifica tutte le sue pagine web con estensione .html o .asp aggiungendo una routine JS in grado di attivare il download e l'esecuzione automatica del file *readme.eml* (versione in formato e-mail del worm). L'utente quando apre con IE una delle pagine web malevole fornite dal web server, inconsapevolmente, causa l'esecuzione del codice JS della pagina. Quest'ultimo permette così ad Outlook di aprire, in un'altra finestra, il messaggio readme.eml e di eseguire il codice del worm stesso.

Nimda worm

Nimda eseguiva anche altre due azioni per compromettere la sicurezza:

1. Condivideva il Drive C in rete: questo consentiva a qualsiasi utente di accedere ai file contenuti all'interno.
2. Creava un account Guest da aggiungere al gruppo Administrator: qualsiasi utente poteva così loggarsi nel sistema con i permessi di root.

Solitamente quando un virus inizia a diventare popolare molti utenti infettati cercano in rete dei rimedi, noti

comunemente come **fix**. Nel caso di Nimda è stato diffuso un fix malevolo (cioè agiva come trojan) che provocava ulteriori danni. Si diceva essere prodotto da SecurityFocus e TrendMicro come eseguibile *FIXNIMDA.exe, diverso dal vero fix FIXNIMDA.com* prodotto dalla TrendMicro stessa.

Research Worms

Nel tempo ci sono stati anche worm prodotti per scopi di ricerca. In questi lavori si mostrano quali le possibili tecniche ottimizzate per aumentare la velocità di infezione e la probabilità che un worm abbia maggiore effetto.

Warhol: con questo worm vengono studiate in maniera scientifica diverse possibilità di rilevare nuove vittime. In particolare, vengono introdotte delle tecniche di scansione ottimizzate:

1. **Hitlist scanning**: per velocizzare l'infezione iniziale. Molto prima che il worm venga rilasciato, l'autore del worm colleziona una lista di 10mila fino a 50mila potenziali macchine vulnerabili e con una buona connessione di rete. Il worm, una volta rilasciato presso la macchina iniziale della hitlist, inizia la scansione dell'elenco. Quando infetta una macchina divide a metà la hitlist, comunicando una metà alla macchina infettata e mantenendo con sé l'altra metà.
2. **Local subnet scanning**: si scansiona la sottorete locale. Questo è molto efficace poiché le macchine vulnerabili sono spesso raggruppate assieme e si rivela un ottimo modo per creare scompiglio nelle reti interne.
3. **Permutation scanning for complete, self-coordinated coverage**: ogni istanza del worm porta con sé una copia di una sequenza di indirizzi IP che rappresenta una permutazione dello spazio di tutti gli indirizzi IP. Ogni istanza del worm avvia la propria scansione da un IP scelto casualmente nella sequenza e prosegue con gli IP immediatamente successivi. Se l'istanza del worm colpisce una macchina già infetta può presumere che un'altra istanza del worm sta già lavorando su tale sezione della sequenza. Pertanto, salta a un altro punto casuale della sequenza e continua la scansione.

Con queste ottimizzazioni il worm può teoricamente infettare tutte le macchine vulnerabili in un tempo compreso tra 15 minuti e 1 ora.

Flash worms: lettura del paper consigliata.

Conficker (November 2008)

Ha avuto diverse varianti e ha compromesso tra i 9 e i 15 milioni di macchine nel corso del tempo.

Aveva diversi meccanismi di auto-difesa che miravano a disabilitare software in grado di rilevare o rimuovere l'infezione:

1. Disabilita le chiamate DNS ai vendors di antivirus: ciò impedisce ai software anti-malware di ottenere degli aggiornamenti, come ad esempio la possibilità di scaricare le nuove firme che gli consentirebbero di rilevare e rimuovere Conficker.
2. Disabilita i servizi di Windows Update e Windows Defender: ciò ostacola l'aggiornamento del sistema per rimediare ai danni prodotti dal worm.

I vettori di diffusione erano (effettuati in parallelo):

1. Vulnerabilità del servizio MS Server della Microsoft: con una richiesta di connessione remota (RPC) si poteva aprire la strada ad eseguire del codice sulla macchina remota senza essersi autenticati;
2. Attacco dizionario sugli account amministratori delle macchine connesse in rete;
3. Una replica del worm con estensione DLL veniva costruita su un supporto rimovibile, come ad esempio le chiavette USB.

Le macchine infette cercavano di:

1. Attaccare e installare il worm su computer della stessa sottorete inviando casualmente il codice che cercava di sfruttare il buffer overflow sulla chiamata RPC.
2. Fare il login sulle macchine condivise in rete mediante attacco dizionario, dove il dizionario era una lista

di password comuni.
3. Caricare il software malevolo nel momento in cui si collegava un drive USB e sui drives condivisi in rete.

Email Worms: Spreading as Email attachments

In quegli anni c'era un largo fiorire di worm con diversi effetti e diversi numeri per quanto riguarda la quantità di danni e la diffusione.

Love Bug worm (2000), noto soprattutto come **ILOVEYOU worm**.

MyDoom worm (2004)

Storm worm & storm botnet (2007): aveva una frase iniziale contro l'Europa e i server eseguivano in maniera automatica il worm in modo tale che ogni 2 ore si ricodificava, cioè mutava e veniva diffuso nelle macchine vicine.

State of worm technology

La tecnologia di un worm ha delle caratteristiche ben precise:

1. **Multipiattaforma**: cerca di abbracciare più piattaforme possibili
2. **Multi-exploit**: cerca di sfruttare diverse vulnerabilità
3. **Diffusione ultraveloce**: cerca di diffondersi nella maniera più veloce possibile
4. **Polimorfico**: ad ogni iterazione critta il proprio codice (quindi sé stesso) con una chiave di encryption sempre diversa in modo tale che ogni copia del worm sembra diversa. La routine di decryption del worm rimane però la stessa e, a causa di questa parte statica di codice, risulta più facile per un software anti-malware identificare il worm.
5. **Metamorfico**: ad ogni iterazione cambia il codice di cui è composto in uno funzionalmente equivalente in modo tale che per i software anti-malware diventi difficile rilevare la sua firma o il suo pattern. Più a lungo il worm risiede nel sistema, più iterazioni produce e più diventa complicato rilevarlo.

Pertanto, la differenza tra un worm polimorfico e metamorfico sta nel fatto che nel primo caso a istanze diverse il codice eseguito (quello decrittato) è lo stesso, nel secondo caso invece no.

Mobile phone Worms

I primi worms per smartphone hanno avuto come oggetto smartphone con sistema operativo Symbian e si diffondevano via bluetooth o via MMS.

Il più famoso è stato **CommWarrior**, che si replicava:

1. Via Bluetooth con gli smartphone presenti nelle vicinanze
2. Via MMS ai numeri di cellulare presenti nella rubrica dello smartphone infetto e a tutti i cellulari che comunicano con esso.
3. All'interno della scheda SD.

La maggior parte dei malware per smartphone osservati sfruttavano delle app trojan per installarsi negli smartphones.

Mobile Phone Trojans

La presenza di trojan all'interno di uno smartphone è particolarmente rilevante perché quest'ultimo diventa un mezzo di sorveglianza estremamente invasivo: può diffondere contatti, messaggi, la propria posizione e può permettere a un attaccante l'utilizzo in remoto della fotocamera e del microfono.

Alcuni di questi trojan sono stati trovati in app disponibili sul normale mercato. Nel 2011 ad esempio sono state rimosse dall'Android Market delle apps che contenevano il malware DroidDream. Chi le installava, installava

anche il malware facendo diventare lo smartphone uno zombie soggetto al controllo da remoto.

Nel 2012 uno studio riguardante un campione di 1200 malwares trovati in vari Android stores ha rilevato che il 90% di essi compromettevano uno smartphone in modo che venisse aggregato a una botnet.

Gli iPhone al momento sono rimasti un po' più protetti da trojan. Questi ultimi hanno avuto successo solamente per le app distribuite attraverso siti non ufficiali.

Drive-by-downloads

In generale quello che succede è che i malware vengono installati ed eseguiti attraverso un'azione volontaria dell'utente.

Ci sono però diversi esempi di malware che vengono installati solamente perché viene visualizzata una pagina web controllata da un avversario. Con questa modalità di infezione il malware viene installato senza la consapevolezza e il consenso dell'utente (drive-by) proprio perché si sfruttano dei bug del broswer o della piattaforma adoperata affinché l'installazione e l'esecuzione avvengano in background, senza alcuna azione da parte dell'utente.

I drive-by-downloads li si hanno principalmente con la navigazione web perché il gioco è facile per l'attaccante: vengono preparate delle pagine appealing per l'utente (sport, porno, notizie di un settore particolare) e si fa in modo che navigando su tali pagine venga scaricato e installato il malware in modo nascosto.

Un drive-by-download può avvenire anche per lo sfruttamento di vulnerabilità dei visualizzatori PDF in modo che i malware vengano installati guardando semplicemente un documento PDF malevolo.

Clickjacking

Con i meccanismi di clickjacking (cioè rubare i click dell'utente) innanzitutto si fa in modo che l'utente venga rediretto su siti web che contengono del codice malevolo.

Successivamente vengono utilizzate diverse tecniche per spingere l'utente a cliccare su dei pulsanti nascosti senza che se ne accorga:

1. **Utilizzare JavaScript**: un attaccante può inserire un pulsante nascosto al di sotto di un pulsante legittimo. L'utente clicca sul pulsante legittimo per compiere un'azione ma in realtà il suo clic viene reindirizzato a sua insaputa sul pulsante nascosto e ciò causa la redirezione su un'altra pagina o il download di un malware.
2. **Utilizzare dei layer sovrapposti**: questo meccanismo è spesso abbinato a tecniche di phishing. L'attaccante crea una pagina trasparente e la appoggia sulla pagina opaca e legittima. In questo modo l'utente crede di consultare la pagina legittima (ad esempio della propria banca) quando in realtà sta navigando sulla pagina trasparente. Pertanto, il completamento di un campo di una web form risulta in un'intercettazione delle digitazioni dell'utente (MITM sulla tastiera dell'utente). Ciò che l'utente sta digitando viene infine rediretto sulla pagina legittima per far sì che non si accorga di nulla.

Zombie e Botnet

Se un computer viene compromesso sfruttando delle vulnerabilità può essere messo sotto controllo remoto di un altro utente, divenendo così uno **zombie**. Uno zombie può essere incluso in una collezione di computer compromessi, nota come **zombie network** o **botnet**, e venire così utilizzato per altri scopi. Una botnet tipicamente esegue programmi malevoli quali worms, backdoors, Trojan horse e gestita attraverso un'unica infrastruttura di command and control.

Dopodiché, una volta che uno zombie diventa una delle tante macchine che l'avversario controlla, si può anche coordinare un attacco più grosso quale DDoS, di phishing, di spamming o di cracking.

Detailed steps (Creazione botnet)

1. L'attaccante scansiona Internet per rilevare delle macchine poco sicure e che possono essere compromesse. Le mette in una lista.
2. L'attaccante sfrutta delle vulnerabilità nelle macchine listate per installare una backdoor, un tool di command and control o più in generale una qualsiasi modalità che possa rendere comandabile la macchina dall'esterno. Le macchine listate vengono così trasformate in zombies.
3. L'avversario dà delle istruzioni agli zombies per riconoscere un **master server**, il coordinatore delle azioni di attacco.
4. L'attaccante può mettere in atto una campagna di attacco coordinandosi con il master server. Nell'esempio i due si coordinano per un attacco DDoS su un particolare sistema target.
5. Il master server istruisce tutti gli zombies per avviare l'attacco DDoS a un determinato orario nei confronti del sistema target.
6. Il sistema target è inondato di richieste da parte degli zombies e, ovviamente, le richieste degli utenti legittimi saranno negate. L'attacco DDoS ha così effetto.

Storm botnet

Il primo esempio risale alla **botnet Storm** nel 2007. C'erano diversi modi con cui l'infezione poteva propagarsi:

1. Allegati email
2. Download di un programma per mostrare un video
3. Drive-by downloads

Una volta che una macchina era infettata, un tool permetteva il controllo da remoto della macchina e il coordinamento con gli altri zombies della botnet.

Botnet

In generale si utilizzano per:

1. Attacchi DDoS (Distributed Denial of Service)
2. Spamming: cioè fare dello spam
3. Sniffare il traffico
4. Fare keylogging
5. Fare da agenti di diffusione per nuovi malware
6. Installare dei componenti aggiuntivi pubblicitari e BHO (Browser Helper Object, programma installato da un altro software e che parte in automatico quando si accede al browser)
7. Attaccare chat IRC
8. Manipolare giochi e sondaggi online

Rootkit

Un rootkit è un insieme di programmi per avere accesso da root a una macchina in maniera nascosta.

Ci sono diversi modi di agire ma l'effetto è sempre quello che l'avversario ha la possibilità di penetrare una macchina per poi controllarla da remoto. Le due tipologie di rootkit per ottenere questa funzionalità sono:

1. **Rootkit semplici**: agiscono a livello di programmi utente. Modificano i servizi di base del sistema operativo (ad esempio i comandi Unix *ls* e *ps*) senza che l'utente se ne accorga per installare i meccanismi di controllo remoto. Rilevabili da tool come Tripwire.
2. **Rootkit sofisticati**: modificano il kernel del sistema operativo e sono difficili da rilevare dallo spazio utente.

Rootkit classification

I rootkit possono essere anche classificati come:

1. **Application-level rootkit**: programmi malevoli a livello utente;

2. **Traditional rootkit**: programmi che cambiano la funzionalità dei programmi di sistema (per esempio per avere dei trojan);
3. **Kernel-level rootkit**: programmi che lavorano con i pivilegi più elevati nel SO, avendo così la possibilità di modificare il kernel;
4. **Under-kernel rootkit**: programmi che hanno un effetto negli ambienti virtualizzati. Si ha una macchina che sembra funzionare bene ma in realtà è sotto il controllo di una macchina virtuale malevola. (?) Tutto quello che succede all'interno della macchina è sempre sotto il controllo dell'avversario poiché ha il controllo dell'hypervisor.

Tradizionalmente per rootkit ci si riferisce alla classificazione 2.

Rootkit classification

La classificazione dei rootkit si basa su diverse proprietà:

1. **Persistent**: si attiva ogni volta che il sistema fa il boot. Se non è persistente significa che una volta che il sistema viene rebootato il rootkit viene cancellato.
2. **Memory based**: il codice del rootkit risiede completamente in RAM. In tal caso non sopravvive ai reboot.
3. **User mode**: se intercetta chiamate alle API di sistema. Viene eseguito con permessi di root e risiede nello spazio utente.
4. **Kernel mode**: intercetta chiamate alle API native in kernel mode.
5. **Virtual machine based**: il rootkit può intervenire direttamente nell'ambiente virtualizzato.
6. **External mode**: è al di fuori del normale funzionamento del sistema target. Ha direttamente accesso all'hardware quindi è ancora più difficile da rilevare, come nella classificazione 4.

Spyware

Malware che colleziona un po' di informazioni alla volta sugli utenti a loro insaputa. Tale software malevolo può fare ad esempio del keylogging. Tipicamente non si auto-propaga.

Scareware

Software la cui installazione viene suggerita mediante metodi di social engineering provocando shock, ansia o la percezione di una minaccia all'utente che utilizza la macchina sul quale lo scareware viene eseguito. Ha un'utilità limitata o nulla per l'utente.

Ransomware

Un'altra tipologia che è stata oggetto di diverse campagne e di attenzione da parte della cronaca giornalistica. Sono dei software che rendono inaccessibili le informazioni personali dell'utente presenti nella macchina vittima mediante cifrature molto robuste (es: RSA4096). Per accedere nuovamente ai dati in chiaro viene richiesto un riscatto da pagare in forma anonima utilizzando delle criptovalute.

Gli attaccanti dietro ai ransowmware utilizzano un meccanismo di fiducia per tutti i riscatti: affinché questo sistema sia credibile, pagando il riscatto restituiscono con abbastanza serietà la chiave di cifratura e, quindi, l'accesso ai contenuti.

Esempio reale: un ospedale inglese ha pagato tra le 50mila e le 80mila sterline come riscatto perché, facendo un'analisi costo-benefici, pagare il riscatto e riavere il database ospedaliero costava meno che ricominciare da zero.

Le campagne di ransomware sono all'ordine del giorno e sono guidate, nel senso che prendono di mira ospedali e aziende pubbliche dove il livello di sicurezza non è altissimo e dove si ha un'alta probabilità di essere pagati.

Kaspersky report

Rapporto di due-tre anni fa (in sostanza sono dei vecchi dati!) contenente dei dati un po' gonfiati. Kaspersky infatti è una delle grosse multinazionali che si occupano di sicurezza e mostra che il business importante per loro c'è.

Attacks e Black energy trojan

Gli attacchi informatici si rivestono sempre più di connotati politici. Il BlackEnergy cyber attack è un attacco che ha a che fare con il settore della gestione delle reti energetiche in Ucraina. Ha avuto anche impatto nel mondo fisico: l'oggetto dell'attacco era infatti la disabilitazione dell'energia elettrica nell'ovest dell'Ucraina.

2014 - Attacchi in tutto il mondo
L'attacco in Ucraina era basato su un vecchio codice (BlackEnergy) sviluppato nel 2007. Già nel 2014 un gruppo di utenti del codice di BlackEnergy hanno effettuato un attacco direttamente ai dispositivi fisici che adottano il protocollo SCADA, utilizzato per gestire diversi tipi di dispositivi tra cui quelli che riguardano l'erogazione di energia.

2015 - Attacco in Ucraina
Per l'attacco svolto in Ucraina nel 2015, un vettore di attacco di BlackEnergy erano i documenti Excel con delle macro che depositavano un trojan su disco se l'utente decideva di eseguire lo script contenuto nel documento. La macchina infetta si metteva quindi in comunicazione con il server C&C sulla porta 80 e l'attaccante poteva così impartire dei comandi da remoto. Interagendo con i dispositivi che utilizzavano il protocollo SCADA poteva quindi mettere in crisi la distribuzione dell'energia elettrica. Poco prima della disabilitazione dell'erogazione di energia elettrica fu lanciato un attacco DDoS ai call center per impedire ai clienti di chiamare per segnalare l'interruzione.

Vulnerable applications used by cybercriminals

In alcuni di questi rapporti è possibile vedere quali sono i maggiori software e piattaforme target dei potenziali nemici. Navigare sul web è una delle attività più diffuse, per questo motivo la maggior parte degli attacchi sfruttano del codice malevolo eseguito all'interno dei browser o dei plugin dei browser.

Zero-day vulnerabilites

Sono quelle più appetibili e più remunerate nel mercato nero dei malware perché nessuno le ha patchate, cioè nessuno ha posto rimedio. Zero-day vulnerability significa dare in mano le chiavi ad un potenziale attaccante.

Un'attaccante esperto deve avere un minimo di conoscenza tecnica per usarle nella maniera più opportuna. Ci può essere anche qualcun altro che magari ha già predisposto una piattaforma, un qualche rootkit o un tool che si appoggia ad una zero-day vulnerability.

La figura mostra un riassunto degli ultimi 13 anni di vulnerabilità: nell'ordine sono state trovate più zero-day vulnerabilities in Other (altri), Windows, Adobe Flash Player, Explorer e così via.

Spear phishing

Con lo **spam** tipicamente si lanciano milioni di email, anche artefatte in maniera grossolana e facilmente rilevabili dall'utente minimamente attento. Nelle mail di spam il tempo di sviluppo è particolarmente ridotto:

1. Online si trovano database free di milioni di indirizzi email. Al limite si può spendere qualche dollaro per avere a disposizione delle rubriche con indirizzi attivi e che sono verificati.
2. Si utilizza Google Translate per tradurre il testo spam in 30-40 lingue
3. Si sviluppa un piccolo engine in Python dove, a seconda della desinenza dell'indirizzo mail, lo si traduce in una certa lingua e lo si spedisce.

Il risultato che si ottiene è che il destinatario nota che la lingua non è proprio quella giusta poiché c'è qualche termine che non va.

Lo **spear phishing** è un'evoluzione di questo attacco dove entrano in gioco anche nozioni derivanti dal social engineering. Le mail sono molto indirizzate alla singola persona o azienda dove lavora l'impiegato vittima, con il risultato di essere realizzate parecchio bene e difficilmente distinguibili da generiche mail di spam.

Ad esempio può capitare che non prendano Pinco Pallino che scrive da uno stato sconosciuto come in una mail di spam ma, al contrario, facendo uno studio sui diversi social come mittente possono prendere un contatto di Facebook o di Instagram della vittima.

La mail sono anche contestualizzate: se il destinatario ad esempio si occupa di scrivere per lavoro degli articoli legati alla sicurezza, una mail plausibile ma di spear phishing potrebbe contenere:

1. Come campo mittente: "Springer.com", infatti tale campo può essere facilmente modificabile a piacimento dall'attaccante;
2. Come body della mail: un coautore della vittima che afferma "C'è questa opportunità su questa rivista, se ti interessa clicca qui!"

L'utente esperto dovrebbe consultare il messaggio originale e analizzarne l'intestazione osservando se i passaggi che la mail ha fatto sono sensati: se la mail proviene da Springer.com determina che in effetti sono stati utilizzati i server della Springer per recapitare la mail al server di unimi. Al contrario, se la mail ha attraversato qualche hop con un dominio strano (ad esempio springerfake.com) si può concludere che la mail è di spear phishing.

In aziende con utenti meno esperti come gli ospedali e le scuole queste tipologie di mail non fanno fatica ad avere successo. Dal grafico 2002-2015 si può notare che tali campagne di spear phishing sono molto in crescita perché danno buoni frutti.

Advanced Persistent Threats

Le APT sono minacce avanzate e persistenti che provengono da gruppi ben specifici di persone che conducono cyberwars, anche a livello globale, con diversi risvolti politici.

Definizione del NIST: avversari che possiedono livelli sofisticati di esperienza e che investono molte risorse per raggiungere i propri obiettivi utilizzando diversi vettori di attacco.

1. **Advanced**: mettono insieme diverse tecniche di attacco avanzate e recenti.
2. **Persistent**: cercano di attaccare un sistema con delle campagne mirate che possono durare anche abbastanza a lungo;
3. **Threats**: sono organizzate perché dietro non c'è il singolo hacker che si diverte ma una vera e propria organizzazione.

Gli **obiettivi** di queste APT riguardano:

1. Segreti finanziari
2. Proprietà intellettuale: ad esempio progetti industriali.
3. Segreti nazionali
4. Infrastrutture: ad esempio piani di sviluppo nazionale, informazioni sensibili che riguardano il settore energetico o nucleare.
5. Informazioni personali: ad esempio di qualche esponente pubblico o di un manager di un certo livello che può essere utile da attaccare.

Come **target** non hanno la singola persona ma dei campi ben precisi:

1. Aerospaziale
2. Gestione dell'energia
3. Industria chimica
4. Telecomunicazioni: aziende telefoniche, ISP
5. Salute: ad esempio il segreto industriale riguardante alcuni medicinali.
6. Governi
7. Istruzione

Intrusion kill chain

Queste APT sono state categorizzate perché abbiamo una certa scansione delle fasi di attacco:

1. Ricognizione: cerco di capire qual è il target e quali sono le persone coinvolte e cerco di ricostruire le relazioni sociali di queste persone. Effettuo quindi uno studio sistematico di quali possono essere i singoli obiettivi.
2. Metto insieme le mie armi: trojan, diversi tipi di vulnerabilità che posso utilizzare
3. Delivery: individuo quali possono essere i mezzi per depositare queste vulnerabilità all'interno del perimetro di difesa della vittima: attachments di messaggi email, siti web e chiavette USB.
4. Sfrutto la possibilità di depositare e installare il codice. A seconda della tipologia di codice o di minaccia che sto portando avanti avrò a disposizione un trojan o un kit di command and control attraverso il quale prendo possesso della macchina vittima e poi … (?)
5. Actions on Objectives: azioni malevole come (?) normale funzionamento di un sito web fino al reperimento di informazioni personali
6. Conceal activity: cerco anche di nascondermi

La mappa del sito Fireeye mostra gli attacchi in tempo reale.

Il secondo link fornisce una visualizzazione diversa delle minacce, mostrando da quanto sono attive le varie campagne di attacco in corso.

APT samples

Ci sono un po' di esempi di APT storiche. Quella più conosciuta forse è Stuxnet: è una campagna che avuto come oggetto gli impianti nucleari dell'Iran. (?)

L'obiettivo principale era sabotare il programma nucleare iraniano, rallentando il programma di sviluppo. (?)