Too Many Cooks

Exploiting the Internet of TR-069 Things

Shahar Tal @jifa
Lior Oppenheim @oppenheim1
Malware and Vulnerability Research @ Check Point

We Secure the Internet.

1. Find Problems
2. Tell Vendors
3. Share with Community
Agenda

• TR-069 quick tour / DEF CON recap
• Motivation
• The TR-069 Census 2014
• Research Highlights
• Mass Pwnage
• A Pessimistic Outlook
TR-069

- a.k.a. **CPE WAN Management Protocol (CWMP)**
  - 2004: v1.0
  - 2013: v1.4 (amendment 5)
  - 2015: amendment 6?

- This is what ISPs use to provision, monitor and configure your home routers (and more)
TR-069 Provisioning Session

SOAP RPC
(XML over HTTP)

Always initiates session
ACS can issue “Connection Request”

Inform(reason)
<empty>
GetParameterValues(<key>*)
SetParameterValues(<key, value>*)
<empty>

Dual authentication mechanism
Presented at DEF CON 22

Our research uncovered implementation and configuration flaws in many ISP’s ACS deployments

- ACSs are a single point of pwnage in modern ISP infrastructure
- Many TR-069 implementations just aren’t serious enough
- Leads to ISP fleet takeover

REMTLEY MANAGE

ALL THE THINGS
Connection Request

- "The ACS can at any time request that the CPE initiate a connection to the ACS using the Connection Request notification mechanism. Support for this mechanism is REQUIRED in a CPE."

<table>
<thead>
<tr>
<th>Port</th>
<th>Service</th>
<th>Hit Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>HTTP</td>
<td>1.77</td>
</tr>
<tr>
<td>7547</td>
<td>CWMP</td>
<td>1.12</td>
</tr>
<tr>
<td>443</td>
<td>HTTPS</td>
<td>0.93</td>
</tr>
<tr>
<td>21</td>
<td>FTP</td>
<td>0.77</td>
</tr>
<tr>
<td>23</td>
<td>Telnet</td>
<td>0.71</td>
</tr>
</tbody>
</table>
Port 80 Analysis

- Port 80 - ~70m
  - 50% Web Servers
  - 50% IoT things
    - Routers
    - Webcams
    - VoIP Phones
    - Toasters

Apache

IIS8

NGINX
Port 7547 Analysis

- TR-069 - ~45m
  - 100% IoT
The TR-069 Census 2014

• We scanned 7547 (Nov 2014)
  – A few times
  – Help from friends (Rapid7, UMich)

• 1.18% respond
  – 46,093,733 IoT devices
  – All over the world
  – 0.06% = 2.2m
TR-069 CR Server Distribution

Apache 15%
KTT-SOAP 8%
mini_httpd 6%
gSOAP 19%
RomPager 52%
What is RomPager

Embedded HTTP server by Allegro Software
  - Massachusetts based company

Optimized for minimal environments
  - Small binary, small memory requirements

First introduced in 1996

Many versions since
  - Current version in 5.4
THIS IS WHERE I GREW SUSPICIOUS.
1. As we have updated the security mechanism on the device, once you have upgraded to this firmware, you will not be able to downgrade to the old one.

2. You have to restore the device to factory default, new functions take effect; Click Maintenance->System->Factory Default, choose Factory Default Settings, Click RESTART.
RomPager 4.07

- Dated to 2002
- Appears in many new firmwares

- 2,249,187 devices on port 80
- 11,328,029 devices on port 7547

- 200 different identified models
- 50 different brands
**Dig Deeper**

- Explore the firmware
  - Firmware update is one file called "ras"
  - Binwalk

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEX</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>84992</td>
<td>0x14C00</td>
<td>ZynOS header, header size: 48 bytes, rom image type: ROMBIN, uncompressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ags: 0xE0, uncompressed checksum is valid, the binary is compressed,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>compressed checksum is valid, memory</td>
</tr>
<tr>
<td>85043</td>
<td>0x14C33</td>
<td>LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, un-</td>
</tr>
<tr>
<td>128002</td>
<td>0x1F402</td>
<td>GIF image data, version 8&quot;9a&quot;, 200 x 50</td>
</tr>
<tr>
<td>136194</td>
<td>0x21402</td>
<td>GIF image data, version 8&quot;9a&quot;, 560 x 50</td>
</tr>
<tr>
<td>350208</td>
<td>0x55800</td>
<td>ZynOS header, header size: 48 bytes, rom image type: ROMBIN, uncompressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ags: 0xE0, uncompressed checksum is valid, the binary is compressed,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>compressed checksum is valid, memory</td>
</tr>
<tr>
<td>350259</td>
<td>0x55833</td>
<td>LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, un-</td>
</tr>
</tbody>
</table>

- **Bootloader**
- **Vendor logo**
- **Main binary**
• Downloaded all the RomPager 4.07 firmwares I could find

• All of them had ZynOS header! (mipsb32)
ZynOS

• Basic RTOS
• One binary
• No file system

• Notoriously known for the “rom-0” vulnerability (CVE-2014-4019)
  — 1,219,985 vulnerable world-wide (May 2014)
http://192.168.1.1

Authentication Required

The server http://10.10.10.199:80 requires a username and password. The server says: TD-W8961ND.

User Name: 
Password: 

Protected Object

Username or Password error
http://192.168.1.1:7547

Object Not Found

The requested URL '/' was not found on the RomPager server.

Return to last page
Manual Testing

- Fuzzing over http headers
- Crashed on username sub-header of digest authentication
  
  {Authorization: Digest username='a'*600}
Handling HTTP requests

```
la $t0, HttpRedirectHandler
sw $v0, 0x24($a0)
la $t7, aContentLengt_0  # "content-length"
sw $t7, 0x34($a0)
li $t5, 0xE
sh $t5, 0x38($a0)
la $t2, HttpContentLengthHandler
sw $t2, 0x30($a0)
la $t0, aReferer    # "referer"
sw $t0, 0x40($a0)
li $a2, 7
sh $a2, 0x44($a0)
la $v1, HttpRefererHandler
sw $v1, 0x3C($a0)
la $t8, aHost       # "host"
sw $t8, 0x4C($a0)
li $t6, 4
sh $t6, 0x50($a0)
la $t3, HttpHostHandler
sw $t3, 0x48($a0)
la $t1, aAuthorization  # "authorization"
sw $t1, 0x58($a0)
li $a3, 0xD
sh $a3, 0x5C($a0)
```
Vulnerability #1

Start 0x8010e234

.ent DigestUsernameHandler

var_8= -8
var_4= -4

addiu $sp, -8
addiu $a0, 0x3D60
sw $ra, 8+var_4($sp)
addu $at, $a1, $a2
sw $fp, 8+var_8($sp)
sh $zero, 0($at)

jal strcpy

move $fp, $sp
lw $ra, 8+var_4($sp)
lw $fp, 8+var_8($sp)
jr $ra

addiu $sp, 8
.end DigestUsernameHandler

End 0x8010e264
Pimped up my router

- Open up the router, looking for JTAG
- No JTAG
- U-ART?
TLB refill exception occurred!

EPC = 0x61616161
SR = 0x10000003
CR = 0x50801808
$RA = 0x00000000

Bad Virtual Address = 0x61616160

UTLB_TLBL ..\core\sys_isr.c:267 sysreset()

$r0 = 0x00000000 $at = 0x80350000 $v0 = 0x00000000 $v1 = 0x00000001
$at = 0x805D7AF8 $a2 = 0xFFFFFFFF $a3 = 0x00000000
$t0 = 0x8001FF80 $t1 = 0xFFFFFFFF $t3 = 0x804A9E47
$t2 = 0x804A8F38 $t4 = 0x804A96D0 $t6 = 0x804A9D00
$s0 = 0x804A8A60 $s1 = 0x804C114 $s2 = 0x805E2BF8 $s3 = 0x80042A70
$s4 = 0x804A96D0 $s5 = 0x80000000 $s6 = 0x804E5FC $s7 = 0x00000000
$s8 = 0x804A96D0 $s9 = 0x00000000 $s10 = 0x61616160 $s11 = 0x80000000
$gp = 0x8040F004 $sp = 0x805E2B90 $fp = 0x805E2BF8 $ra = 0x8003A3D0

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
805e2bf8: 80 5e 2c 28 80 04 2a 70 80 40 f8 ac 80 40 f3 e0  .^,(..*p.@.
805e2c08: 80 40 e5 fc 00 00 00 00 80 40 e6 0c 80 48 4e 29  @.........@
805e2c18: 00 55 54 4c 42 5f 54 4c 42 4c 00 ac 00 00 00 00  .UTLB_TLBL
805e2c28: 80 5e 2c 40 80 10 16 d0 80 40 f3 e0 00 00 00 00  .^,@......@
Exploit #1

- Unprotected strcpy
- 1. send large username
- 2. overwrite function pointer with ptr to shellcode
- 3. profit!
- Too easy?
Variance in the wild

- Each device/firmware version has a different address space layout ("Nature’s ASLR")
- If you know your target firmware and the exact memory layout, you can run code without too much hassle
- Attacker gets one chance per router because of dynamic IP allocation
- A potential generic solution would include finding an anchor for the shellcode using another infoleak vuln.
- That could work, but let’s keep looking!
Poor Man's GDB

• ZynOS has unknown memory access debug primitives in serial
  — Pre-boot
• Dynamic reversing is very slow
  — Patch, crash, repeat
• ZORDON - ZynOs Remote Debugger (Over the Network)
  — Breakpoints
  — View/Edit Memory and registers
Vulnerability #2

• Each incoming HTTP request populates a pre-allocated “request structure”.
  — No dynamic memory allocation, remember?
• RomPager 4.07 handles processing of up to 3 concurrent requests (3 pre-allocated structures)
• By sending 3 consecutive requests, one can overwrite the HTTP handlers structures
TLB refill exception occurred!
EPC = 0x61616161
SR  = 0x10000003
CR  = 0x5801808
$RA = 0x00000000
Bad Virtual Address = 0x61616160
UTLB_TLBL ..\core\sys_isr.c:267 sysreset()

$r0 = 0x00000000 $at = 0x80350000 $v0 = 0x00000000 $v1 = 0x00000001
$a0 = 0x00000001 $a1 = 0x805D7AF8 $a2 = 0xFFFFFFFF $a3 = 0x00000000
$t0 = 0x8001FF80 $t1 = 0xFFFFFFFF $t2 = 0x804A8F38 $t3 = 0x804A9E47
$t4 = 0x804A9E60 $t5 = 0x804A8A60 $t6 = 0x804A9D00 $t7 = 0x00000040
$s0 = 0x804A8A60 $s1 = 0x8040C114 $s2 = 0x805E2BF8 $s3 = 0x80042A70
$s4 = 0x00000001 $s5 = 0x8000007C $s6 = 0x8040E5FC $s7 = 0x00000000
$t8 = 0x804A9E48 $t9 = 0x00000000 $k0 = 0x616161610 $k1 = 0x8000007C
$gp = 0x8040F004 $sp = 0x805E2B90 $fp = 0x805E2BF8 $ra = 0x8003A3D0

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
805e2bf8: 80 5e 2c 28 80 04 2a 70 80 40 f8 ac 80 40 f3 e0
805e2c08: 80 40 e5 fc 00 00 00 00 80 40 e6 0c 80 48 4e 29
805e2c18: 00 55 54 4c 42 5f 54 4c 42 4c 00 ac 00 00 00 00
805e2c28: 80 5e 2c 40 80 10 16 d0 80 40 f3 e0 00 00 00 00
805e2c38: 80 40 e8 00 00 00 00 00 00 5e 2c 50 80 10 1e 00
EXPLOIT #2

• How can you exploit this?
  — Blind memory read (by replacing the HTTP header string ptr)

• Problem: only works on port 80.
  — already have “rom-0” for that
VULNERABILITY #3

• Rom pager supports cookies
  – No dynamic memory allocation, remember?
• Pre-allocated cookies array
  – 10 cookies, 40 bytes long each
  – C0,C1,C2,...,C9

Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,he;q=0.6
Cookie: C0=21232f297a57a5a743894a0e4a801fc3;

HTTP/1.1 200 OK
Content-Type: text/html
Date: Sat, 01 Jan 2000 00:05:13 GMT
addiu $s0, 1
move $a0, $s0
jal FindTokenDelimiter
nop
move $a0, $s0
move $s1, $v0
addiu $s1, 1
jal atoi
sb $zero, -1($s1)
move $a0, $s1
jal FindCookieEnd
move $s3, $v0
li $a2, 40
mul $t2, $s3, $a2
move $a1, $s1
move $s0, $v0
addiu $t5, $s4, 0x6B28
move $s0, $v0
addu $at, $s1, $s0
addu $a0, $t5, $t2
jal strncmp
sb $zero, 0($at)
j loc_8010E644
j loc_8010E644
addu $s0, $s1, $s0
Exploit #3 - Misfortune Cookie

- Arbitrary memory write relative to a fixed anchor in the RomPager internal management struct
  - Pretty much controls everything RomPager does
  - Overflow 32-bit for negative offsets 😊

- Non-harmful example as a POC:
  
  
  ```
  cookie: c107373883=/omg1337hax
  ```

  Object Not Found

  The requested URL '/omg1337hax' was not found on the RomPager server.

  Return to [last page](#)

- The technique works on any model of any brand that we had access to
Exploit #3 - Misfortune Cookie

With a few magic cookies added to your request you bypass any authentication and browse the configuration interface as admin, from any open port.
Countermeasures

- Cancel Internet subscription
- Alternative firmware
- Don’t buy these models until they’re fixed

Too Many Cooks Do Spoil the Broth
We contacted AllegroSoft and the major affected vendors
  — Provided full description of the vulnerability and a non-harmful POC that triggers it
• Despite some broken English, the message got through
  — Most of the time
  — Some patched firmware already out
• AllegroSoft
  — “Can’t force any vendor to upgrade to latest version” (they actually provided a patched version in 2005)
F.A.Q.

• Is RomPager bad?
  — No, they were actually very responsive and security aware. We just happened to research an old version of their software.

• Is this an intentionally placed backdoor?
  — Doesn’t look like it.

• Can you share the exploit?
  — No.

• Can you tell me which IPs are affected in my country?
  — Scan 80 + 7547 + custom ISP TR-069 connection request ports
We found a pretty serious vulnerability in the most popular service exposed in IPv4.

– As far as we know

Hey industry, fix this.
Thank You

@jifa @oppenheim1