(Too much) Access Points – Exploitation Roundup

CONFidence 2010

Cristofaro Mune

Take Off

Me

- **Cristofaro Mune**
  - Independent Security Researcher
  - Preferably focused on *Mobile and Embedded Security*

- In the Past
  - Security Research Lead @ Mobile Security Lab (www.mseclab.com)
  - Various consulting works on Mobile & IT security

- Previous works
  - *Mune, Gassirà, Piccirillo* - **"Hijacking Mobile Data Connections"** - BlackHat Europe '09
  - *Mune, Gassirà, Piccirillo* – **"Hijacking Mobile Data Connections 2.0: Automated and Improved"** - Deepsec 2009

Talk Goals

- ***Demonstrate*** ***arbitrary code execution*** on Access Points from multiple Vendors
  - Platform: Linux/MIPS

- ***Demonstrate*** ***a blind remote*** attack scenario:
  - Exploitation achieved by “reflection” by “Browser-in-the-Middle”

- ***Release*** many previously undisclosed vulnerabilities
  - Hoping to stimulate Vendor response and, hopefully, have them ***FINALLY*** fixed

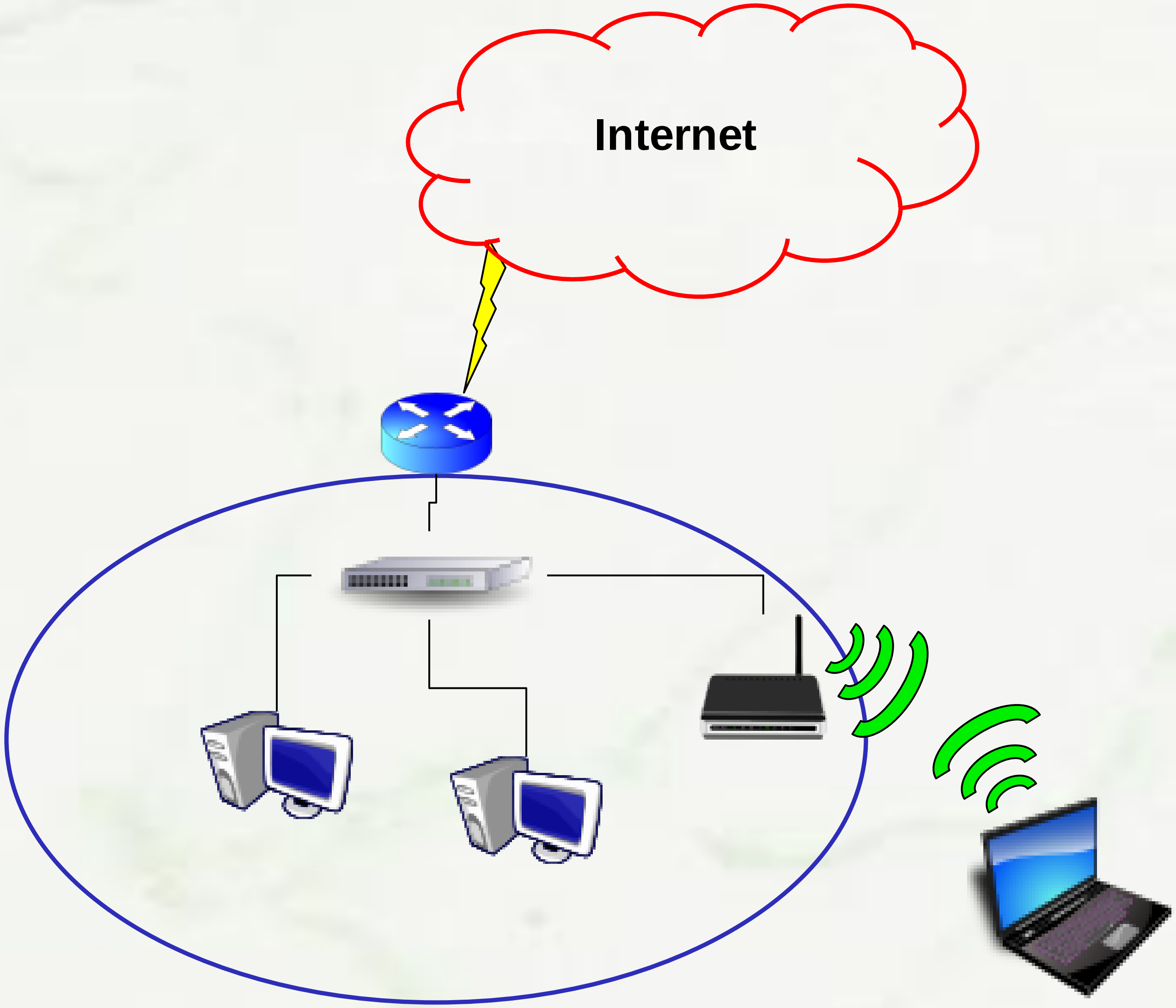
Recognition

Embedded networking devices

- *RISC processors:*
  - MIPS/ARM (both little and big endian)
  - Lower consumption
- *Low resources:*
  - RAM: Typically 4/64 Mbytes
  - Flash: 2/16 Mbytes
- Several Open source distributions
  - eg: DD-WRT, OpenWRT,...
- **Linux/MIPS** quite common pair

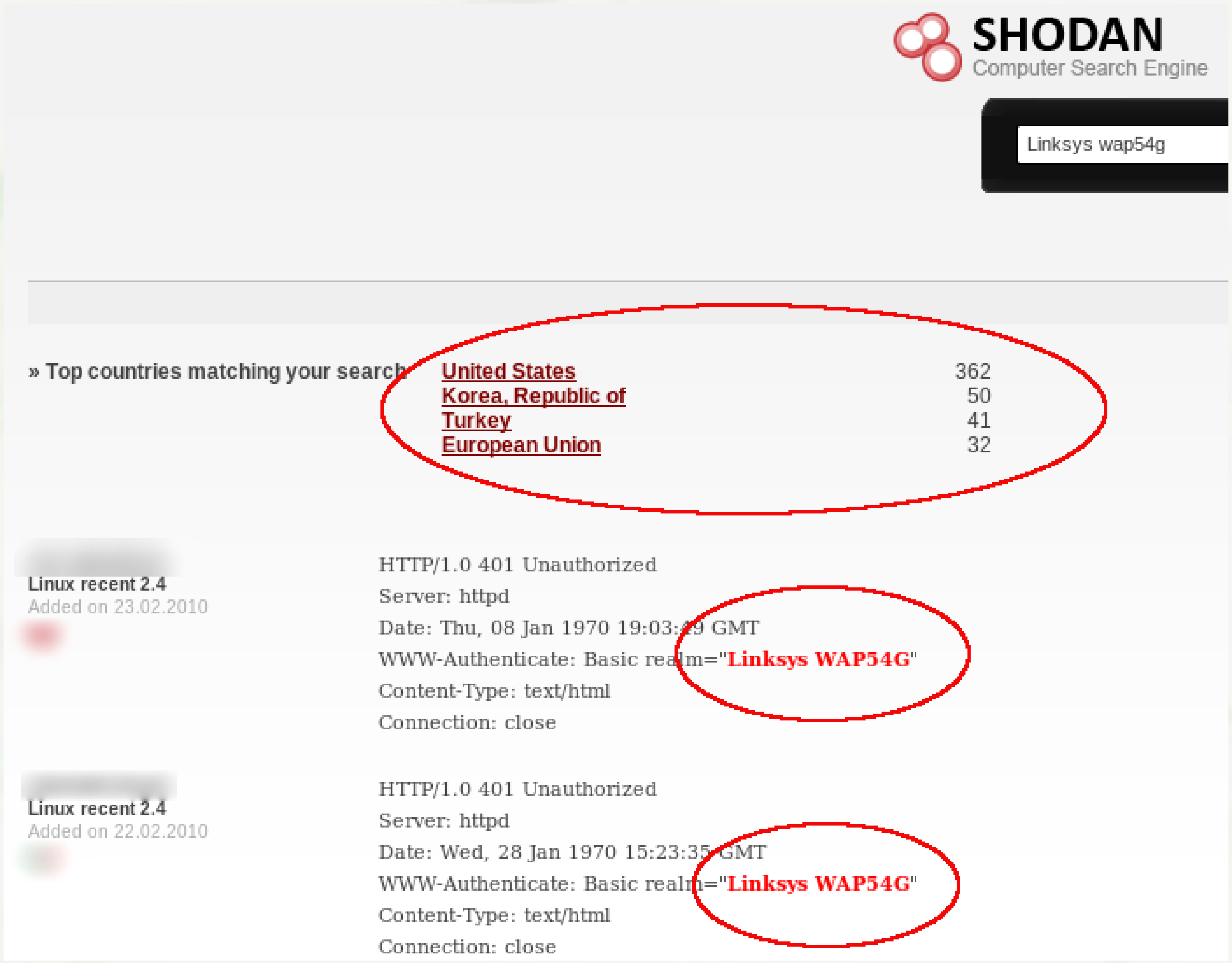
Access Points

- Even simpler Hardware
- Stripped down software
- Usually located in LAN
  - *Private IP addressing*



Not directly reachable from the Internet…

Really...!?



SHODAN
Computer Search Engine

Linksys wap54g

» Top countries matching your search

United States	362
Korea, Republic of	50
Turkey	41
European Union	32

[redacted]
Linux recent 2.4
Added on 23.02.2010

```
HTTP/1.0 401 Unauthorized
Server: httpd
Date: Thu, 08 Jan 1970 19:03:49 GMT
WWW-Authenticate: Basic realm="Linksys WAP54G"
Content-Type: text/html
Connection: close
```

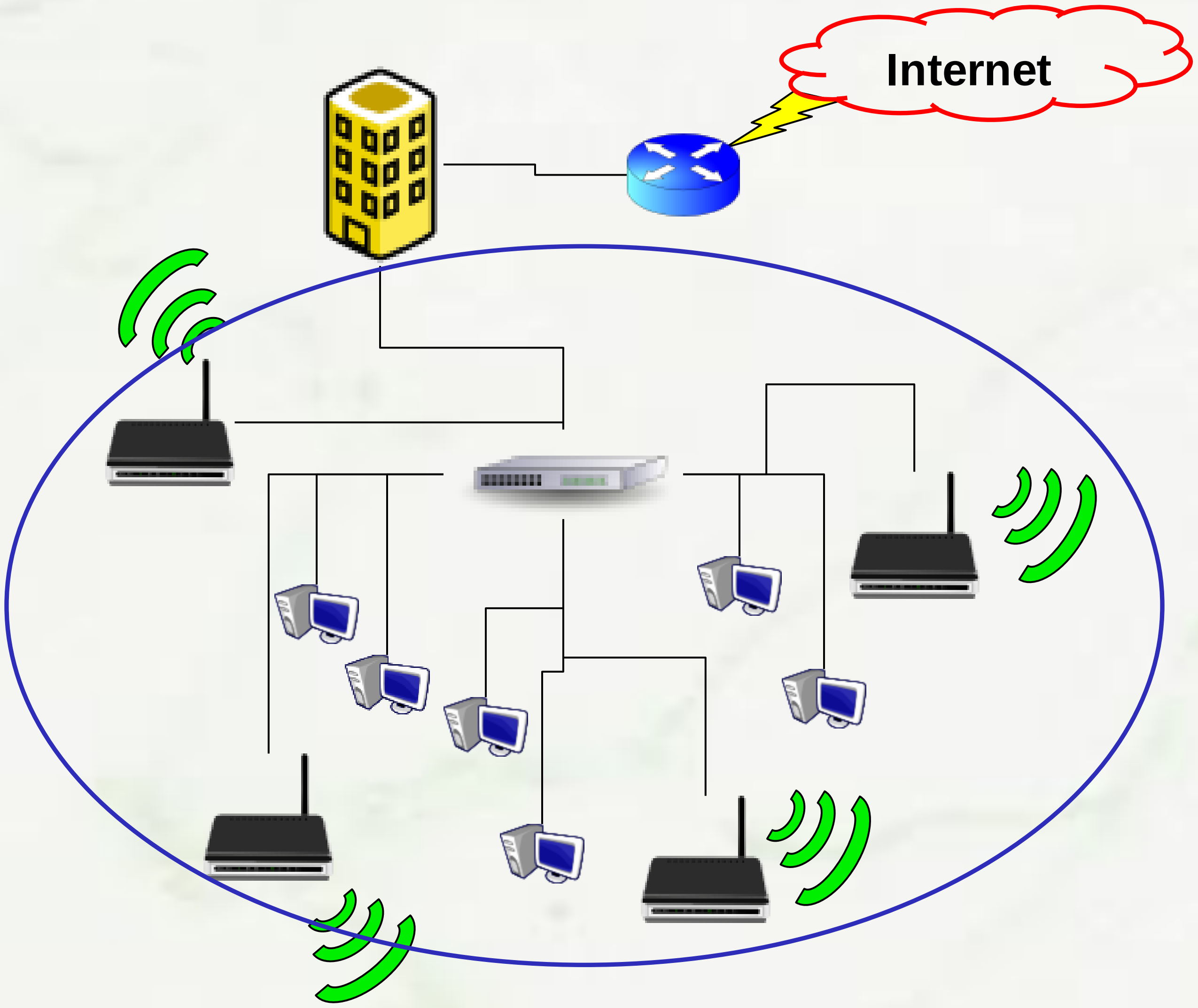
[redacted]
Linux recent 2.4
Added on 22.02.2010

```
HTTP/1.0 401 Unauthorized
Server: httpd
Date: Wed, 28 Jan 1970 15:23:35 GMT
WWW-Authenticate: Basic realm="Linksys WAP54G"
Content-Type: text/html
Connection: close
```

…in the Enterprise

- *Larger number of devices*

- ***Monocultures***



Typical …

- ***Attack avenues:***
  - Weak admin credentials
  - Web interface vulnerabilities
    - Auth bypass, Command injection, XSS, XSRF,…
  - UPNP
  - Wireless related attacks

- ***…and goals:***
  - Access/enable remote management:
    - Web interface or network services (FTP, SSH, Telnet, SNMP)
  - DNS manipulation
  - Wireless passphrases extraction
  - Modified firmware upload

AP or Linux/MIPS specific works

- ***Papers:***
  - Laurent Butti - *"Wi-Fi Advanced Fuzzing"* – BlackHat Europe 2007
  - Julien Tinnes – *"Linux MIPS ELF reverse engineering tips"*
  - *...more in Reference section*

- ***Binary exploits:***
  - ***???***
  - ***Be patient ☺***

- ***Shellcoding:***
  - Linux/MIPS LE port bind shellcode – 276 bytes
  - Linux/MIPS LE execve shellcode – 60 bytes
  - Joshua Drake – "shell_reverse_tcp" (BE and LE) – Metasploit payload
  - Julien Tinnes – "MIPSLE XOR Encoder" – Metasploit encoder

AP exploitation advantages

- ***Stealthiness:***
  - Poor management/monitoring
  - Interesting “hiding place”
- ***Full access to remote wireless networks***
  - Remote extraction of Hidden SSIDs, Keys
  - At the choke point of wireless networks traffic
- ***Foothold/jumppad in the Internal network***
  - Do you protect ***FROM*** your AP?
- ***Enterprises***
  - **Monocultures** →
    - *“One vuln to rule them all..”*
    - ***“Infective”*** 0wnage (worm-like exploitation)
- ***Botnets***

Refining target

- **Stock firmwares** most interesting target for attacker

- Which **entry point?:**
  - *Wi-Fi:*
    - *Pro:* Wi-Fi drivers vuln may lead to kernel level exploitation
    - *Con:* Requires being in the range of the wireless signal
    - *Con:* Auth required for accessing IP stack and services
  - *Ethernet:*
    - *Pro:* Does not require target proximity.
    - *Pro:* IP stack and network services directly accessible
    - *Pro:* any vuln may be present on wireless “side” also (possibly after auth)
    - *Con:* Private IP addressing may not allow direct IP reachability

Setting goals…

- ***Primary*:**
  - Execution of **arbitrary code** on APs loaded with ***stock firmware***
  - Exploitation **shall not require target proximity**

- ***Secondary*:**
  - Exploitation **should not depend upon authentication**
  - Exploitation should be possible for **not “directly IP-reachable”** targets

Can this be done?

At which extent??

Aiming: Choosing Weapon

By distance…

Symbols

- **Local attacks**
  - Physical interaction required (eg: FW modifications)
- “Range” attacks
  - Proximity required (eg: WiFi)
- **Remote attacks**
  - Target IP address ***MUST*** be reachable
    - Public address or…
    - Attacker located in Internal Lan
- **Remote blind attacks**
  - Target IP ***MAY*** be also not reachable
  - Leverage a 3rd party, that actually performs the attack
  - ***Possible if vulnerability allows “reflection”***

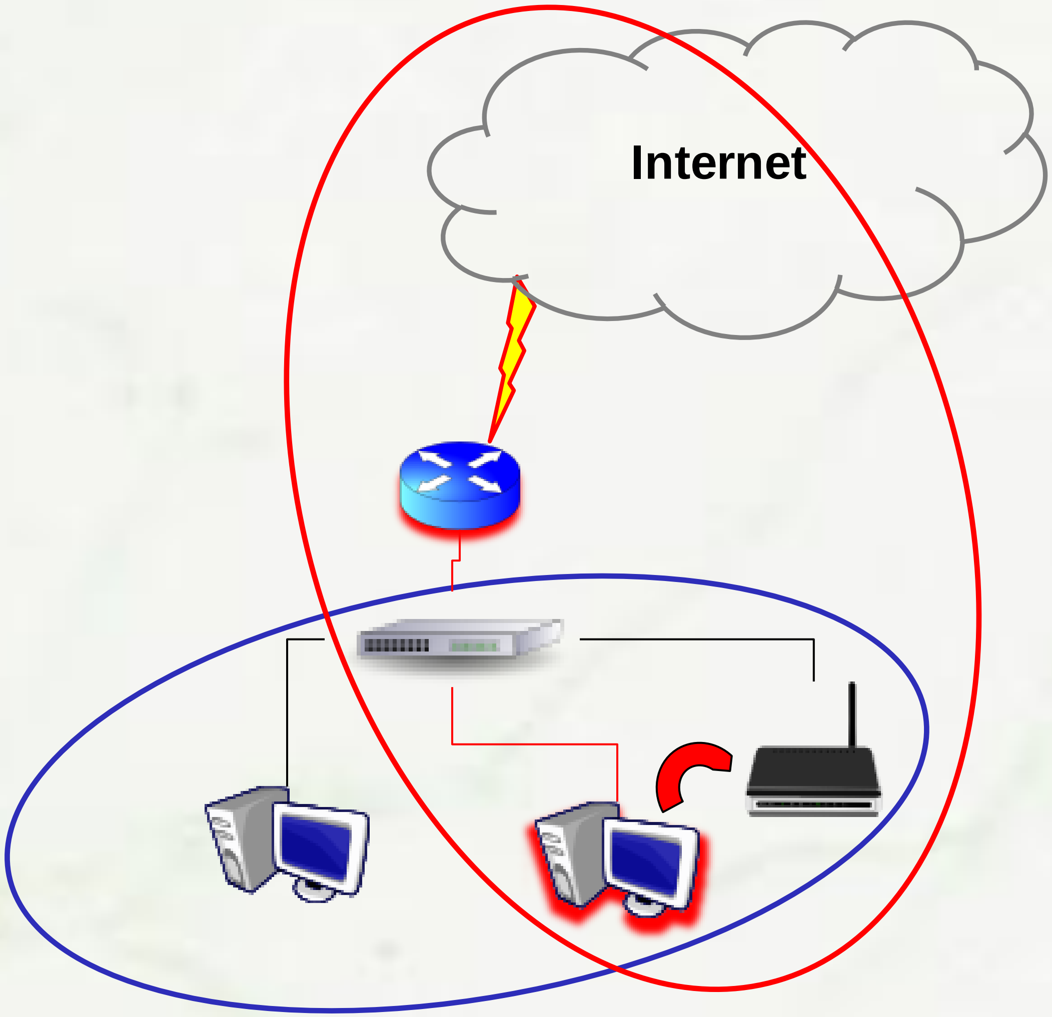
Not all vulns are created equal…

- *Generic UDP daemon vulnerability*
  - Cannot be easily reflected

Remote attack only

- *Web server request URL length buffer overflow*
  - "Reflected" attack is possible
    - eg: via <*img src*> tag

Remote Blind attack

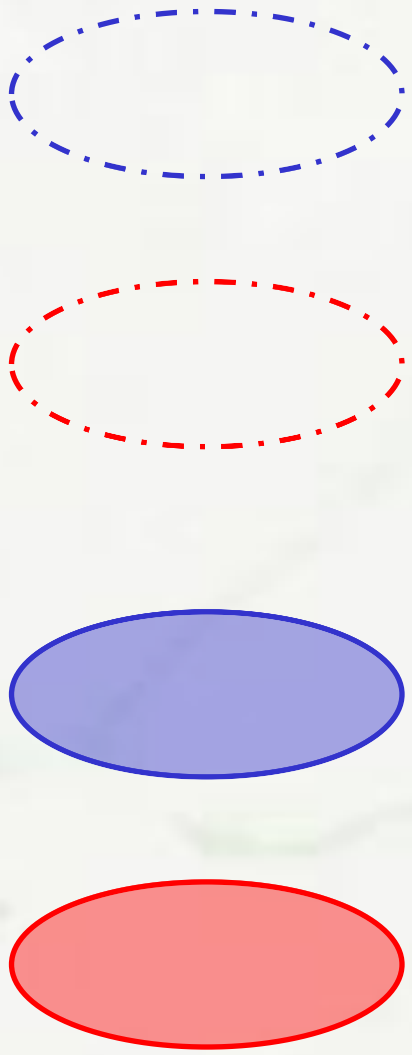


Choosing weapon: by impact

Symbols

- **Authentication needed *(POST-AUTH)***
  - Authentication required for the vulnerable resource
  - Vulnerable code path accessible only *AFTER* auth

- **Authentication not needed *(NO-AUTH)***
  - PRE-Auth
  - Auth Bypass



Aiming: Challenges

Challenges: Vulns Research

- ***Source code***
  - Not generally available
  - Version mismatches

OR...

- ***Firmware image***
  - May not be available for download
  - Version mismatches

OR...

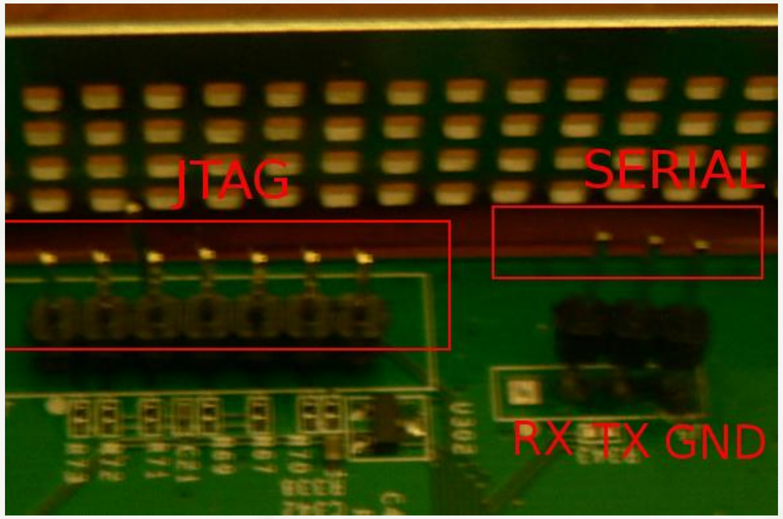
- ***Firmware dump***
  - May be possible with:
    - Serial/JTAG interface
    - Hardware flash dump

Challenges: Exploit development

- **Communications**
  - Serial console (if any)

- **Build your own *WORKING* firmware image**
  - May be needed for uploading tools
  - JTAG may be helpful for recovery from bricking

- **Few resources available for exploitation**
  - eg: just a couple of shellcodes available
  - ***Write your own shellcodes!***



Netgear WG602v4 pinout

Challenges: Exploit development/2

- Debugging or.. ***"How do you look at registers?"***
  - Debugging tools not available
    - Cross compiling needed
    - Little Flash space: ***write your own "nano-scaled" tools***
  - Instruction pointer not accessible
    - How do you know where your exploit failed?
  - Stripped down environment
    - Needed libraries may be not available
    - Very minimal shell may be present on the target
- **Cache incoherency**
  - Separate caches may bring very erratic behavior
    - **Affects exploit reliability**
    - Issue not present on x86 exploitation

Firing

Targets

Netgear WG602v4

D-Link DAP-1160

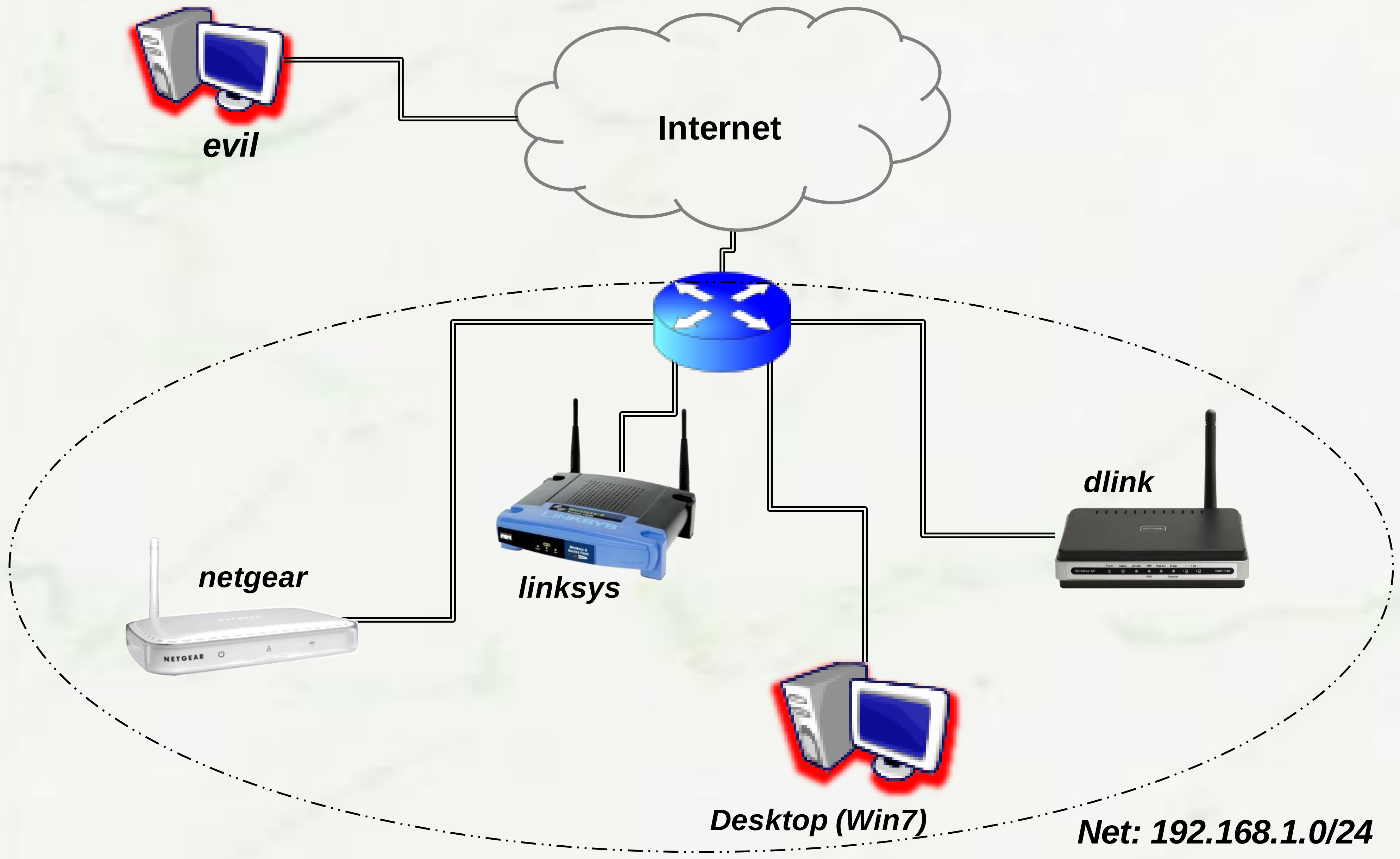
Linksys WAP54gv3

Goal:

Gain a Connect-back

TCP root shell on each!

Demo setup



Net: 192.168.1.0/24

MIPS (very) basic notes

- ***Registers & Instruction set***
  - 32 *general purpose* registers
    - Instruction pointer not accessible
  - 32 bits instruction set
    - Instruction and data alignment required
    - No instructions for explicit stack manipulation

- ***Calling convention (o32)***
  - **Args passed via registers *($a0-$a3)***
    - stack used after 4th arg
  - **Return address saved in register *$ra*** at call (*jal/jalr $t9*)
    - But.. ***also saved on the stack*** *in prologue*
    - Return performed via *jr $ra* ***(retrieved from stack)***
  - Return value in *$v0*

Netgear WG602v4

Overview

- ***CPU*:** MIPS @ 240 Mhz (Broadcom SoC BCM5354)
- ***Byte "sex":*** Little-endian

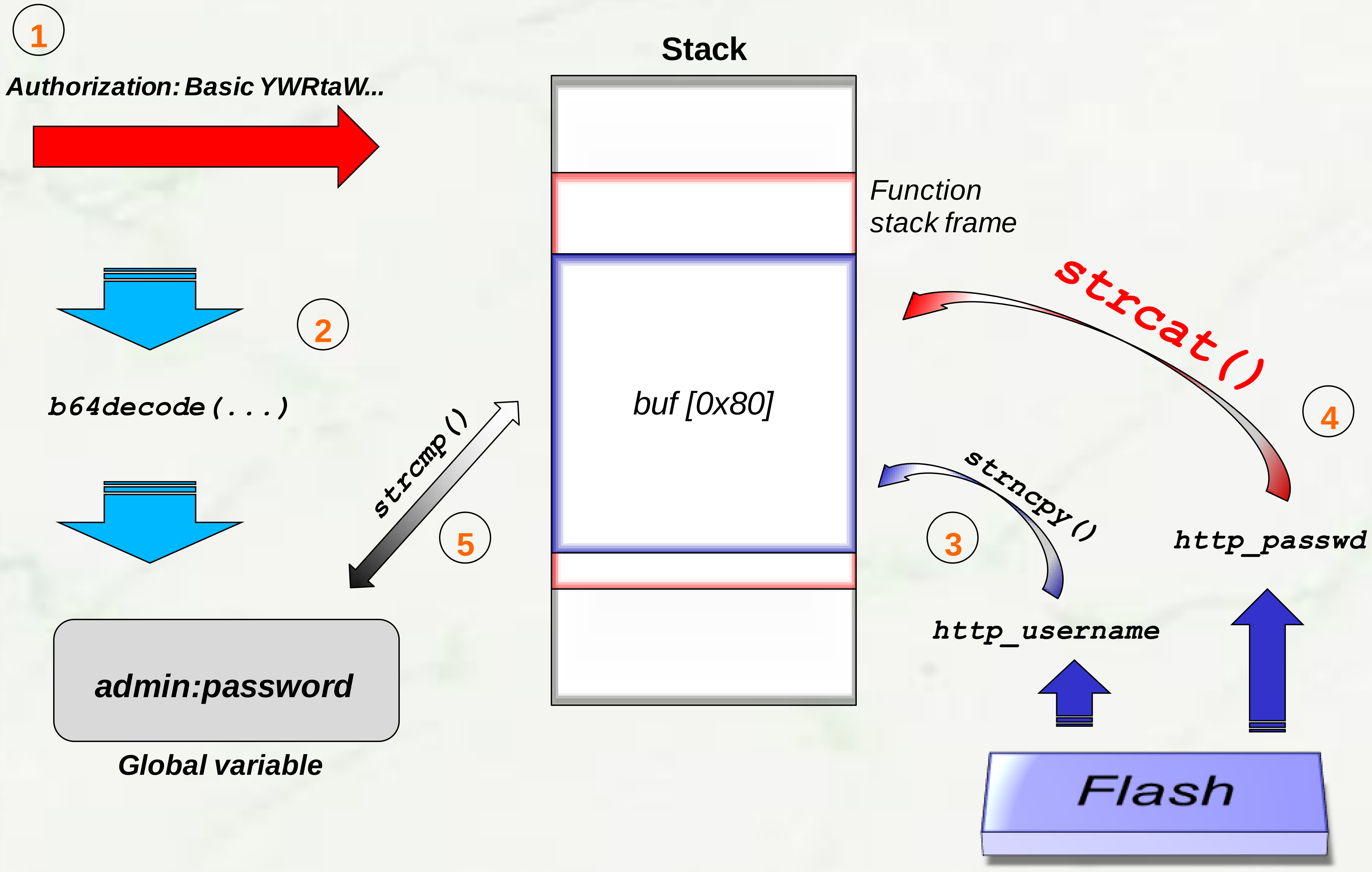
- ***Memory***
  - 8Mbytes RAM
  - 2Mbytes Flash

- ***OS:*** Linux 2.4.20
- ***Web Server:*** Boa/0.94.11

- ***Firmware analysis***
  - Version: 1.1.0
  - Source code available: Yes
  - Firmware image available: No
  - Dumped firmware: Yes

Defaults:
IP: **192.168.0.227**
User: **admin**
Password: **password**

Auth overview



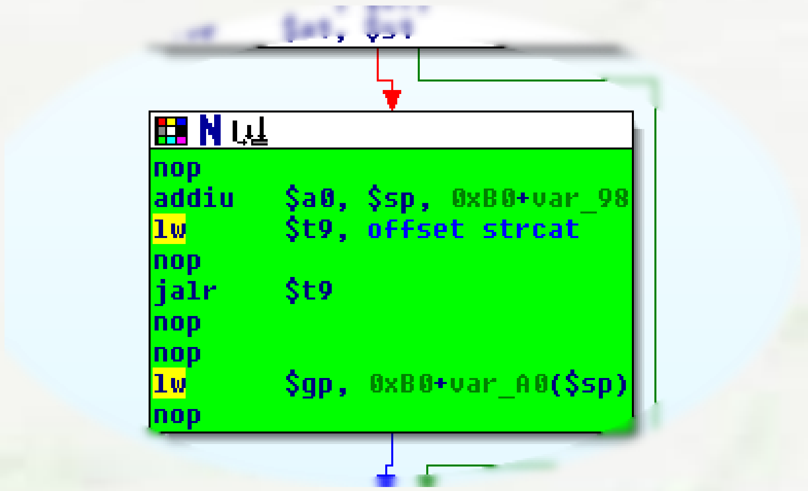
Vuln 1.1: “Saved password Stack Overflow”

- Authentication handled by *auth_authorize()* in auth.c
  - **NOT PRESENT in Boa 0.94.11 original source code**

- Password stored in flash copied in fixed size buffer on the stack

- No lenght check ⇨ ***Buffer overflow***

⇨ Saved *$ra* overwrite ⇨ ***Code execution***



```
nop
addiu   $a0, $sp, 0xB0+var_98
lw      $t9, offset strcat
nop
jalr    $t9
nop
nop
lw      $gp, 0xB0+var_A0($sp)
nop
```

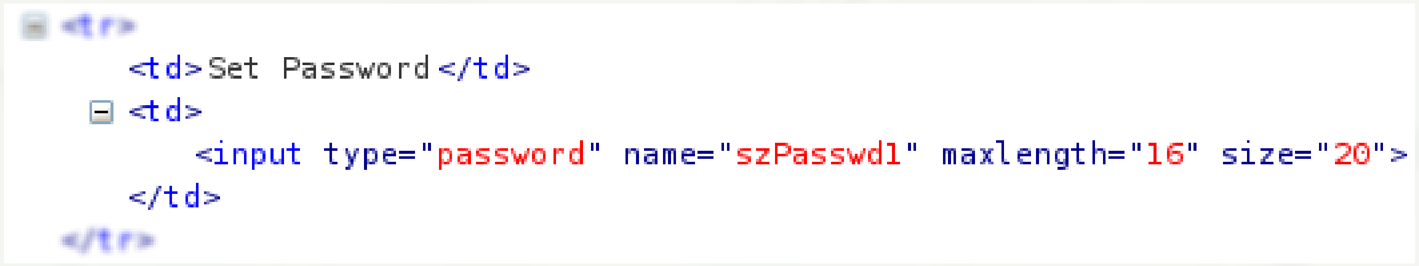
NOTE: Vulnerability is *PRE-AUTH* “per se”… but:

- Changing stored password requires knowledge of login credentials

⇨ ***POST-AUTH Exploitation***

Changing password

- Password can be changed via POST request
  - *<IP_address>/cgi-bin/passwd.html*
  - Client side restrictions on password size (....)



```
<tr>
    <td>Set Password</td>
    <td>
        <input type="password" name="szPasswd1" maxlength="16" size="20">
    </td>
</tr>
```

- No need to restart server:
  - New password wil be re-read at next authentication attempt

Exploitation strategy

- **Change admin password**
  - Send POST request:
    - URL: *http://<IP_address>/cgi-bin/passwd.cgi?passwd.html*
    - Body: *setobject_pwd=<payload>*
  - Embed valid basic authorization in request!

CR/LF not allowed in payload!

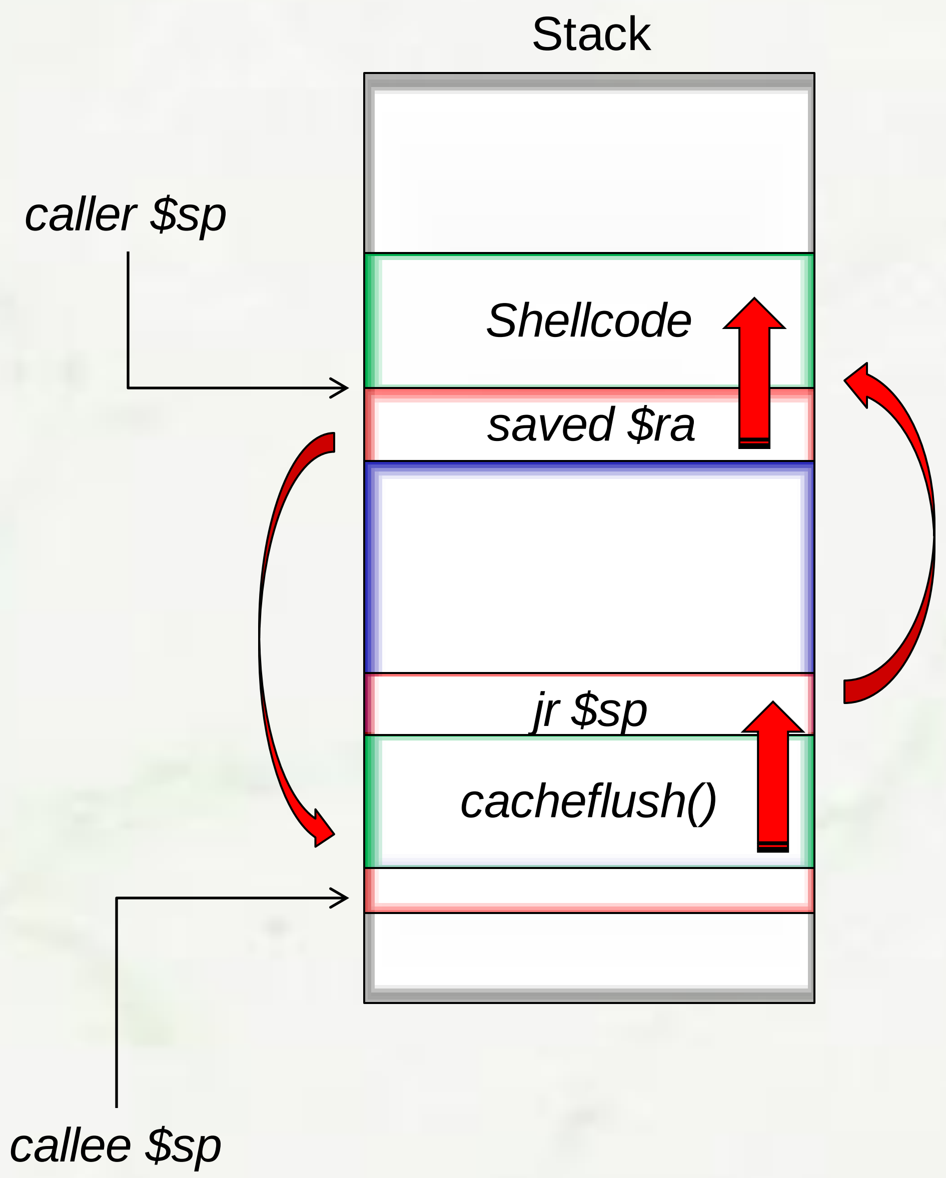
- **Attempt a new authentication**
  - Payload retrieved from NVRAM

Overflow occurs here!

- **Execute payload**
  - *$ra* saved in stack overwritten with payload address
  - *$ra* loaded from stack in function epilogue
  - *$sp* "raised" to value in caller function
  - *jr $ra*

Payload

- MIPSLE TCP Connect back shellcode (215 bytes):
  - no “\x00”, “\x0d”, “\x0a”
  - **Placed above the callee stack frame**
    - Too large for fitting in local buffer
- ***Unreliable*** if payload is directly executed (cache incoherency?)
  - ***Mitigation*** trick:
    - Use SYS_CACHEFLUSH Linux/MIPS syscall
    - jump to small (20 bytes) cacheflush shellcode in buf
    - ***cross fingers...***
    - jump at caller *$sp (jr $sp)*
- **NOTE:** Pad for alignment (2 bytes)

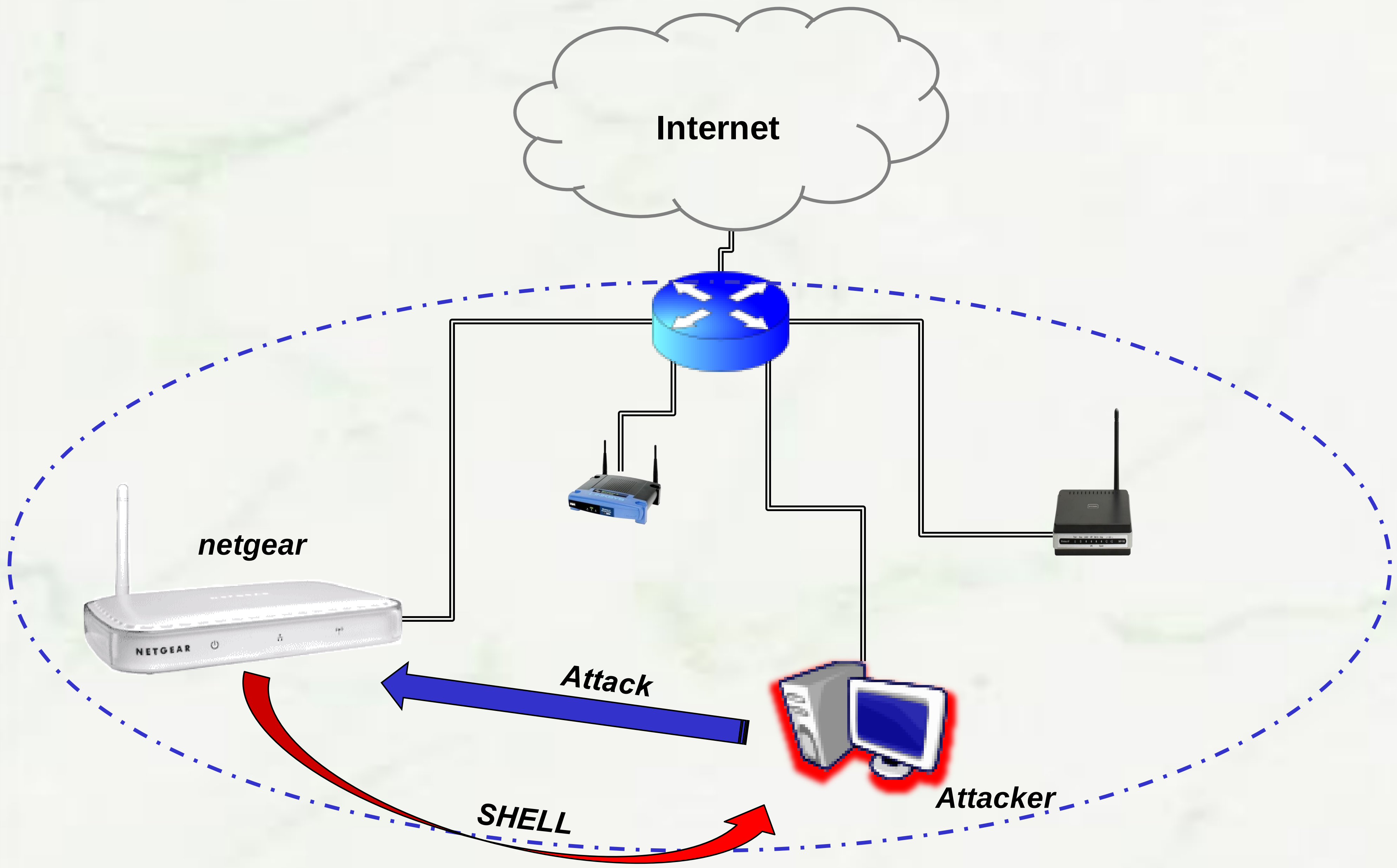


Netgear WG602v4

-

Demo

WG602v4 POST-AUTH Remote



- Interesting side effects:
  - Payload stored in Flash → **Survives to reboot!**
  - Payload executed at EVERY authentication → **A remote root shell comes for free ☺**
  - User is not able to authenticate via web → **Payload cannot be easily removed**
    - Payload can be removed via serial connection

- POST-Auth Remote attack demo'ed:
  - Can be upgraded to POST-Auth Remote Blind
    - Payload could be embedded into a malicious web page
    - Social engineering may entice user to perform authentication on target

D-Link DAP-1160

Overview

- ***CPU*:** MIPS @ 180 Mhz (Realtek SoC RTL8186)
- ***Byte "sex":*** Big-endian

- ***Memory***
  - 16Mbytes RAM
  - 4Mbytes Flash

- ***OS:*** Linux 2.4.18
- ***Web Server:*** CAMEO-httpd

- ***Firmware analysis***
  - Version: 1.20
  - Source code available: Yes (*only object files for httpd…)*
  - Firmware image available: Yes
  - Dumped firmware: No

Defaults:
IP: **192.168.0.50**
User: **admin**
Password: **<blank>**

An interesting find...

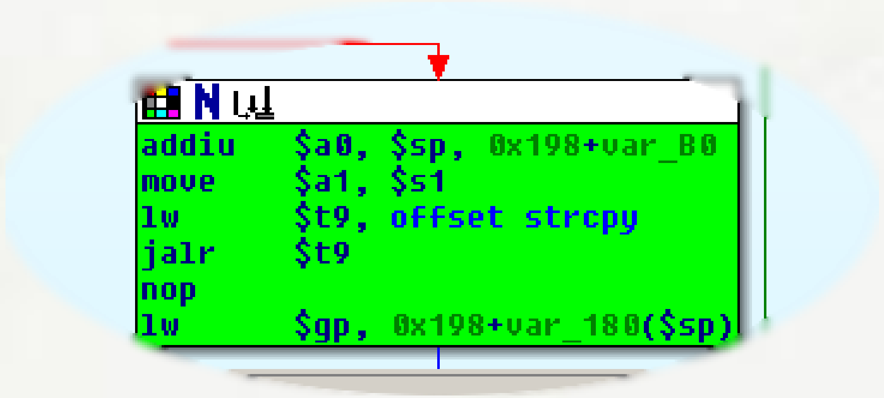
- Configuration changes applied by *apply.cgi*
  - Form handling functions specified as cgi params
    - ***eg: http://<IP_ADDR>/apply.cgi?handling_function***
- Filtering supported via formFilter() function
- Function not reachable by UI browsing... ***but***:
  - Referred by some non-linked (hidden?) webpages :
    - *Code meant for gateways??*
    - ***eg: http://<IP_ADDR>/adv_webfilter.htm***
  - Can be also directly called by:
    - ***http://<IP_ADDR>/apply.cgi?formFilter"***

Vuln 2.1: “URL filtering buffer overflow”

- URL filtering supported by formFilter function *(“Parental Control”)*

- Fixed size stack buffer for storing URL
- URL copied without length check

Buffer overflow!!



```
addiu   $a0, $sp, 0x198+var_B0
move    $a1, $s1
lw      $t9, offset strcpy
jalr    $t9
nop
lw      $gp, 0x198+var_180($sp)
```

- Auth **still** required...

POST-AUTH Exploitation

....but not for long ;-)

Exploitation strategy

- **Perform authentication**
  - Send POST request:
    - URL: *http://<IP_address>/apply.cgi?formPasswordAuth*
    - Body: *login_name=admin&login_pass=<b64encode(password)>*

- **Exploit**
  - Send *POST* request*:*
    - URL: *http://<IP_address>/apply.cgi?formFilter*
    - Body: *addFilterUrl=1&url=<payload>*
    - addFilterUrl=1 needed for taking vulnerable code path

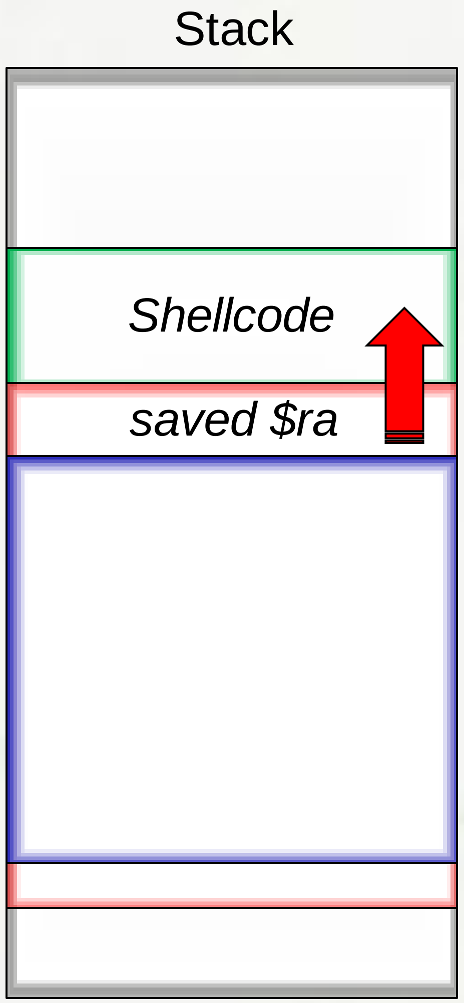
- **Payload**
  - MIPS Big Endian TCP connect back shellcode
  - No CR, LF, NULL

Payload

- Shellcode placed above stack frame
  - Too large for fitting in local buffer
    - 168 bytes available

- ***Stack is very stable!***
  - Saved *$ra* overwritten directly with shellcode address
  - NOP sled not even needed!

- No evident sign of cache incoherency

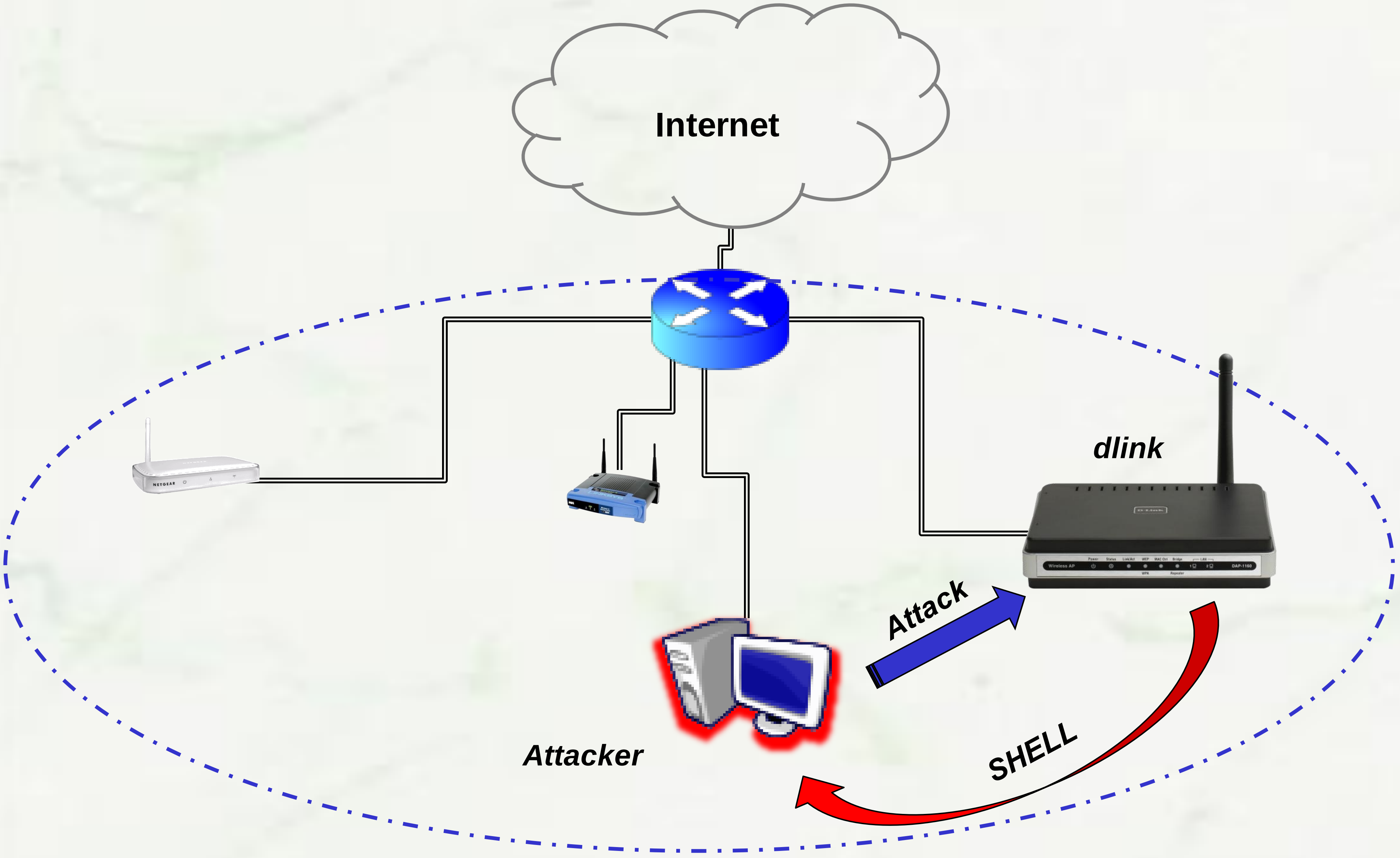


D-Link DAP-1160

–

Demo 1

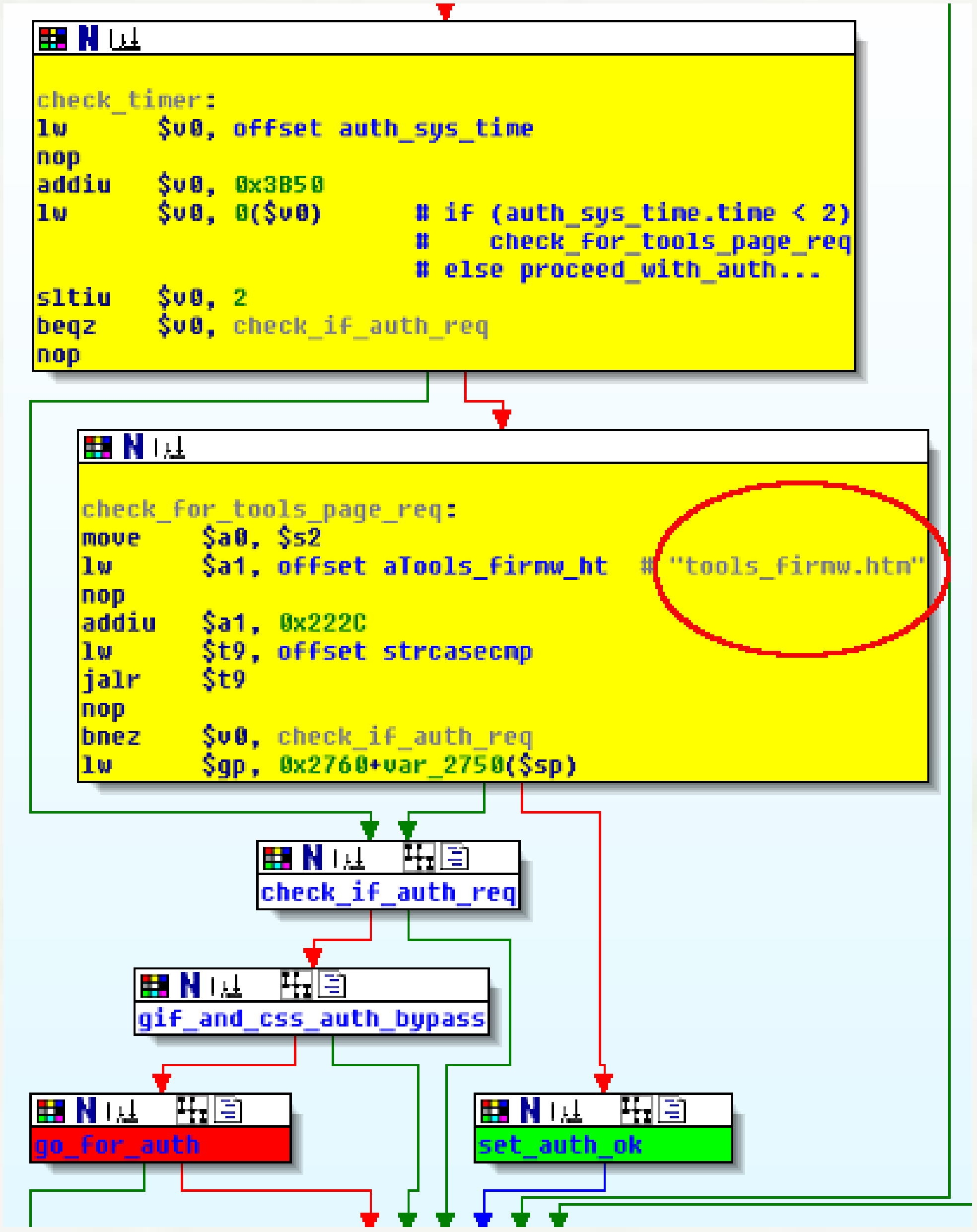
DAP-1160 POST-AUTH Remote



Vuln 2.2: Authentication bypass

- Accessing a specific web page allows **authentication bypass**:
  - *http://<IP_address>/tools_firmw.htm*
- ***Get a free ride! ☺***
  - Full unauthenticated access to the whole Web UI
- **Conditions:**
  - Must be ***first request &&***
  - ***within ~40 seconds*** from server start

Remote reboot?

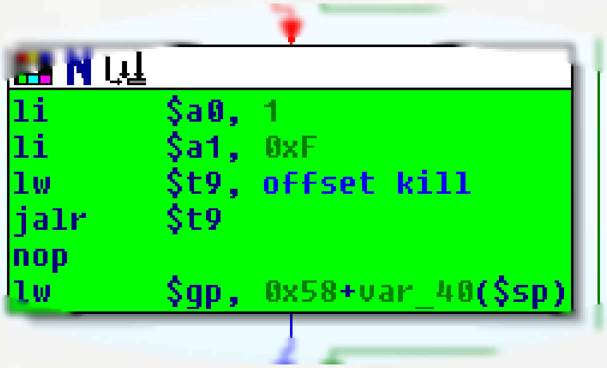


```
check_timer:
lw      $v0, offset auth_sys_time
nop
addiu   $v0, 0x3B50
lw      $v0, 0($v0)       # if (auth_sys_time.time < 2)
                          #     check_for_tools_page_req
                          # else proceed_with_auth...
sltiu   $v0, 2
beqz    $v0, check_if_auth_req
nop
```

```
check_for_tools_page_req:
move    $a0, $s2
lw      $a1, offset aTools_firmw_ht    # "tools_firmw.htm"
nop
addiu   $a1, 0x222C
lw      $t9, offset strcasecmp
jalr    $t9
nop
bnez    $v0, check_if_auth_req
lw      $gp, 0x2760+var_2750($sp)
```

DCCD: These reBOOTS are made for..

- DCC (D-LINK Click 'n Connect) makes AP configuration: easier
  - UDP daemon on port 2003 (DCCD)
  - Unathenticated access
- ***Rebooting*** is one of the *"supported"* functionalities...
- Sending binary command to DCCD:
  - Sends SIGTERM to *init*
  - AP reboots



```
li      $a0, 1
li      $a1, 0xF
lw      $t9, offset kill
jalr    $t9
nop
lw      $gp, 0x58+var_40($sp)
```

"\x05\x00" + "\x00" * 6

Attack Upgrade: NO-AUTH Remote exploitation

Reboot

*“\x05\x00” + “\x00” * 6*

2003/UDP (DCCD)

Sleep

…ZzZzZZz…

Auth bypass

http://<IP_ADDR>/tools_firmw.htm

Exploit

URL filtering buffer overflow...

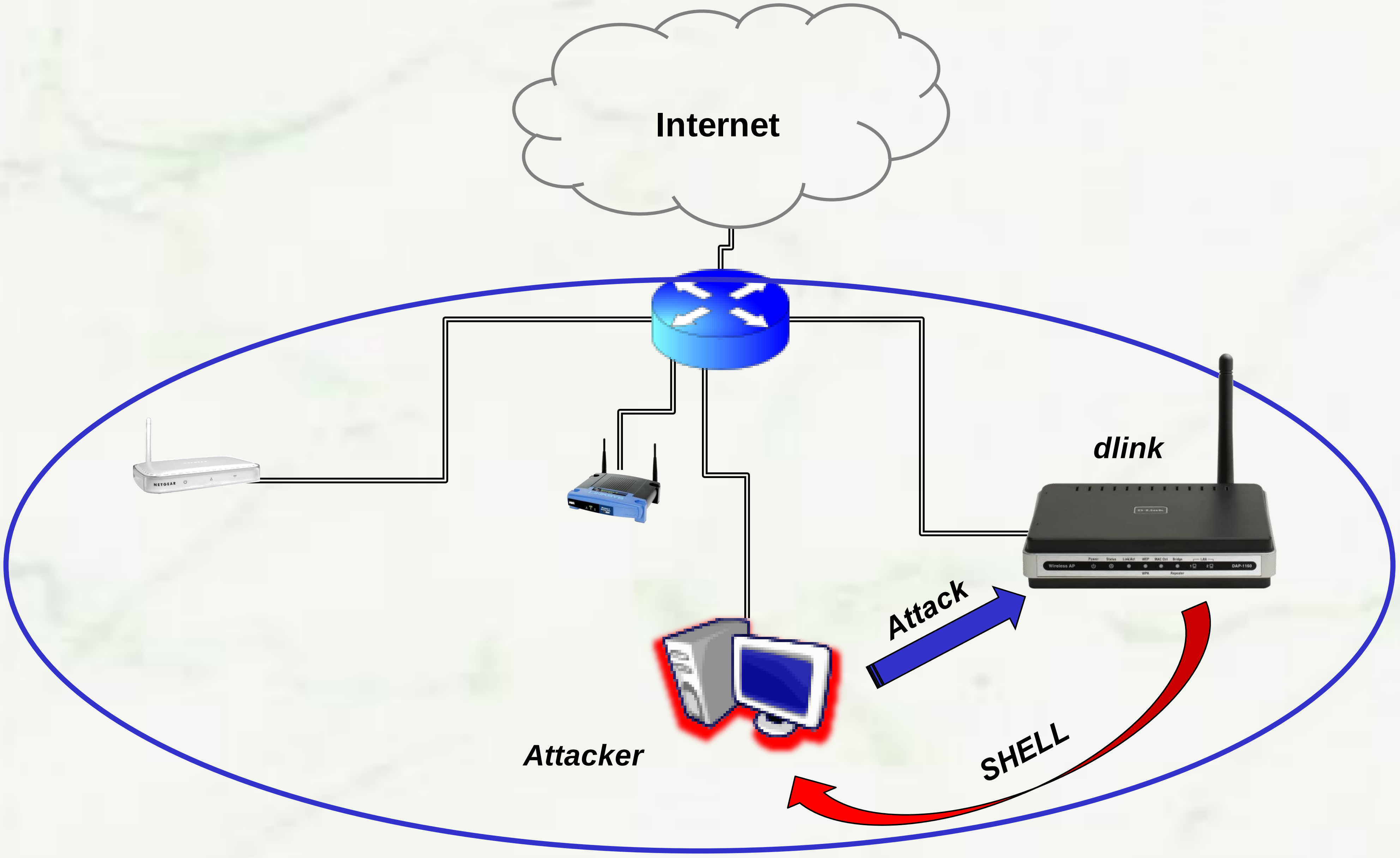
Enjoy your shell!

D-Link DAP-1160

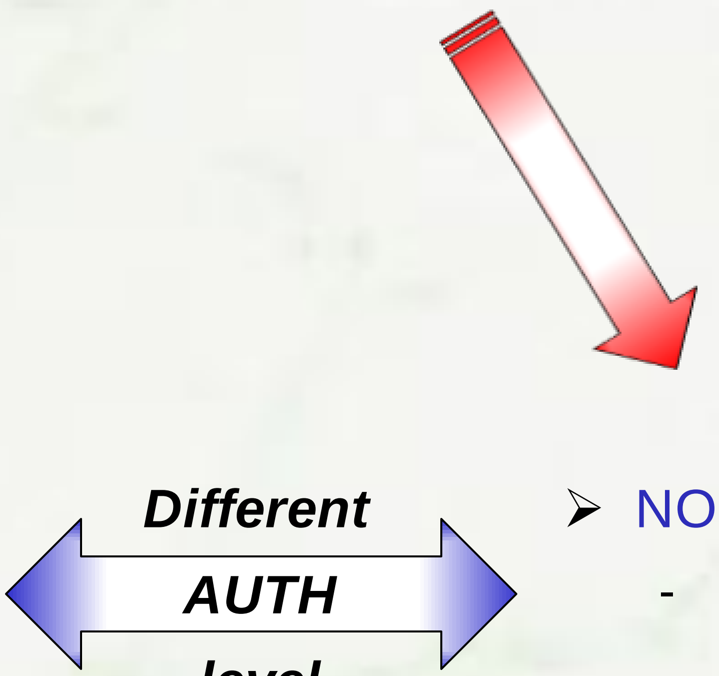
–

Demo 2

DAP-1160 NO-AUTH Remote



ONE vulnerability...



- POST-Auth Remote attack
  - Authentication needed ***but..***
  - Can be upgraded to Remote Blind

- NO-Auth Remote attack
  - Auth bypassed ***but...***
  - Not easily upgraded to Remote Blind

....TWO attack flavours

Linksys WAP54G

Overview

- ***CPU*:** MIPS @ 200 Mhz (Broadcom SoC BCM5352)
- ***Byte "sex":*** Little-endian

- ***Memory***
  - 8Mbytes RAM
  - 2Mbytes Flash

- ***OS:*** Linux 2.4.20
- ***Web Server:*** milli_httpd

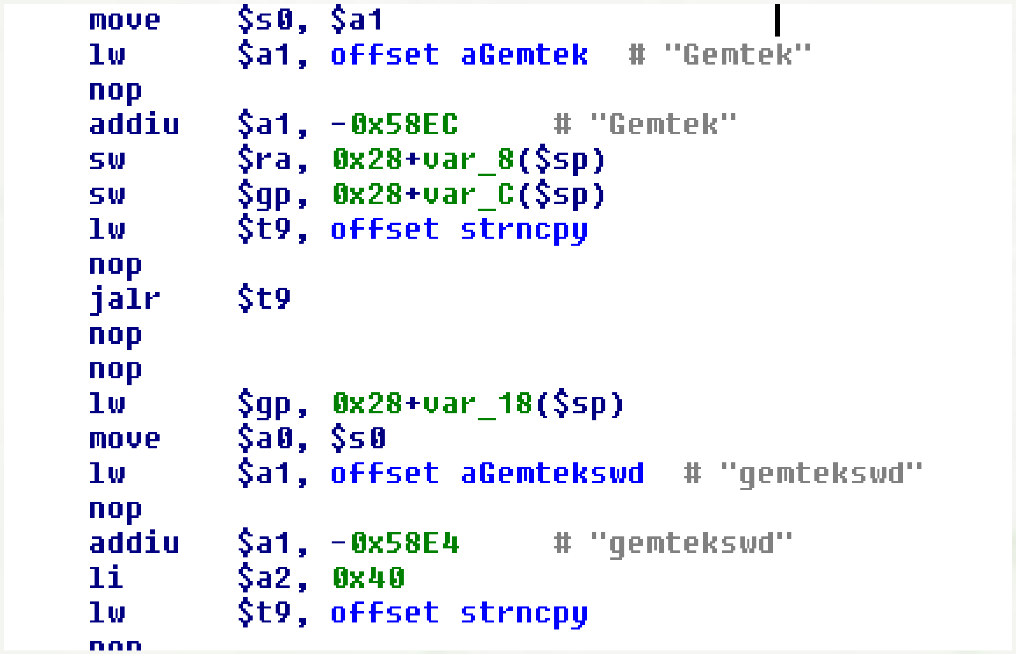
- ***Firmware analysis***
  - Version: EU 3.05 (.03?)
  - Source code available: Yes (version 3.04.03)
  - Firmware image available: Yes
  - Dumped firmware: No

Defaults:
IP: **192.168.1.245**
User: **<blank>**
Password: **admin**

Vuln 3.1: Hidden Debug

- An **hidden account** is present on the device
  - Used only for accessing a *debug page*
  - Can be used with HTTP Basic Authentication
  - Cannot be used for accessing the normal UI

- ***BUT…***
  - *Embedded in firmware*
  - *Cannot be changed by user!*



```
move    $s0, $a1
lw      $a1, offset aGemtek   # "Gemtek"
nop
addiu   $a1, -0x58EC      # "Gemtek"
sw      $ra, 0x28+var_8($sp)
sw      $gp, 0x28+var_C($sp)
lw      $t9, offset strncpy
nop
jalr    $t9
nop
nop
lw      $gp, 0x28+var_18($sp)
move    $a0, $s0
lw      $a1, offset aGemtekswd   # "gemtekswd"
nop
addiu   $a1, -0x58E4      # "gemtekswd"
li      $a2, 0x40
lw      $t9, offset strncpy
nop
```

User: **Gemtek**
Password: **gemtekswd**

And....

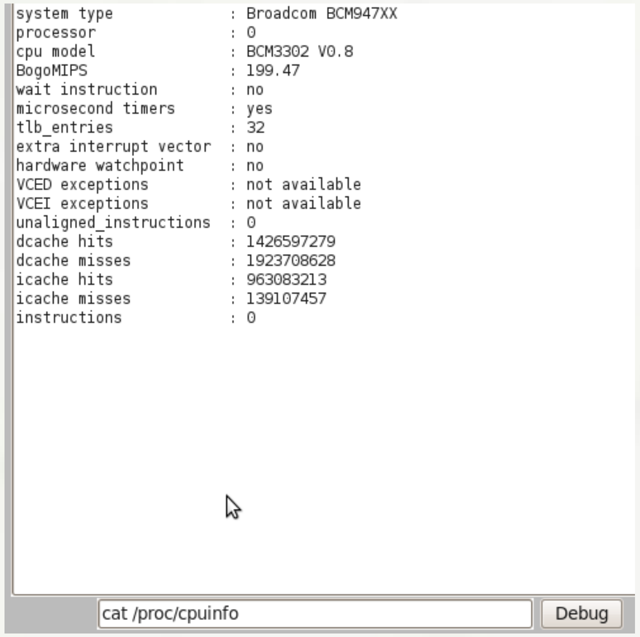
Vuln 3.1: Hidden Debug (cont' ed)

- **Debug interface** accessible with hidden account:
  - root shell over HTTP
  - *URL: http://<IP_ADDR>/debug.cgi*

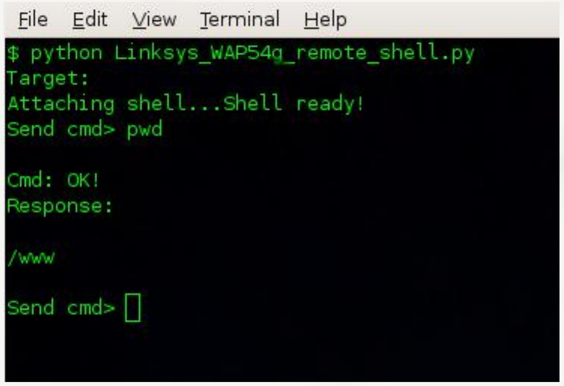
- Handled by function *cgi_cmd_ui_debug*:
  - located outside httpd code branch (?)
    - *release/src/shared/broadcom.c*

- **A bunch of vulns**:
  - Credentials extraction and modification:
    - *eg: nvram get http_passwd*
  - Command injection
  - XSS

But...we're interested in binary exploitation!



```
system type             : Broadcom BCM947XX
processor               : 0
cpu model               : BCM3302 V0.8
BogoMIPS                : 199.47
wait instruction        : no
microsecond timers      : yes
tlb_entries             : 32
extra interrupt vector  : no
hardware watchpoint     : no
VCED exceptions         : not available
VCEI exceptions         : not available
unaligned_instructions  : 0
dcache hits             : 1426597279
dcache misses           : 1923708628
icache hits             : 963083213
icache misses           : 139107457
instructions            : 0
```

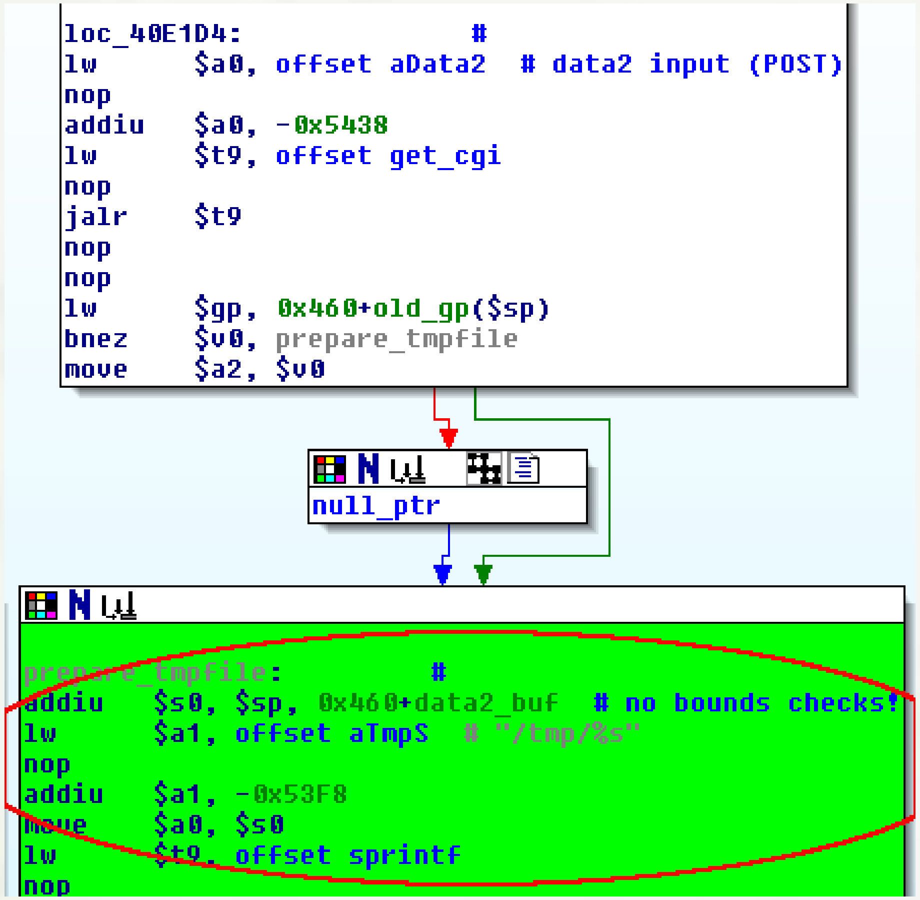


```
$ python Linksys_WAP54g_remote_shell.py
Target:
Attaching shell...Shell ready!
Send cmd> pwd

Cmd: OK!
Response:

/www

Send cmd> 
```

a quick shell

Vuln 3.2: debug.cgi buffer overflow(s)

- Code processes 3 POST variables
  - *data1 (command), data2 (tmpfile), data3 (PID to be killed)*
- Two stack buffers for allocating *data1* and *data2:*
  - *data2* buffer allocated *above data1* buffer
- **Buffer overflows possible for *both(!)* variables**



```
loc_40E1D4:                 #
lw      $a0, offset aData2   # data2 input (POST)
nop
addiu   $a0, -0x5438
lw      $t9, offset get_cgi
nop
jalr    $t9
nop
nop
lw      $gp, 0x460+old_gp($sp)
bnez    $v0, prepare_tmpfile
move    $a2, $v0
```

```
null_ptr
```

```
prepare_tmpfile:            #
addiu   $s0, $sp, 0x460+data2_buf  # no bounds checks!
lw      $a1, offset aTmpS  # "/tmp/%s"
nop
addiu   $a1, -0x53F8
move    $a0, $s0
lw      $t9, offset sprintf
nop
```

Debug account access ⇨ ***NO-AUTH Exploitation!!***

Exploitation strategy

- **Exploit**
  - Send POST request:
    - URL: *http://<IP_address>/debug.cgi*
    - Body: *data1=<payload>&data2=<align_padding><payload_address * n>*
  - Embed hidden debug account in HTTP Authentication header

- **Payload executed**
  - MIPS Little Endian TCP connect back shellcode
  - Sent as Percent-encoded
    - Decoded by unescape() function
    - Allows for inclusion of otherwise problematic chars (eg: ‘&+’)

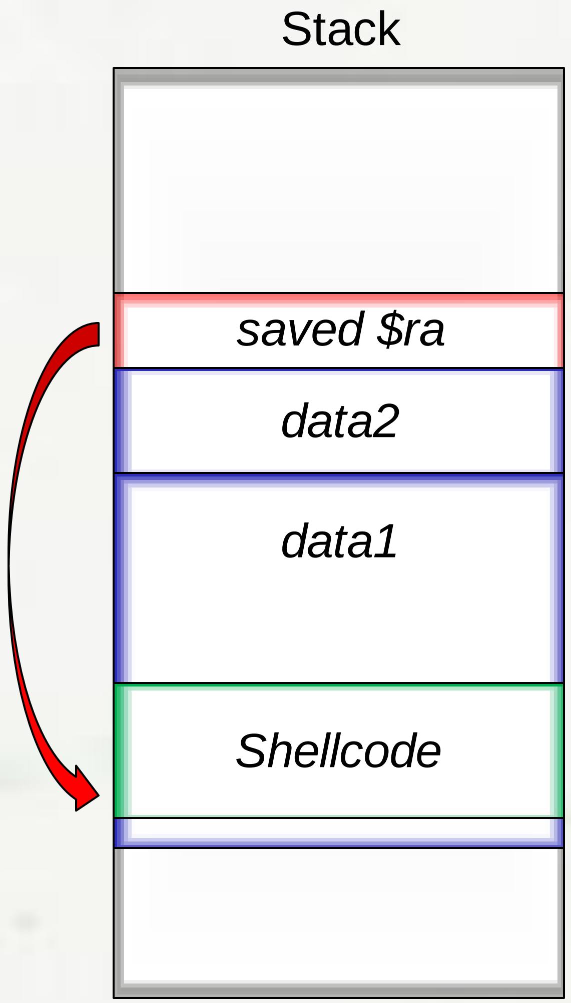
Payload

- Shellcode placed in *data1* buffer
    - Buffer size: 1024 bytes

- *Saved $ra* overwritten via ***data2*** **buffer overflow**

- ***Stack is very stable!***
    - Saved *$ra* overwritten directly with shellcode address
    - NOP sled not even needed!

- No evident sign of cache incoherency

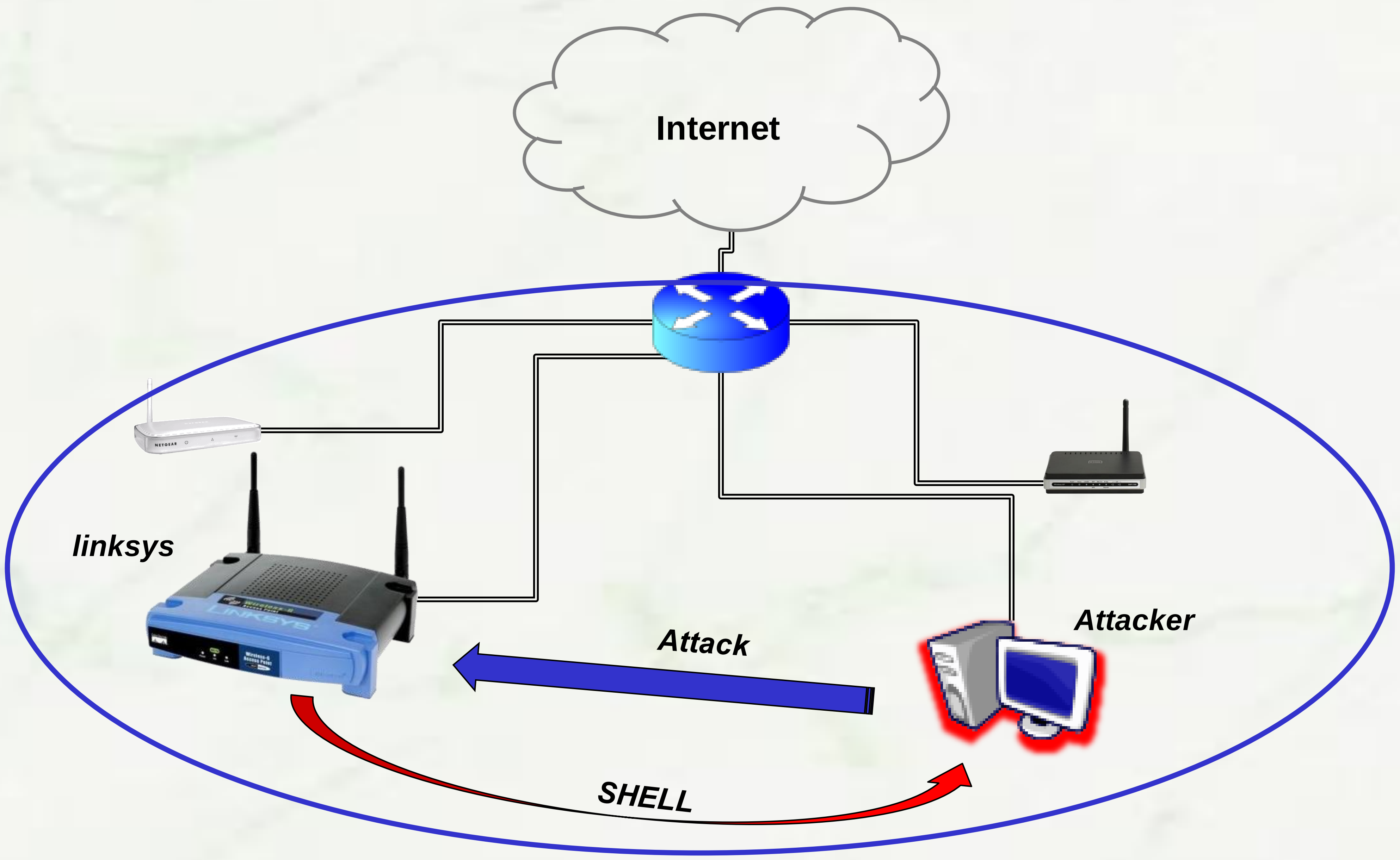


Linksys WAP54G

–

Demo 1

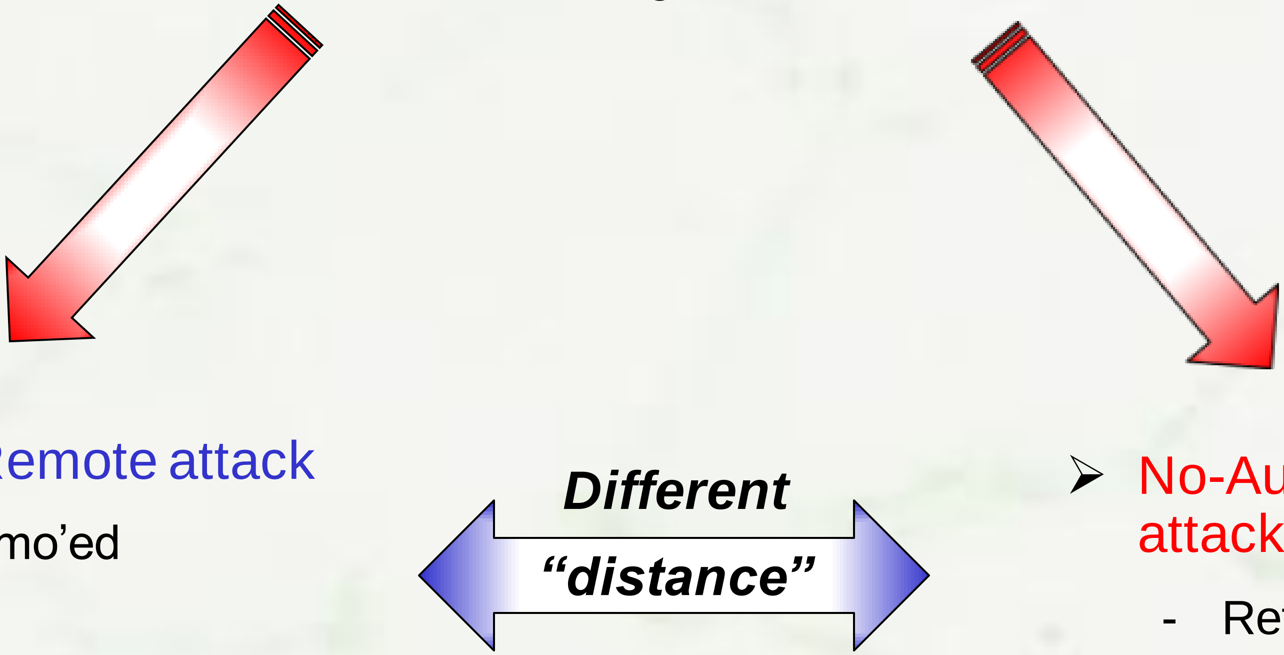
WAP54g NO-AUTH Remote



- ***Vulnerability***
  - Found in debug code
  - Authentication bypass via debug account

"WORMABLE"!

Fix needed!!!

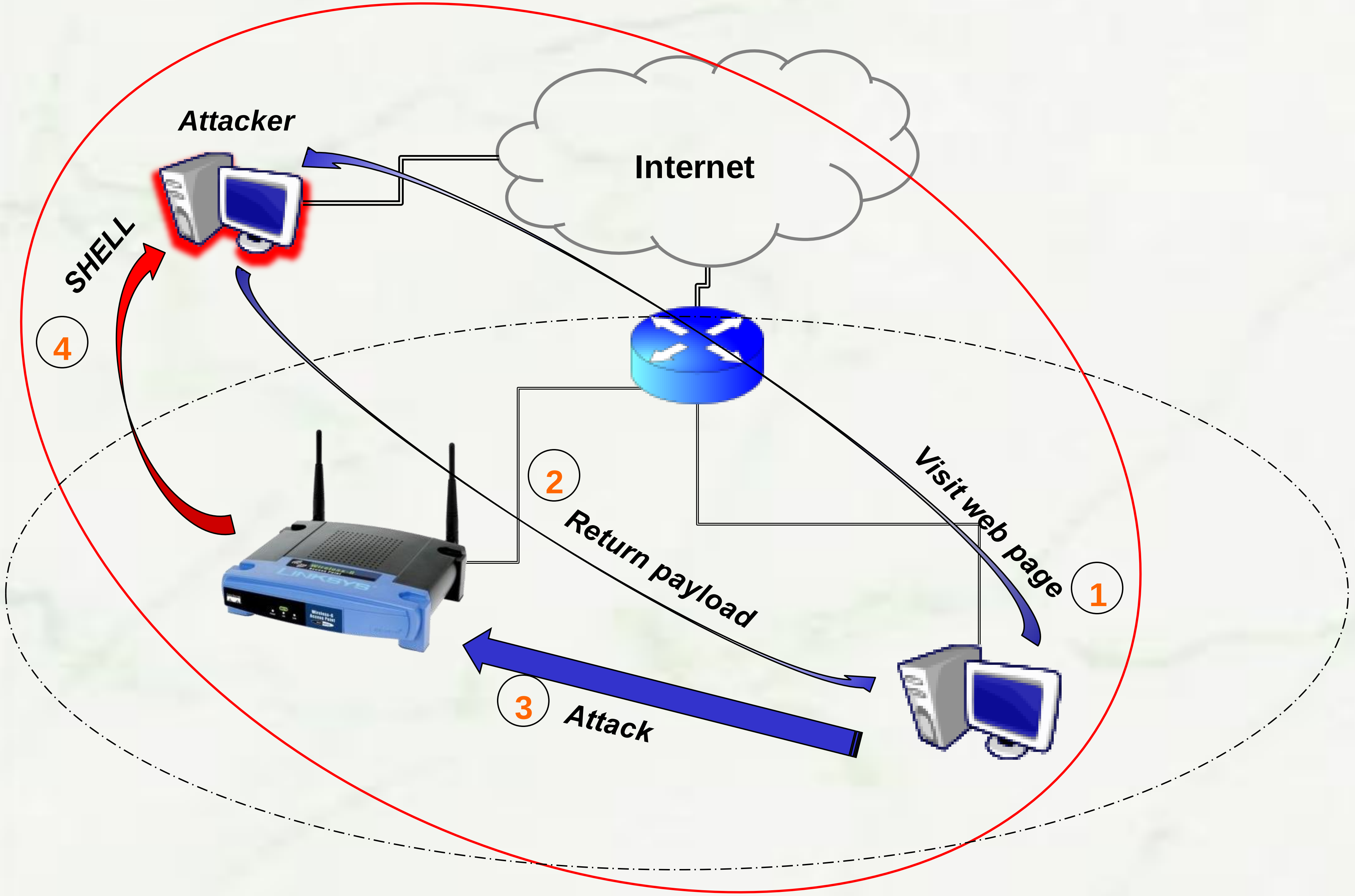


- No-Auth Remote attack
  - Just demo'ed

- No-Auth Remote Blind attack
  - Reflection possible
  - ***See next demo...***

Linksys WAP54G – Demo 2

WAP54g NO-AUTH Remote Blind



- ***No-Auth Remote Blind*** attack
  - Demonstrated with:
    - Firefox 3.6.3
  - Javascript only needed

User visits malicious page...

Attacker gets reverse root shell!

URL shortening anyone??

Back to base...

Summary

- **Achieved 100% of *Primary Goals***
  - Exploitation of targets loaded with stock firmware
    - TCP connect-back root shell on each
  - Target proximity *not required*
    - *Remote* exploitation ***demonstrated*** in all the cases
    - *Remote blind* exploitation possible in all the cases

- ***Secondary Goals:***
  - One *No-auth Remote* attack ***demonstrated*** *(D-Link DAP-1160)*
  - One *No-Auth Remote Blind* attack **demonstrated** *(Linksys WAP54g)*

Conclusions

- A *determined* attacker may easily take ***complete control***
  - **Easy finding vulnerabilities**
  - **Exploitation "per se" is smooth:**
    - *NO countermeasures (eg: Stack Canaries, ASLR, DEP..)*
    - Root privileged services..
  - **More challenging:**
    - Dealing with firmware images
    - Exploit development (writing tools & shellcodes, debugging)
    - Exploit reliability (separate caches)

Questions

Thanks!!!

References

- Dominic Sweetman - “See MIPS Run” – Morgan Kaufmann
- MIPS Technologies - “MIPS32™ Architecture For Programmers”
- scut - “Writing MIPS/IRIX shellcode”
- Julien TINNES – “Linux MIPS ELF reverse engineering tips”
- Raphaël Rigo - mips-analyzer IDA Pro plugin (http://syscall.eu/progs/)
- Peter Werner - “Writing MIPS exploits” – Ruxcon 2003
- Laurent Butti – Julien Tinnes – Franck Veysr - “Wi-Fi Implementation Bugs: an Era of New Vulnerabilities” – Hack.lu 2007.
- Michal Sajdak - “Remote root shell on a SOHO class router” – Confidence 2009
- Flash Based UPNP attacks (http://www.gnucitizen.org/blog/flash-upnp-attack-faq)
- Barnaby Jack – “Exploiting Embedded Systems” – BlackHat Europe 2006
- FX – “Cisco IOS Attack & Defense – State of the Art” – 25C3
- Paul Asadorian - “Things That Go Bump In The Network: Embedded Device (In)Security” – 2008 SANS/New Orleans
- Alexander Sirotkin – “Hacking embedded Linux” – LinuxConf 2007
- Naxxatoe – “Malware for SOHO Routers”
- Columbia University: Ang Cui, Yingbo Song, Pratap V. Prabhu and Salvatore J. Stolfo - “Brave New World: Pervasive Insecurity of Embedded Network Devices” – June 2009

Contacts

Cristofaro Mune

pulsoid_at_icysilence_dot_org

http://www.icysilence.org