Hacking Team

how they infected your Android device by 0days

Attila Marosi
Senior Threat Researcher
OSCE, OSCP, ECSA, CEH
Agenda

• Hacking Team - Remote Control System (RCS)
• Leak – ]Hacked Team[
• WebView exploit for Android
• DEMO
• Avoiding techniques they used
• Conclusion
Hacking Team – Remote Control System

• Product for Law-enforcement agencies (only)
  o Flagship - Remote Control System (RCS)
    • Windows, OSX, Linux,
    • Android, iOS, Blackberry, Windows Phones, Symbian
  o remote exploits (many 0days)
  o UEFI BIOS rootkit
  o remote injectors for ISP side

• civil right activist – “Enemy of the Internet”
  • Some of their customers - non-democratic countries
  • Using this tool against journalists and protesters
Microsoft releases critical out-of-band security patch for Windows

by Jason Murdock 21 Jul 2015

Microsoft has released an emergency out-of-band security fix for Windows, following the Patch Tuesday updates earlier this month.

The latest update (MS15-078) patches a critical flaw in how Windows Adobe Type Manager Library handles OpenType fonts. The fix is marked as 'critical' for all versions of Windows.
LEAK/HACK – STOLEN DATA

• July 5\textsuperscript{th} of 2015
• still no information who did it
• what was stolen = “almost” everything (400GB)
  • all source codes
    • full git repositories (53 repos.)
• 6 0day exploits
  • CVE-2015-\{5119, 2387, 5122, 5123, 2425, 2426\}
• release brochures, product documentation
• company emails
LEAK/HACK – Market of 0day

This article documents Hacking Team's third-party acquisition of zero-day (0day) vulnerabilities and exploits. The recent compromise of Hacking Team's email archive offers one of the first public case studies of the market for 0days. Because of its secretive nature, this market has been the source of endless debates on the ethics of its participants. The archive also offers insight into the capabilities and limits of offensive-intrusion software developers. As a private company, Hacking Team had to contend with the fact that many vendors would only sell directly to governments and would not work with them. As a result, their 0day providers tended to be small and unestablished. Some established exploit vendors, like VUPEN and COSEINC, did offer to sell Hacking Team exploits, but they were predominantly overpriced, second-rate, and not even 0day. As a result, Hacking Team was seriously exploit supply constrained because they had difficulty finding suppliers that they deemed reliable and reasonably priced. Their competitors, like Gamma International and NSO Group, prominently advertised their 0day capabilities, forcing Hacking Team to be defensive with prospective customers.

https://tsyrklevich.net/2015/07/22/hacking-team-0day-market/
INFRASTRUCTURE

1. HT RCS – RAT agent to monitor everything in all interesting platforms
2. Infection
   • Melting tool
   • Exploit Delivery Network (Windows / Android)
   • Remote Mobile Infection (vector-rim – crafted MMS)
   • Injection Proxy Appliance (vector-ipa)
     • Inject malicious contents
     • Melt on-the-fly
   • Offline infection (with bootable devices)
3. Control
   • proxy chain by Anonymizer
   • Fancy control panel for agents
EXPLOIT DELIVERY NETWORK
EXPLOIT DELIVERY NETWORK

• Separated systems for Windows and Android
• They were “one-shot” exploits – just a very limited time they were available
• Each customer (victims 😊) had a dedicated folder – URLs and place for logs
• httpd, URL rewrite, one file – download.php
• all resources had a .ini file – how it has to be handled
  • static or dynamic values,
  • how long could it be accessible
EXPLOIT DELIVERY NETWORK

1. http://...../docs/BR2u9Z/fwd

2. Expiry value will be updated

3. [valid]
   path = go.html
   headers[Content-Type] = text/html
   type = data

4. http://...../docs/BR2u9Z/tjPZnU

   [valid]
   path = ./stage4.js.py
   headers[Content-Type] = text/javascript
   type = exec
EXPLOIT DELIVERY NETWORK

http://exploit_delivery_network/docs/BR2u9z/fwd

fwd = fwd.ini ➔ go.html

Customer = victim ID (BR2u9z)

<table>
<thead>
<tr>
<th>Hits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hits</td>
<td>hits left</td>
</tr>
<tr>
<td>Expiry</td>
<td>reachable until this time</td>
</tr>
<tr>
<td>Useragent</td>
<td>User-Agent header must contain this string</td>
</tr>
</tbody>
</table>

The content of the go.html file was sent back as a result of the call

HTTP Redirection to a harmless website

[related] If the call was right all related files’ configs were updated – within 5 mins they were accessible
RCS FOR ANDROIDS
RCS agent for Android

• Install custom root service (ddf / rilcap) - instead of using ‘su’
• Modify permissions of APK
• Install itself as an administrator application
• hooking into the MediaServer system service to intercept all audio content – all calls (regardless of the app) can be motorized by this technique
• Traditional evidence gathering features
  • Take screenshot, monitoring clipboard, location tracking
  • Contact and messages for these apps
  • FaceBook, Viber, Skype, wechat, whatsapp, snapchat, gtalk, bbm, build in mail app & contacts
RCS Android root tool (ddf / rilcap)

Usage:

```java
shellFile = M.e("/system/bin/ddf");
oldShellFileBase = M.e("/system/bin/rilcap");
```

- **fb**: try to capture a screen snapshot
- **vol**: kill VOLD twice
- **reb**: reboot the phone
- **blr**: mount /system in READ_ONLY
- **blw**: mount /system in READ_WRITE
- **rt**: install the root shell in /system/bin/rilcap
- **ru**: remove the root shell from /system/bin/rilcap
- **rf**: `<mntpoint> <file>` - remove `<file>` from `<mntpoint>`
- **sd**: mount /sdcard
- **air**: check if the shell has root privileges
- **qzx**: "command" - execute the given commandline
- **fhc**: `<src> <dest>` - copy `<src>` to `<dest>`
- **fhs**: `<mntpoint> <src> <dest>` - copy `<src>` to `<dest>` on mountpoint `<mntpoint>`
- **fho**: `<user> <group> <file>` - chown `<file>` to `<user>:<group>`
- **pzm**: `<newmode> <file>` - chmod `<file>` to `<newmode>`
- **adm**: `<package name/receiver>`
- **qzs**: start a root shell
- **lid**: `<proc> <dest file>` - return process id for `<proc>` write it to `<dest file>`
- **ape**: `<content> <dest file>` - append text `<content>` to `<dest files>` if not yet present
- **srh**: `<content> <file>` - search for `<content>` in `<file>`
Exploit for Android
Exploit for Android

• Remote code execution (webview)
  • They joined 3 vulnerabilities to create this exploit
  • For code execution 4 stages
  • The most stages are encrypted or obfuscated
  • Information leakage vulnerability helped them to bypass ASLR
  • They used ROP gadgets to bypass NX

• Local root exploit
  • exynos exploit (Samsung)
  • CVE-2013-6282 - get_user and (2) put_user
  • CVE-2014-3153 - futex_requeue (TowelRoot)
Vulnerabilities they joined together (webview)

• **Information Leak (CVE-2011-1202)**
  - “The xsltGenerateIdFunction function in functions.c in libxslt 1.1.26 and earlier, as used in Google Chrome before **10.0.648.127** and other products, allows remote attackers to obtain potentially sensitive information about heap memory addresses via an XML document containing a call to the XSLT generate-id XPath function.”
  - “combined information leakage vulnerability CVE-2011-1202, to obtain the base address and then get libwebcore.so libc.so base address.”

• **Arbitrary Memory Read (CVE-2012-2825)**
  - "The XSL implementation in Google Chrome before **20.0.1132.43** allows remote attackers to cause a denial of service (incorrect read operation) via unspecified vectors.”

• **Heap-Buffer-overflow (CVE-2012-2871)**
  - "libxml2 2.9.0-rc1 and earlier, as used in Google Chrome before **21.0.1180.89**, does not properly support a cast of an unspecified variable during handling of XSL transforms, which allows remote attackers to cause a denial of service or possibly have unknown other impact via a crafted document, related to the _xmlNs data structure in include/libxml/tree.h.”

  - [https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2825](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2825)
  - [http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2871](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2871)
  - [http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-1202](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-1202)
HT WebView exploit

<table>
<thead>
<tr>
<th>Version</th>
<th>Codename</th>
<th>API</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Froyo</td>
<td>8</td>
<td>0.2%</td>
</tr>
<tr>
<td>2.3.3 - 2.3.7</td>
<td>Gingerbread</td>
<td>10</td>
<td>4.1%</td>
</tr>
<tr>
<td>4.0.3 - 4.0.4</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>3.7%</td>
</tr>
<tr>
<td>4.1.x</td>
<td>Jelly Bean</td>
<td>16</td>
<td>12.1%</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>17</td>
<td>15.2%</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>18</td>
<td>4.5%</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>19</td>
<td>39.2%</td>
</tr>
<tr>
<td>5.0</td>
<td>Lollipop</td>
<td>21</td>
<td>15.9%</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>22</td>
<td>5.1%</td>
</tr>
</tbody>
</table>

Data collected during a 7-day period ending on September 7, 2015.

35.5% are still vulnerable

<table>
<thead>
<tr>
<th>Android version</th>
<th>WebKit version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android 4.0.1</td>
<td>534.30</td>
</tr>
<tr>
<td>Android 4.0.2</td>
<td>534.30</td>
</tr>
<tr>
<td>Android 4.0.3</td>
<td>534.30</td>
</tr>
<tr>
<td>Android 4.0.4</td>
<td>534.30</td>
</tr>
<tr>
<td>Android 4.1.1</td>
<td>534.30</td>
</tr>
<tr>
<td>Android 4.1.2</td>
<td>534.30</td>
</tr>
<tr>
<td>Android 4.2</td>
<td>534.30</td>
</tr>
<tr>
<td>Android 4.2.1</td>
<td>534.30</td>
</tr>
<tr>
<td>Android 4.2.2</td>
<td>534.30</td>
</tr>
<tr>
<td>Android 4.3</td>
<td>534.30</td>
</tr>
</tbody>
</table>
HT WebView exploit

<table>
<thead>
<tr>
<th>Device</th>
<th>Version</th>
<th>R2L</th>
<th>L2R</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alcatel One Touch</td>
<td>4.1.1</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>CAT B15</td>
<td>4.1.2</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>HTC One</td>
<td>4.x</td>
<td>NO</td>
<td>?</td>
<td>(1)</td>
</tr>
<tr>
<td>LG G2</td>
<td>4.2.2</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>LG Nexus 4</td>
<td>4.2.2</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy Nexus</td>
<td>4.0.4</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy Nexus</td>
<td>4.3</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy Note</td>
<td>4.1.2</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy Note 2</td>
<td>4.1.1</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy S2</td>
<td>4.0.4</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy S3</td>
<td>4.3</td>
<td>YES</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy S3 Mini</td>
<td>4.1.1</td>
<td>YES</td>
<td>YES</td>
<td>(2)</td>
</tr>
<tr>
<td>Samsung Galaxy S4 Mini</td>
<td>4.2.2</td>
<td>NO</td>
<td>NO</td>
<td>(2)</td>
</tr>
<tr>
<td>Samsung Galaxy Tab 2 7.0</td>
<td>4.0.3</td>
<td>YES*</td>
<td>YES</td>
<td>(3)</td>
</tr>
<tr>
<td>Samsung Galaxy Tab 2 7.0</td>
<td>4.1.2</td>
<td>YES*</td>
<td>YES</td>
<td>(3)</td>
</tr>
<tr>
<td>Huawei Ascend Y530</td>
<td>4.3</td>
<td>YES</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

(1): Versions up to 4.4.3 are vulnerable but due to phone peculiarities the browser might not be exploitable
(2): This phone runs a patched version of the browser and is therefore not vulnerable
(3): Exploitation is not very reliable
WebView exploit

Parent process

WebView

Stage0: Preparing memory (4MB controlled memory space)

Stage1: mem. prep. – memory search (CVE-2012-2825)

Stage2: get base address of libwebcore.co (CVE-2011-1201) and (CVE-2012-2871)

Stage3: preparing stage4
- Loading module.so in memory
- ROP – mprotect()
- Build RCECall

Stage4: Call RCECall

module.so

exploit (ELF - root exploits)

installer.apk – scout agent

exploit (ELF - root exploits)

HT EDN Server

go.html + script.js

data.xml?id=2053320704

data.xml?id=2053320704&contentId=2053325124

module.so

stage4.js?trk=-213173581276

exploit

installer.apk

/system/bin/

ddf root shell
Plan A:
• Email with malicious link
• Click on it to trigger the exploit
Plan B:
• hijack network flow (free wifi, ISP)
• inject malicious content on-the-fly
• exploit any app which uses webview
HOW THEY FLEW UNDER THE RADAR
Code protection

- Source code obfuscation & code protection
  - Obfuscation on all levels
  - Own packer
  - Melt with legal app (all platforms)
  - VMProtect for Windows
  - ProGard for Android
  - ELF string obfuscator
- Other solutions
  - Different version of the tool (Scout, Soldier, Elite)
  - blacklisted applications

```c
unsigned char* deobfuscate(unsigned char *s) {
    unsigned char key, mod, len;
    int i, j;
    unsigned char* d;

    key = s[0];
    mod = s[1];
    len = s[2] ^ key ^ mod;

    d = (unsigned char *)malloc(len + 1);

    // zero terminate the string
    memset(d, 0x00, len + 1);

    for (i = 0, j = 3; i < len; i++, j++) {
        d[i] = s[j] ^ mod;
        d[i] = mod;
        d[i] = key;
    }

    d[len] = 0;
    return d;
}
```

```c
static unsigned char ptmx_device[] = "\x13\xfa\xe0\xcc\x8b\x8a\xa5\xcc\xa7\x9b\x82\x9f"; // "/dev/ptmx"
static unsigned char daemon_opt[] = "\x3d\xe4\xd1\x10\x10\xd9\xa4\xd8\xd0\xd2\xd3"; // "--daemon"
```
Avoiding Emulation (Windows)

• virtualization / sandbox detection
  (scout-win-master/core-scout-win32/antivm.cpp)
  • AntiVMWare() - VMWare
    • WMI query “SELECT SerialNumber FROM Win32_Bios”
  • AntiVBox() - VirtualBox
    • WMI query “SELECT DeviceId FROM Win32_PnPEntity”
    • Seeking for this value: “PCI\VEN_80EE&DEV_CAFE”
Cuckoo avoiding (Windows)

```c
VOID AntiCuckoo()
{
    LPDWORD pOld, pFake;

    pFake = (LPDWORD) malloc(4096*100);
    memset(pFake, 1, 4096*100);
    _asm
    {
        mov eax, fs:[0x44]       // save old value
        mov pOld, eax

        mov eax, pFake
        mov fs:[0x44], eax

        // this will not be logged nor executed.
        CreateThread(NULL, 0, (LPDWORD_START_ROUTINE) Sleep, (LPVOID) 1000, 0, NULL);
    }
    _asm
    {
        mov eax, pOld       // restore old value, not reached if cuckoo
        mov fs:[0x44], eax
    }
    free(pFake);
}
```

<table>
<thead>
<tr>
<th>Position</th>
<th>Length</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS:[0x44]</td>
<td>124</td>
<td>NT, Wine</td>
<td>Win32 client information (NT), user32 private data (Wine), 0x60 = LastError (Win95), 0x74 = LastError (WinME)</td>
</tr>
</tbody>
</table>

**cuckoomon.dll crash here**
Avoiding Emulation (Android)

```java
TelephonyManager tm = (TelephonyManager)
    Status.getAppContext().getSystemService(
        Context.TELEPHONY_SERVICE);

"0000000000000000" == tm.getDeviceId();
"3102600000000000" == tm.getSubscriberId();
"Android" == tm.getSimOperatorName();
"15555215554" == tm.getLine1Number();

"unknown" == Build.MANUFACTURER;
"generic" == Build.BRAND;
"generic" == Build.DEVICE;
"sdk" == Build.PRODUCT;
"test-keys" == Build.TAGS;
"test-keys" == Build.FINGERPRINT;

// This file does not exist on emulators
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```
AntiVirus testing environment

• They had a dedicated VM cluster for testing all their product against many AV products to be sure they are still undetectable
• Emulating critical events
  • Screenshots
  • Evidence gathering (email, messages, files)
  • Communications, hooks
• It was a QA process before release
## AntiVirus testing environment

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Hostname</th>
<th>Antivirus Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.100.111</td>
<td>win7kis</td>
<td>Kaspersky Antivirus 2013</td>
</tr>
<tr>
<td>192.168.100.112</td>
<td>win7panda</td>
<td>Panda Internet Security 2013</td>
</tr>
<tr>
<td>192.168.100.113</td>
<td>win7gdata</td>
<td>Gdata Internet Security 2013</td>
</tr>
<tr>
<td>192.168.100.114</td>
<td>win7trendm</td>
<td>Trend Micro Titanium</td>
</tr>
<tr>
<td>192.168.100.115</td>
<td>win7pctools</td>
<td>PCTools Internet Security 2013</td>
</tr>
<tr>
<td>192.168.100.115</td>
<td>win7norton</td>
<td>Norton Internet Security 2013</td>
</tr>
<tr>
<td>192.168.100.117</td>
<td>win7avira</td>
<td>Avira Internet Security 2013</td>
</tr>
<tr>
<td>192.168.100.118</td>
<td>win7drweb</td>
<td>DrWeb</td>
</tr>
<tr>
<td>192.168.100.119</td>
<td>win7secure</td>
<td>F-Secure Internet Security</td>
</tr>
<tr>
<td>192.168.100.120</td>
<td>win7eset</td>
<td>ESET Smart Security</td>
</tr>
<tr>
<td>192.168.100.121</td>
<td>win7avg</td>
<td>AVG Internet Security 2013</td>
</tr>
<tr>
<td>192.168.100.122</td>
<td>win7mcafee</td>
<td>McAfee Antivirus 2013</td>
</tr>
<tr>
<td>192.168.100.123</td>
<td>win7avast</td>
<td>Avast Internet Security 2013</td>
</tr>
<tr>
<td>192.168.100.124</td>
<td>win7bitdef</td>
<td>Bit Defender</td>
</tr>
<tr>
<td>192.168.100.125</td>
<td>win7sophos</td>
<td>Sophos EndUser Antivirus + Firewall</td>
</tr>
<tr>
<td>192.168.100.126</td>
<td>win7messentials</td>
<td>Microsoft Security Essential</td>
</tr>
<tr>
<td>192.168.100.127</td>
<td>win7zoneal</td>
<td>ZoneAlarm Antivirus + Firewall</td>
</tr>
<tr>
<td>192.168.100.128</td>
<td>win7ahnlab</td>
<td>Anlab</td>
</tr>
<tr>
<td>192.168.100.129</td>
<td>win7bytes</td>
<td>Malwarebytes Anti-Malware PRO</td>
</tr>
<tr>
<td>192.168.100.130</td>
<td>win7norman</td>
<td>Norman Antivirus</td>
</tr>
<tr>
<td>192.168.100.131</td>
<td>win7comodo</td>
<td>Comodo Internet Security Pro</td>
</tr>
<tr>
<td>192.168.100.132</td>
<td>win7emisoft</td>
<td>Emsi Soft</td>
</tr>
<tr>
<td>192.168.100.133</td>
<td>win7350cn</td>
<td>350.cn</td>
</tr>
<tr>
<td>192.168.100.134</td>
<td>win7risint</td>
<td>Risint</td>
</tr>
<tr>
<td>192.168.100.135</td>
<td>win7adaware</td>
<td>Adaware</td>
</tr>
<tr>
<td>192.168.100.136</td>
<td>win7kis14</td>
<td>Kaspersky Internet Security 2014</td>
</tr>
</tbody>
</table>

```python
parser = argparse.ArgumentParser(description='AVMonitor avtest')

parser.add_argument('action', choices=['scout', 'elite', 'internet', 'test', 'clean', 'pull'])
parser.add_argument('-o', '--platform', default='windows')
parser.add_argument('-b', '--backend')
parser.add_argument('-f', '--frontend')
parser.add_argument('-k', '--kind', choices=['silent', 'melt'])

parser.add_argument('--verbose', action='store_true', default=False, help='Verbose')

#parser.set_defaults(blacklist=blacklist)
#parser.add_defaults(platform_type=platform_type)

args = parser.parse_args()

winhostname = socket.gethostname().lower()

if "winxp" in winhostname:
    avname = winhostname.replace("winxp", ").lower()
else:
    avname = winhostname.replace("win7", ").lower()
    avname = winhostname.replace("win8", ").lower()

platform, mobile = ["android", "blackberry", "ios"]

soldierlist = ["bitdef,comodo,gdata,drweb,360cn,kis32,avg32,avg32,lobit32"].split(\',\')
blacklist = ["emisoft,sophos"].split(\',\')
```
CONCLUSION
Conclusion

• About HT and their stuff
  • it was well designed
    (leaked but never reverse engineered fully)
• The Android exploit (webview)
  • This is a quite good exploit and now it is freely available for anyone – for criminals as well
  • There are still millions of vulnerable devices
    (4.0 Ice Cream Sandwich - 4.3 Jelly Bean)
  • There are many devices in use which can not be updated
    • no official way to patch this vulnerability
References

- http://blog.azimuthsecurity.com/2013/02/re-visiting-exynos-memory-mapping-bug.html
- http://blog.nativeflow.com/the-futex-vulnerability
- https://www.4armed.com/blog/hacking-team-rcs-analysis-hacked-team/
- http://www.slideshare.net/jiahongfang5/mosec2015-jfang