Turla - development & operations

THE BIG PICTURE

ANDRZEJ DERESZOWSKI
Agenda

- Introduction
- Part I – Development
- Part II – Operations
- How to protect yourself
- Attribution ?
A bit of history from my perspective

- 2008 – Agent.BTZ – threat that hit Pentagon
- 2009 – Some Agent.BTZ incidents here and there
- 2011 – me, tecamac and other researchers get together to analyze certain complex threat
- Beginning 2013 – we started distributing our report and help others handle infections
- Beginning 2014 – a series of discoveries started by G-Data and BAE Systems
What has been published so far?

- G-DATA – “Uroburos – Highly complex espionage software with Russian roots” – Feb 2014
- Sourcefire VRT – “Snake Campaign: A few words about the Uroburos Rootkit” – Apr 2014
- F-Secure – “Anatomy of Turla Exploits” – May 2014
- Kernelmode.info threads – Jun 2014
- Symantec – “Turla:Spying tool targets governments and diplomats” – Aug 2014
Many publications – many names

- Currently there is a lot of confusion in naming scheme
- Final stage: Agent.BTZ/Snake/Turla/Uroburos/Carbon/Pfinet/Snark/Sengoku
- Reconnaissance stage: Epic/Tavdig/WipBot/WorldCupSec/TadjMakhal
- NOT all of them describe the same « product »
PART I
Development
What is Turla?

- Family of related sophisticated backdoor software
- Name comes from Microsoft detection signature – anagram of Ultra
  (Ultra3 was a name of the fake driver)
- All related by shared code
Code history

Agent.BTZ:
- Sengoku

Pfinet:
- Carbon
- Usermorde-centric Snake

Snake:
- Urouros
- Snark
- Kernelmode-centric Snake
## Features: summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Agent.BTZ</th>
<th>Pfinet</th>
<th>Snake</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>Hidden folder</td>
<td>Encrypted VFS</td>
<td>Encrypted VFS</td>
</tr>
<tr>
<td>Configuration</td>
<td>Hardcoded</td>
<td>Text file</td>
<td>Key-value store (queue)</td>
</tr>
<tr>
<td>Networking</td>
<td>Separate exes</td>
<td>Userland payloads</td>
<td>Kernel+userland</td>
</tr>
<tr>
<td>Incomming transports</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
VirtualBox exploit to load the driver

Uses a vulnerability in old (yet still signed!) VirtualBox driver to load its own driver.

Source: Sourcefire VRT

Source: F-Secure

Source: kernelmode.info

DSEFix - Defeating x64 Driver Signature Enforcement

DSEFix - Defeating x64 Driver Signature Enforcement

What is Driver Signature Enforcement? It is a security feature added to the NT6 which main purpose is to disallow loading drivers without digital signing, see http://msdn.microsoft.com/en-us/library/windows/hardware/dn653559(v=vs.85).aspx for more info. In reality this is yet another marketing bullshit from MS which ruined many freeware programs, and didn’t fixed anything in antimalware field - if malware authors really want to load their driver - they will do this. Mainstream crapware like ssdt hooking trash were dying even without this "improvements"
Udis86: on-the-fly manipulation of dissassembled code in live kernel

Source: deresz & tecamac

1.6 Force kernel mode

The rootkit will interact between user-mode and kernel-mode. Such interaction is delicate; in particular several system calls behave slightly differently when called from each mode. On such calls, the system traps the caller and uses the routine ExGetPreviousMode to determine whether the parameters are from a user-mode or kernel-mode source.

For example on Windows 2003, the code of ExGetPreviousMode is quite simple, it just gets and returns the PreviousMode value

```
.text:0044086C     mov    eax, large fs:124h
.text:00440872     mov    al, [eax+OD7h]
.text:00440878     ret
```

The tricky part is that the code may change across Windows versions. The current rootkit implements an easy solution for this issue: it disassembles ExGetPreviousMode, copies each instruction to the generated directive until the last return, and replaces mov al, [eax+OD7h] by mov [eax+OD7h] MODE where the D7 is dynamically computed according to the disassembly process and MODE is the requested mode (see Figure 1).

The disassembly is achieved with the support of Udis864
2.1 Hooking engine

The hooking engine is very pragmatic, it relies on a custom interrupt (C3h). When a location is to be hooked, the instructions that cover the first four bytes are relocated to a handler structure. A callback is also set in this structure and the structure is inserted in a table handler_table (9A75C) where it is associated to an ID.

The target location is hooked replacing the first four following bytes with `push h; int C3h` where h is a handler identifier.

Figure 1: Snake advanced assembly manipulation when relocating the code.
Encrypted VFS

- Implemented in Carbon and Snake
- CAST 128 encryption used
- Decryption/encryption implemented on low level by hooking sector processing mechanisms:
Encrypted VFS

Two volumes: permanent (mapped to a file on a real file system) and volatile storage

Encrypted container located in %windows%\$NtUninstallQ817473$\hotfix.dat

Source: deresz & tecamac
Configuration mechanism

- **Agent.BTZ:**
  - Config hardcoded in the user mode executables

- **Pfinet:**
  - Configuration file stored on the VFS in a flat file: `config.txt`
  - Transports implemented in user mode
  - Usermode payloads hardcoded in the rootkit body:
    - `cryptoapi.dll`
    - `inetpub.dll`
SNAKE:
- Uses « queue » file that contains configuration parameters in the form of key/value pairs
- « queue » file located on the VFS
- Queue contains:
  - Transports configuration
  - Userland payloads:
    - inj_snake_Win32.dll – a counterpart of a rootkit for userland
    - inj_services_Win32.dll
    - rkctl_Win32.dll
Modular transports – combined together

- `enc.frag.np`
- `domc.np`
- `frag.enc.reliable.doms.np`
Protocols to choose

- Datagram covert channels:
  - Raw layer 2 (Ethernet type 0x7FF)
  - Raw ICMP
  - Raw UDP - DNS
  - Raw IP

- Stream covert channels and activation triggers:
  - Raw TCP
  - HTTP: URL parameters of an HTTP request
  - HTTP: Hidden in HTTP headers
  - HTTP: Hidden in local part of the URL
  - SMTP: triggered by a recipient e-mail address
Examples of *incoming transports* – covert channels

SMTP covert channel – rootkit resides on the mail server of pwned-prg.com

HELO whatever.com
250 Hello whatever.com, I am glad to meet you
MAIL FROM: <you.bet@you.are.not>
250 OK
RCPT TO: <trueburger@pwned-org.com>
250 OK
354 End data with <CR><LF>,<CR><LF>
(commands>
.

Recipient user name must be 10 characters
Last two characters (in red) are the checksum calculated on the first 8:

\[
\begin{align*}
\text{username}[9] &= \text{sum} / 26 + 65 \\
\text{username}[10] &= 122 - \text{sum} \mod 26
\end{align*}
\]
Examples of *incoming transports*

HTTP covert channel – rootkit resides on the web server of pwned-prg.com

GET / HTTP/1.1
SomeHeader: \texttt{trueburgerYmFzZmFzY3J5YjE5MDczMTczCBzdHJpbmcKYmFz}
...
- Same signature calculated on the first 10 bytes of the header value
- Base 64 content that follows is decoded and XOR-ed back with raw buffer starting at offset 0
- First four bytes of the resulting content is a magic value, by default set to 0xDEADBEFF but changed by the initialization queue

**SNORT signatures – difficult to create !**
Possible to create Suricata sigs with the use of LUA
Big picture view of compromised network

INTERNET

SMTP dialog with trigger
Callback channel

mx.pwned-org.com

DMZ

Snake LAN channel

relay.pwned-org.com

Internal Segment 1

server1.pwned-org.com

Snake LAN channel

server2.pwned-org.com

Internal Segment 2

hr.pwned-org.com
Developers

- Vlad, gilg, urik
- Version control info present in some of the samples:

```
$Id: snake_config.c 5204 2007-01-04 10:28:19Z vlad $
$Id: mime64.c 12892 2010-06-24 14:31:59Z vlad $
$Id: event.c 14097 2010-11-01 14:46:27Z gilg $
$Id: named_mutex.c 15594 2011-03-18 08:04:09Z gilg $
$Id: nt.c 20719 2012-12-05 12:31:20Z gilg $
$Id: ntsystem.c 19662 2012-07-09 13:17:17Z gilg $
$Id: rw_lock.c 14516 2010-11-29 12:27:33Z gilg $
$Id: rk_bpf.c 14518 2010-11-29 12:28:30Z gilg $
$Id: t_status.c 14478 2010-11-27 12:41:22Z gilg $
$Id: ll_check.c 4477 2006-08-28 15:58:21Z vlad $
$Id: m2_to_b2_stub.c 4477 2006-08-28 15:58:21Z vlad $
$Id: m_frag.c 8715 2007-11-29 16:04:46Z urik $
```
We are so happy that most of "rootkit" code inside Turla was inspired by our program and features (this level of awareness is never seen anywhere in ITW malware since Rustock), so we decided to create something inspired by Turla in sort of exchange.

This proclaimed to be geverment sponsored lolkit in a reality is just a result, a compilation of several freelancers work (from both UA and RU) to create and support toolkit they sell for various kinds of espionage. For idiots from BAE Systems who are painting fake malware distribution diagrams in the Excel - No KGB or Kremlin here, guys, take a pill and relax with your prepaid propaganda.
PART II
Operations
Hmm, which group are we talking about?

- Not sure we can speak about one « Turla group »
- Turla is just one of the tools « Turla group(s) » uses
- There is however a lot of common things …
- While the tool itself is quite impressive, operators that are using it are sloppy …
Countries of interest

Source: BAE Systems
Publicly known victims

Pentagon computer networks attacked

The cyber-strike on key sites is thought to be from inside Russia.

November 28, 2008 | Julian E. Barnes | Barnes is a writer in our Washington bureau.

Finland admits it's suffered a massive cyber-attack. Is the same thing happening across Europe?

Sweden's National Defense Radio Establishment — its version of the U.S. National Security Agency signals-intelligence agency — told Reuters it had detected a number of attacks by Turla/Snake/Uroboros; officials in Finland also acknowledged having been attacked, but didn't confirm whether the culprit was related to Turla or Agent.BTZ. None of the investigations have turned up proof the malware is Russian, or that it is connected to official Russian intelligence services.

Uroboros rootkit: Belgian Foreign Ministry stricken

First information published on spy attack involving a high profile victim

Belgique : un ministere ciblé par un piratage informatique en lien avec l'Ukraine

Des dossiers et documents liés à la crise ukrainienne ont été piratés au sein du ministère belge des Affaires étrangères, ces derniers jours.
Turla: Staged operation

- Stage 0 – attack stage - infection vector
- Stage 1 – reconnaissance stage - initial backdoor
- Stage 2 – lateral movements
- Stage 3 – « access established » stage – TURLA deployed
- On each stage they can quit if it turns out that the « non-interesting » victim has been encountered
Stage 0: infection vector

- Watering holes (strategic web compromise)
  - “Adobe update” social engineering trick
  - Java exploits (CVE-2012-1723), Adobe Flash exploits (unknown) or Internet Explorer 6, 7, 8 exploits (unknown)
- Third party suppliers compromise
- No use of 0-day exploits (almost no)
Stage 0: Watering holes mechanism

Source: Kaspersky Lab
Stage 0: Watering hole panel

Admin panel

Enter password: [red]

Password it's wrong!

Stats | View Rule | View Exception | Black List | Clear Log+Exception | DELETE TASK
| TASK EDITOR | CONFIG EDITOR | DELETE successful count | Web-shell

Down

Total - 0
Interesting IP - 0

IE 6.0 IE 7.0 IE 8.0 Opera Firefox Safari Chrome Unknown

Total - 0
Interesting IP - 0


Up

(*) File is NOT writable -> /var/www/css/log/maxsuc.css
(*) File is NOT writable -> /var/www/css/task/task.css
(*) File is NOT writable -> /var/www/css/task/time.css

(*) File exists -> /var/www/css/task/debug.css
(*) check time -> 180

--------------------------- TASK ---------------------------

{*} Total - 0

---------------------------
<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Modify</th>
<th>Owner/Group</th>
<th>Permissions</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ log ]</td>
<td>dir</td>
<td>2014-03-06 02:18:12</td>
<td>root/root</td>
<td>drwxr-xr-x</td>
<td>R T</td>
</tr>
<tr>
<td>[ task ]</td>
<td>dir</td>
<td>2014-03-06 02:18:12</td>
<td>root/root</td>
<td>drwxr-xr-x</td>
<td>R T</td>
</tr>
<tr>
<td>.htaccess</td>
<td>161 B</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rwxr-xr-x</td>
<td>R T E D</td>
</tr>
<tr>
<td>ad.php</td>
<td>6.32 KB</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>d.php</td>
<td>144 B</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>drwxr-xr-x</td>
<td>R T</td>
</tr>
<tr>
<td>del.php</td>
<td>486 B</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>index.html</td>
<td>14 B</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>index.php</td>
<td>10 B</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>info.png</td>
<td>20 B</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>nojs.php</td>
<td>1.33 KB</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>of_ct.js</td>
<td>1.22 KB</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>pd_ar.js</td>
<td>33.65 KB</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>pd_fl.js</td>
<td>28.50 KB</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>pd_ja.js</td>
<td>81.36 KB</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>pd_qt.js</td>
<td>22.89 KB</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>pd_sh.js</td>
<td>14.59 KB</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>pd_sl.js</td>
<td>7.81 KB</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
<tr>
<td>rtx.php</td>
<td>608 B</td>
<td>2014-03-05 05:08:27</td>
<td>root/root</td>
<td>-rw-r--r--</td>
<td>R T E D</td>
</tr>
</tbody>
</table>
Stage 1: reconnaissance stage

- Initial backdoor dropped – WipBot/Epic/TavDig
- Simple backdoor with a handful of commands
- Has no code in common with any variant of Turla but exports functions with the same names: ModuleStart and ModuleStop
- Well described in Kaspersky Lab report: [The Epic Turla Operation](https://www.kaspersky.com/blogs/the-epic-turla-operation-solvin)
Stage 1: Some interesting tricks used in WipBot

Source: Trend Micro

- CVE-2013-3346 - zero day used together with a known exploit
- Sets ThreadHideFromDebugger – breaks debugging
- Creates a new process in suspended mode and maps the same section of memory twice, in two different processes
- SetWindowsLong API call to start a thread in the newly created process – breaks most malware sandboxes
- Jumps several times from one process to another
- Wipes out the PE section so that it is harder to rebuild the unpacked executable
Stage 2: lateral movements

- Stage 1 C&C servers are easy targets – for example, they can be caught in spear phishing e-mails and sandbox
- Stage 2 backdoor: So let’s replace this by a less known backdoor
- Go after Domain Admin credentials
- Further explore and compromise the network
Stage 3: Turla

- Network has been found interesting to explore long-term and exfiltrate
- Is fully compromised
- Turla dropped on chosen machines
- Usage of many other tools
- Some networks owned for years...
How to detect Turla?

- Not very easy task …
- One fun story to tell 😊

- Do not only rely on vendors - talk to your partner organizations
- Establish relationships and share information!
- IOCs exchange is good but not enough these days:
  - They are easy to change by the intruders
  - Separate samples and infras used for different victims
- Good Yara sigs and custom detection tools can help
- Check your third party suppliers – for intruders it’s a perfect way to get in
Divagations on attribution

- Development:
  - Vlad, giil, urik
  - “Transmittion”, “Password it’s wrong” etc.
  - Zagruzchik.dll

- Operations:
  - Geographic distribution of infections
  - Virustotal submission countries
  - $default_charset = 'Windows-1251';
Questions ?

deresz@gmail.com
@deresz666