# Remote injection

L'**iniezione remota** è una iniezione che avviene mediante un vettore di attacco remoto. Non si ha a disposizione una shell sulla macchina della vittima per l'immissione diretta dei comandi.

Sono presenti due asset

* **client**: invia richieste: i dati delle richieste contengono iniezioni per uno specifico linguaggio (SHELL, SQL...)
* **server**: riceve richieste, elabora risposte, invia risposte

Nebula07

“The flag07 user was writing their very first perl program that allowed them to ping hosts to see if they were reachable from the web server.”

*Obiettivo*: eseguire il comando **/bin/getflag** con i privilegi dell'utente **flag07**

Alcune considerazioni:

* Non si tratta di un banale attacco "login diretto";
* Non si considerano le iniezioni di variabili di ambiente;
* Ci si concentra sull'iniezione diretta di comandi

Nella directory **/home/level07** non sembrano esserci file interessanti.

In **/home/flag07** invece sono presenti file di configurazione di BASH e altri due file molto interessanti:

* 12844 -rwxr-xr-x 1 root root 368 2011-11-20 21:22 **index.cgi**:
  + non è file SETUID, contiene lo script del eseguito dal server Web
* 12846 -rw-r--r-- 1 root root 3719 2011-11-20 21:22 **thttpd.conf**
  + è un file di configurazione leggibile da tutti gli utenti e modificabile solo da **root**

Lo script **index.cgi**:

* Riceve in input:
  + da una richiesta *GET /index.cgi?Host=IP*
  + ouppure da un argomento *Host=IP* (se invocato tramite linea di comando)
* Crea uno scheletro di pagina **HTML**
* Esegue il comando **ping –c 3 IP 2>&1**
* Inserisce l'output del comando nella pagina HTML.

*Proviamo*

Il modulo Perl CGI permette l'esecuzione dello script in locale. Si può provare una iniezione locale.

Ci si autentichi come utente **level07** e si digiti il comando:

* **/home/flag07/index.cgi "Host=8.8.8.8; /bin/getflag**

Non sembra che venga eseguito il comando /bin/getflag

*Perchè*?

Cerchiamo di capire come funziona nel dettaglio la funzione **param()** di PERL.

Il carattere ' **;** ' assume un ruolo speciale nel contesto degli URL gestiti dallo standard CGI. Tramite il carattere ; si possono separare parametri (come avviene con il carattere &).

Quindi, invocando il comando: **/home/flag07/index.cgi "Host=8.8.8.8; /bin/getflag"** stiamo in realtà passando due parametri:

* *Nome*=Host *valore*=”8.8.8.8”
* *Nome*=/bin/getflag *valore*=””

Solo che lo script estrae solo il primo valore di Host (**$\_[0]**).

1. Possiamo fare escape del carattere ';'?
2. Esistono altri caratteri speciali meritevoli della stessa attenzione?

Procedura di escaping dei caratteri speciali in un URL

Dato il carattere speciale:

1. Si individua il suo codice ASCII
2. Si scrive il suo codice ASCII in esadecimale
3. Si prepende il carattere di escape **%**

*Esempio*:

URL encoding del carattere **;**

1. Codice ASCII in base10: 59
2. Codice ASCII in base16: 3B
3. Codifica URL-encoded: **%3B**

URL encoding del carattere **/**

1. Codice ASCII in base10: 47
2. Codice ASCII in base16: 2F
3. Codifica URL-encoded: **%2F**

*Soluzione*

L'URL corretto da inviare allo script index.cgi prevede l'URL encoding dei caratteri speciali (tutti!):

* **Host=8.8.8.8%3B%2Fbin%2fgetflag**

**L'iniezione ha funzionato, ma index.cgi non ha i privilegi di esecuzione di flag07**

**È necessario eseguire lo script con i privilegi dell'utente flag07**

Bisogna identificare un server Web che esegua **index.cgi** SETUID **flag07,** se questo server esiste, l'input appena usato permette l'esecuzione di **/bin/getflag** con i diritti di **flag07.**

Analisi del file di configurazione **thppd.conf**:

* **port=7007**: il server Web thttpd ascolta sulla porta TCP 7007 (si può contattare il serve Web sulla porta 7007)
* **dir=/home/flag07**: la directory radice del Web server (il Web server vede l'intero file system quindi anche il file eseguibile **/bin/getflag**)
* **nochroot**: il Web server "vede" l'intero file system dell'host
* **user=flag07**: il Web server esegue con i diritti dell'utente **flag07 (**il Web server esegue come utente flag07**).**

Per poter effettuare l'iniezione remota, occorre verificare se il Web server **thttpd** è in esecuzione sulla porta 7007**:**

* **pgrep –l thttp**
* **netstat -ntl | grep 7007**

*C'è un processo in ascolto sulla porta TCP 7007!!*

1. Spostiamoci nella home directory **/home/level07**
2. Facciamo una richiesta al web server:
   * **wget** [**http://127.0.0.1:7007/index.cgi?Host=%3Bgetflag**](http://127.0.0.1:7007/index.cgi?Host=%3Bgetflag)
3. Nella home directory apparirà un file **index.cgi?Host=;getflag**

**You have successfully executed getflag on a target account**

# **Sql injection**

Si parla di **SQL Injection** quando un attaccante è in grado di inserire in input, una opportuna stringa o una concatenazione di comandi in una query SQL.

*Come fare*

Inviare al server una richiesta legittima, valida, non maliziosa per analizzarne la risposta.

L'invio di richieste "anomale", effettuato con l'obiettivo di scoprire malfunzionamenti nel programma, prende il nome di **fuzz testing** (o **fuzzing**). La costruzione di un attacco è sempre preceduta da una procedura di fuzz testing che individua il punto esatto dell'iniezione.

Tecniche di SQL Injection base

* Aggiungere un carattere di commento che "taglia" il resto della query (es., **--**, **#**, ...)
* Aggiungere una tautologia che rende la query sempre vera, cioè tutte le righe della tabella soddisfano la query (es., **OR 1=1**, **OR ' ' = ' '** , …)
* Aggiungere un comando (es., ; **DROP TABLE users**)

Tecniche di SQL Injection avanzate

Generalmente, attacchi multi-step che sfruttano altre keyword di SQL.

[https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection](https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL Injection)