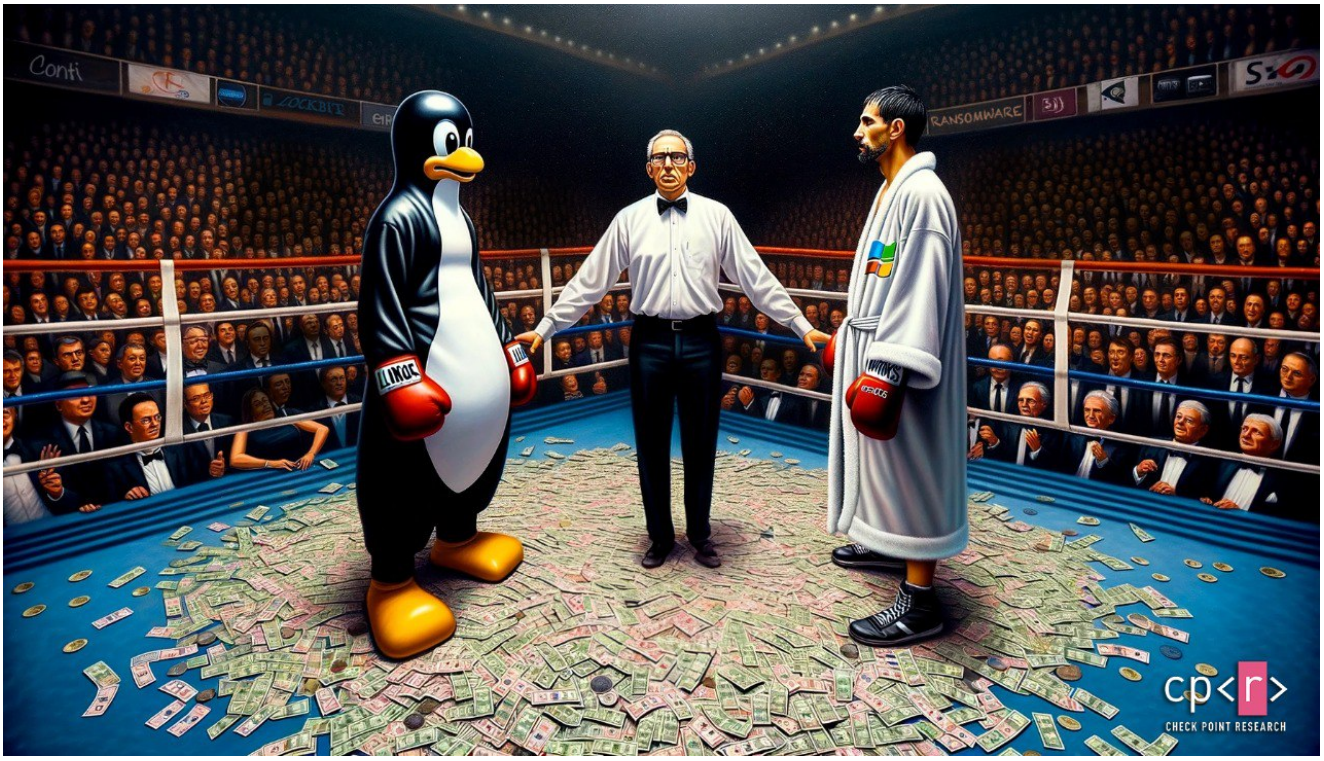
The Platform Matters: A Comparative Study on Linux and Windows Ransomware Attacks

research.checkpoint.com/2023/the-platform-matters-a-comparative-study-on-linux-and-windows-ransomware-attacks/

November 21, 2023



Research by: Marc Salinas Fernandez

Key Points

- Check Point Research (CPR) provides a case study of some of the most recent ransomware attacks targeting Linux systems and ESXi systems which have been increasing over the last few years.
- Although we have long been aware of similar ransomware threats in Windows environments, the versions targeting Linux are still relatively simpler.
- The release of Babuk's source code in 2021 has clearly facilitated the emergence of a multitude of ransomware families.
- Many of the families that target Linux heavily utilize the OpenSSL library along with ChaCha20/RSA and AES/RSA algorithms.

Introduction

During the last few months, we conducted a study of some of the top ransomware families (12 in total) that either directly developed ransomware for Linux systems or were developed in languages with a strong cross-platform component, such as Golang or Rust, thereby allowing them to be compiled for both Windows and Linux indiscriminately.

Our main objectives were to increase our understanding of the main motivations for developing ransomware targeting Linux instead of Windows systems, which historically have been the main target until now. We also tried to identify the main similarities and differences between the ransomware developed by these families and compare them to the ransomware developed for Microsoft systems.

Brief History

To compare the ransomware families developed for Linux and those targeting Windows, we first need to focus on the historical evolution of both systems.

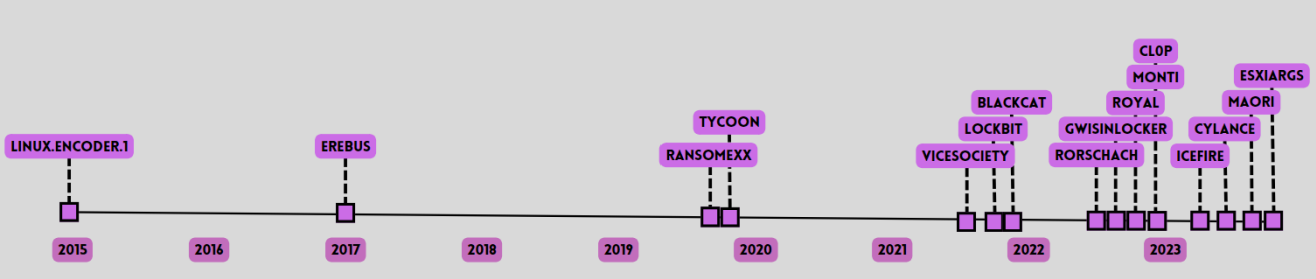


Figure 1 – Linux ransomware families.

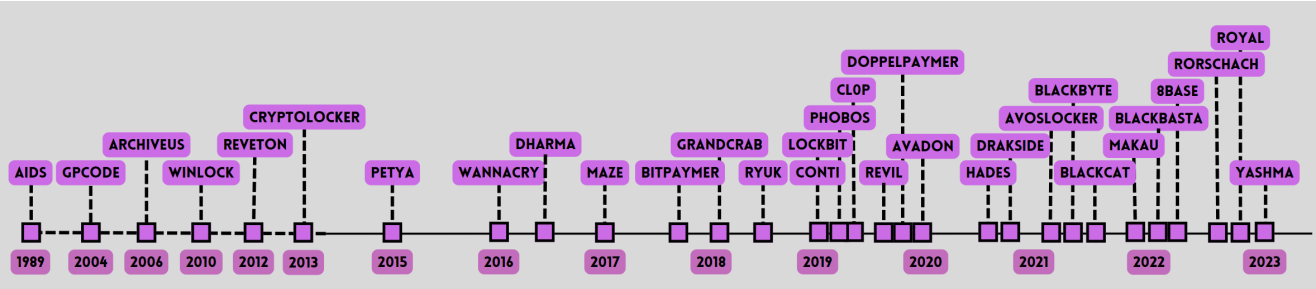


Figure 2 – Windows ransomware families.

To begin with, we should note that the first attributable ransomware sample (albeit in a very early stage) dates back to 1989. This threat, known as **AIDS**, was propagated through floppy disks and targeted Windows systems.

It was not until **GPCode** in 2004 that we started to see the first malware families that truly resembled what we are used to seeing today when we talk about ransomware. All these families focused on Windows environments, and soon the ransomware threat started to evolve, such as improved encryption schemes, as seen in **Archiveus** in 2006, or the appearance of **Reveton** in 2012 as the first RaaS.

It was not until 2015, with **Linux.Encoder.1**, that we began to see ransomware families focused specifically on Linux. By this time, these threats were already highly developed for Windows systems. Despite the level of maturity that these threats show in Windows, the reality is that in many aspects this has not translated into a direct transfer of all these capabilities to Linux. Instead, we have been seeing how these threats undergo the same stages of evolution in these other systems.

In fact, although there was already ransomware for Linux in 2015, it remained relatively insignificant until the last few years, when we began to see a huge proliferation of these threats. Starting in 2020 and continuing through to the present, we have begun to observe a worrying increase in attackers' interest in these systems, with the appearance of Linux versions of the major RaaS and cross-platform samples developed in languages such as Golang or Rust.

Technical overview

Of the families currently targeting Linux-based operating systems, we analyzed some of the most recent ones:

- Maori
- Cl0p
- Cylance
- Royal
- ViceSociety
- IceFire
- BlackCat
- ESXiArgs
- Rorschach
- Monti
- LockBit
- GwisinLocker

One of the first things we noticed in the samples we analyzed is the extent to which the tool itself is simplified in many cases, leaving only minimal capabilities and content within the binary, and in some cases reducing them to only the file encryption code. This leaves the sample very dependent on external configurations, scripts or command lines to configure its targets. One of the most notable examples is **Cl0p**, which only has the encryption capability, and the only parameter it supports is a path to encrypt.

In the ransomware family named "**ESXiArgs**" the binary itself does not even have the RSA public key embedded but needs the path to a file containing the key as a parameter so it can carry out the encryption. This sample doesn't even have the ability to encrypt a whole directory; the attacker has to iterate over every file with a script executing the encryptor. In

fact, the malware name was given due to the TTPs of the actors that use this malware, which is very oriented to this type of system, although the capabilities of the binary itself are totally generic.

Many of the Linux-oriented ransomwares have so little logic apart from the encryption capacity that detecting them can be challenging, as all their code is based on the same crypto code that many other legitimate applications may contain. A communication protocol with a server, the execution of some commands to prepare the system for the encryption, the ability to create some kind of persistence (found on many of the most active Windows families), or even an embedded configuration are, in many cases, anomalous elements that could help to enable more elaborate detections ofthe malware, but which do not exist in most of these ransomware families.

Of course, there are some exceptions such as **BlackCat**, which is a cross-platform sample and has Windows-specific functionalities such as deleting shadow copies or searching for shared folders. Or **GwisinLocker**, which has an embedded encrypted configuration that allows it to work without the need for parameters and act more independently.

The primary and most notable motivation is undoubtedly the special interest in **ESXi** virtualization systems. This makes a lot of sense, as by attacking these systems, the attackers can greatly impact multiple services and machines (all virtualized using this technology) by focusing only on this ESXi server instead of trying to pivot on several different computers and servers running Windows. This is probably why the vast majority of the Linux-targeting ransomware families, despite having very few capabilities apart from the encryption itself, tend to run specific commands aimed at interacting with ESXi systems, in particular:

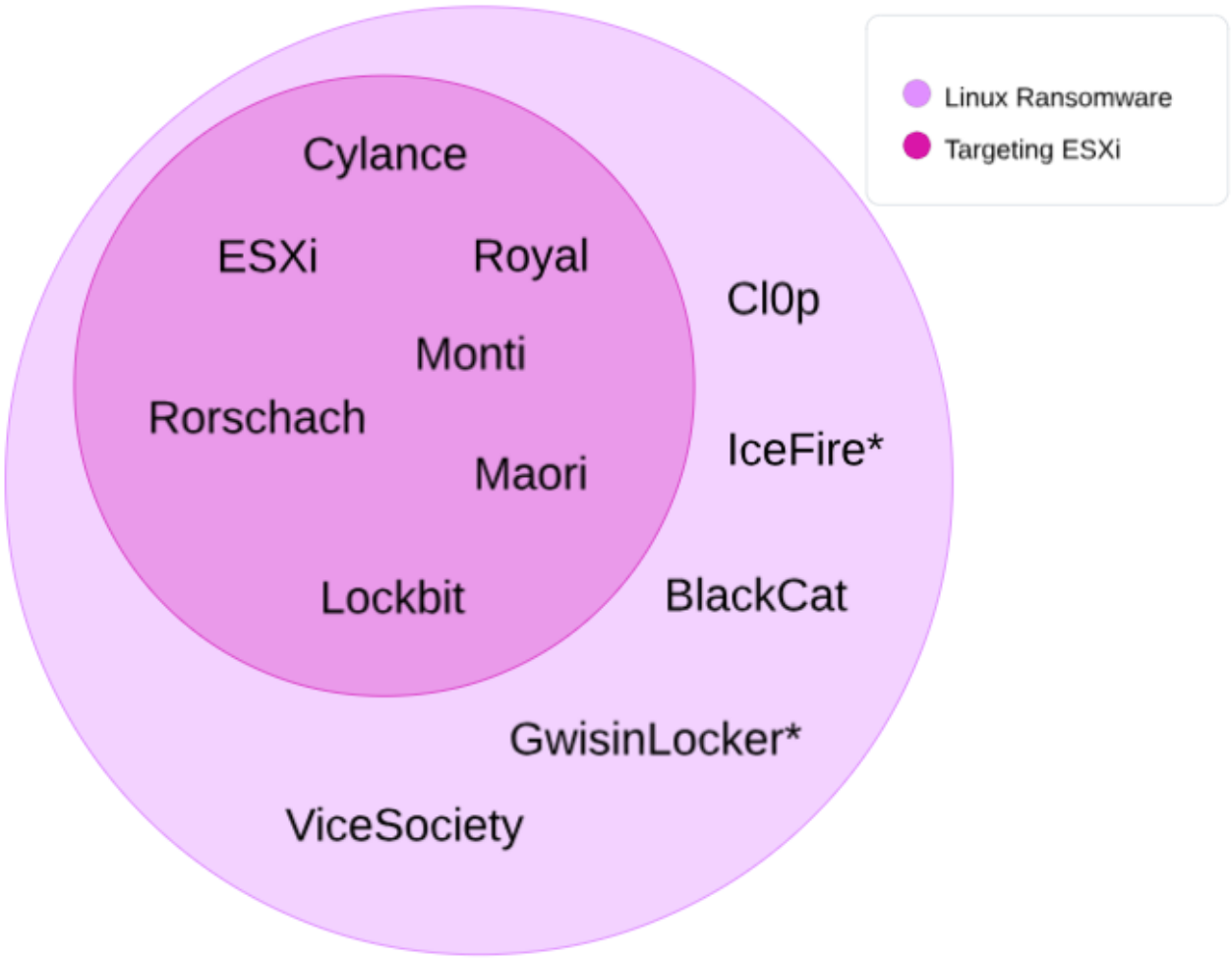


Figure 3 – Subset diagram on Linux ransomware families.
It is important to point out that since ESXi systems are not exactly the same as Linux systems, the different samples released contain the necessary libraries statically linked so that they can run independently on both systems. We have also found samples of the same family compiled specifically for each of the different systems.

A very common pattern in all Linux-centric families is that they tend to focus on specific technologies, which are linked mainly to the main infection path for this type of threat in these systems. Unlike what we are accustomed to in families that target Windows, such as **Ryuk** or **REvil** whose intrusions are often initiated through phishing campaigns to many users, one of the most common infection chains for Linux is exploiting a vulnerability in some exposed service of the victim's servers. This is also true for vulnerabilities in ESXi, but there are also other cases, such as **IceFire** which exploits a vulnerability in an IBM technology (CVE-2022-47986) or **Cl0p** whose Linux version has among its target directories several paths related to Oracle databases along with the generic ones of a Linux system.

Infection Vector

In the Windows environment, ransomware actors employ a wide range of infection vectors to breach systems. Many of the most aggressive Windows-targeting ransomware reach the victim's infrastructure via **phishing emails** containing malicious attachments (commonly using macros inside documents) or links. For example, **Emotet** was often the initial payload delivered, and the full infection of the victim's infrastructure ended with the deployment of a **Ryuk** or **Sodinokibi** sample.

Along with phishing emails, in the past the use of **exploit kits** like Rig and Magnitude to exploit vulnerabilities in software such as browsers or plugins led to ransomware execution (much less common these days).

Another common infection vector is the exploitation or brute-forcing of **Remote Desktop Protocol** (RDP) servers exposed to the internet.

If we try to perform this same analysis with respect to ransomware families developed for Linux systems, we can quickly see how the scenario changes. Many of these systems are deployed for the purpose of running services that are exposed to the Internet, services in which vulnerabilities are eventually found to be particularly critical, as they can allow access to an organization's network.

The **exploitation of vulnerabilities found on exposed services** is one of ransomware's main means of infection. It is worth noting that the Kill Chain in these cases often involves the deployment of a Webshell that ends up being the tool that initially allows them to access and take control of the server in question.

Gaining access with **stolen credentials**, for example, using SSH, is another growing area. Credentials are often stolen as the result of leaks caused by other malware infections or as a result of lateral movement by the same infection that involves the entire network of Windows systems.

In these cases, detection within Linux systems is very complicated, as attackers often use internal system accounts and legitimate tools to access systems instead of backdoors, with a very similar impact as the use of LoLbins on Windows systems.

Another common entry to Linux systems is similar to what happens in Windows systems with the RDP service, the scanning of different exposed services, and the subsequent **brute force attacks** trying to gain access to the servers through weak credentials. This is a much "noisier" technique, but still effective, and is becoming difficult to identify since the access is through legitimate credentials obtained from the exposed service itself.

It is interesting to note that, if we focus on all the common infection vectors for Linux servers, each pattern targets exposed servers and critical services. Once again, we can see that the Linux-targeting ransomware attacks are much more focused on organizations and companies than on general users.

Code Reuse

As in many malware families of other types of threats (like Mirai or Quasar), as soon as the source code of a successful threat is published, other opportunistic groups rapidly appear and try to take advantage of this code to create their own tools through small (and in some cases not so small) modifications. In the case of Linux ransomware, the most notable is Babuk ransomware, of which we can find, among many others, the **Cylance** or **Rorschach** samples.

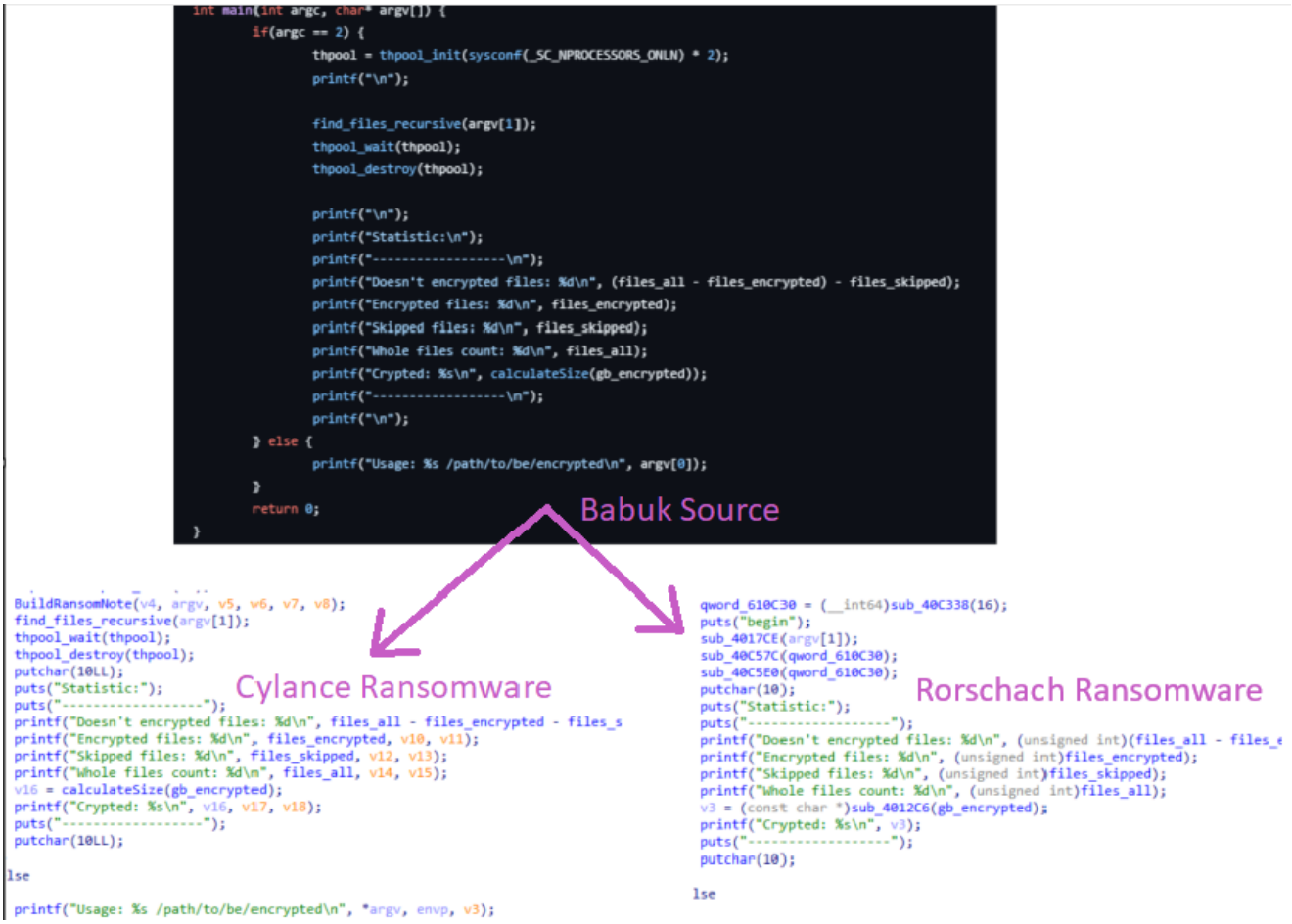


Figure 4 – Code overlaps on Babuk based families

Along with the problem that this fact implies, since many actors with fewer resources or technical capacity quickly get a functional tool, we have at least the advantage that, in many cases, the detection of the initial tool can allow us to detect all the sub-families that may appear from this source code.

Persistence in the system

Persistence is not as big a factor in ransomware as it is in other types of threats because once the victim's files and directories are encrypted, another execution in the same system is largely meaningless. However, the kill chain of this type of malware has indeed evolved greatly, especially in Windows environments, as the aim is not to encrypt a single computer but to spread to others. In most cases, the attackers' objective is to compromise the entire

infrastructure bit by bit. Once they have taken control, the entire AD forest is encrypted, for example, by means of a GPO, or the most critical computers are encrypted to increase the attackers' chances of receiving the ransom payment.

Prior to the execution of the ransomware, a whole series of threats and tools that allow the attackers to access the systems is executed. These do require persistence as the compromise of the entire infrastructure can take a long time. However, while this is the most common scenario, there are cases where the threat itself has its own ability to establish persistence.

In Windows environments, ransomware achieves persistence through various means. The most notable examples include **Registry Manipulation**, like the case of **WannaCry** and **Ryuk**, that ensures their payloads execute during system startup, along with the use of **Scheduled Tasks**, like the case of many threat actors behind **Sodinokibi** (**REvil**), which leveraged the Windows Task Scheduler to create malicious tasks, ensuring ransomware execution at regular intervals.

Another common way for gaining persistence on Windows systems is **Service Creation,** which is the most restrictive as it requires administrator permissions on the victim's computer but is one of the most commonly used in more advanced stages of infection in which the attackers already obtained the necessary credentials and have some control over the infrastructure.

In ESXi and Linux systems it is much less common to see ransomware employing many of the known methods for persistence usually exploited by other kinds of threats. After access, the vulnerable server is directly encrypted and, in many cases, such as **Lockbit**, **ESXiArgs**, **BlackCat,** or **Gwisin**, the malware has the ability to self-delete after execution.

The deployment of Webshells as part of the infection process should also be considered as maintaining persistence. The Webshells act as backdoors and allow the actors to maintain access to these servers after reboots or changes of any kind. In scenarios where servers are accessed through lateral movement during a more complex compromise, persistence in this case is mostly reduced to the creation of **user accounts or the exfiltration of original server credentials**, which allows the attackers to maintain access through legitimate services such as SSH.

Finally, given the clear evolution of incidents related to Linux systems compared to Windows systems, sooner or later the deployment of backdoors such as Merlin or Poseidon , like what is now happening on Windows with Cobalt Strike, will become more common. Therefore, attackers need to take advantage of techniques more similar to those we see in Windows systems, such as **Cron Jobs** (the equivalent of the Windows Task Scheduler) or executions such as **Daemons** (equivalents of Windows Services), to gain persistence.

Primary Objectives

In the area of victim typology and targets of Linux-oriented ransomware, we see some of the biggest differences with respect to their Windows counterparts. First of all, we must take into account the context in which each of these systems is found. Windows is more prevalent in personal computers for individuals and in user workstations for most organizations. However, in the field of servers, the situation is not so clear, especially for certain types of deployments using Linux, which is often the only effective option.

This means that just as we can easily find multiple ransomware families for Windows focused on individuals and endpoints, this is a lot less common for Linux systems. Ransomware targeting Linux is much more clearly oriented towards exposed servers or servers on the internal network that is accessed by pivoting from infections initiated on Windows machines.

As a result, Linux ransomware **is clearly aimed at medium and large organizations** compared to Windows threats, which are much more general in nature.

In the same way, the internal structure of both systems also causes differences in how attackers approach the selection of folders and files to encrypt. We can find listings in many Linux-oriented samples that aim to avoid directories such as /boot, /etc, or /sys that could cause the system to become corrupted, just as we are used to seeing Windows malware avoiding the or directories.

In the absence of a configuration within Windows malware that contains targets, it indiscriminately traverses all the system disks. In Linux malware, it is much more common to find threats **completely dependent** on a parameter or configuration that provides one or more target directories, **without which the threat does not execute**. Some examples of this include **Royal**, **Monti**, **Cylance** or **Lockbit**.

The difference in the management of extensions in Linux and Windows also generates somewhat curious behavior from attackers. One such case is **Cl0p** that uses the characters "." in an attempt to differentiate files from folders. This is very effective in Windows, but that does not necessarily work well in Linux given the little relevance that extensions have in this system.

```
find("/opt", "*.*");
find("/u01", "*.*");
find("/u02", "*.*");
find("/u03", "*.*");
find("/u04", "*.*");
find("/home", "*.*");
find("/root", "*.*");
return sleep(-1);
```

Figure 5 – Usage of "." for extensions by Cl0p ransomware

In any case, although many of them are completely dependent on a parameter at least indicating a path to encrypt, it is not present in all families, and for other samples, it is a remarkable fact that apart from very specific cases for paths related to ESXi or **CL0p** with Oracle paths "`/u01 /u02 /u03 /u04`", the `/home` and `/root` folders are the most recurrent in configurations, followed by `/opt` that appears in certain cases.

Exfiltration

In Linux, exfiltration is usually connected to the infection vector. In cases where the infection occurred using stolen credentials, access is generally gained **using legitimate tools** such as the SSH service, which at the same time allows all types of information to be extracted from the servers without the need to deploy other tools.

Likewise, in many scenarios in which the exploitation of a vulnerability to gain access to servers is linked to the deployment of a Webshell, something similar occurs as the majority of **Webshells**, along with the ability to execute Linux commands, **have their own capabilities for uploading and downloading files** from the victim server. Therefore, they are often also used as a tool to carry out exfiltration.

In Windows systems, the use of RDP, WinSCP, or RClone could be similar to the use of SSH or other legitimate tools such as Curl or Wget in Linux. Something very common in Windows, and not so common in Linux, is the use of more complex threats such as past threats Trickbot or Emotet, or the use of CobaltStrike or other post-exploitation frameworks for this purpose. As we suggested in our discussion of persistence, it is very likely that as ransomware samples mature, the TTPs of the actors will also mature, and we will end up seeing this scenario in Linux with the use of backdoors such as Merlin or Poseidon.

It is worth noting that this aspect is becoming highly relevant.Ransomware **groups have been exploiting double extortion** for some time now since they not only hijack files but also threaten to expose their victims' sensitive information on their leak sites. In fact, several prominent groups, such as **Cl0p, have already carried out campaigns in which they have directly skipped the encryption tool** to focus solely on the theft of information for subsequent extortion.

Impact on the system

During a ransomware incident, one of the critical points, both at the detection and forensic level, is the impact of the attackers on the system beyond the encryption itself. In Windows environments, we are very used to tight monitoring of commands aimed at deleting ShadowCopies, disabling backups in general, and attempting to disable or bypass security tools. The execution of commands aimed at shutting down target services, such as databases, is relatively common as well, as this allows the threat to encrypt most of the critical files, thereby increasing the pressure on the victim to pay the ransom.

In Linux systems, the concern for backups as well as the shutdown of security tools is not yet as common as it is with Windows. However, we can find some elements that impact the system that can help with early detection if proper monitoring is maintained. The first example, which is also common in Windows environments, is a Mutex, created by many threats before starting the encryption to avoid simultaneous executions that can corrupt files without the possibility of return. In the same way that generating a certain specific mutex in Windows acts as a "vaccine" for some families, in Linux, we have samples such as **Lockbit**, which by default generates a file called /tmp/locker.pid that, in case it is already in the system at the time of execution, causes the immediate termination of the process (regardless of whether the process that previously generated it was the ransomware itself).

Similar to what happens in Windows, some families generate less repetitive files, as in the case of **Gwisin**, whose generated Mutex file is much more random: /tmp/.66486f04-bf24-4f5e-ae16-0af0fdb3d8fe.

A much simpler and less effective version when it comes to detection are the log files. It is not uncommon to find samples from real campaigns that, during encryption, generate files with debug information, such as **HelloKitty** ransomware or **Monti**, which generate work.log and result.txt files respectively, with information on their execution and encryption, and whose internal strings are very characteristic of both families. However, it should be noted that the existence of these files does not prevent their execution in any case, as occurs with Mutex.

With respect to the execution of commands that can be monitored, the only really noteworthy case is the one concerning virtualization on ESXi systems. As we discussed previously, most Linux-oriented ransomware samples have an ESXi version or are compiled in such a way that they are directly compatible. This is why obtaining the list of running machines as well as the ability to stop them to allow encryption is really common in these samples. Some examples of this include the **Royal** Ransomware:

```
execlp("/bin/sh", "/bin/sh", "-c", "esxcli vm process list > list", 0LL);
sprintf(v0, "esxcli vm process kill --type=hard --world-id=%s", dest);
```

Figure 6 – Esxi commands embedded on Royal ransomware**Monti** Ransomware:

```
execlp("esxcli", "esxcli", "vm", "process", "kill", "--type=hard", s, 0LL);
execlp("esxcli", "esxcli", "vm", "process", "list", 0LL);
```

Figure 7 – Esxi commands embedded on Monti ransomware
or **Gwisin** and **BlackCat** ransomwares:

```
esxcli --formatter=csv --format-param=fields=="DisplayName,WorldID" vm process list
esxcli vm process kill --type=force --world-id=
esxcli --formatter=csv --format-param=fields=="WorldID,DisplayName" vm process list
esxcli vm process kill --type=force --world-id=
```

The monitoring of this type of execution can be of real interest, as if they occur, it is moments before the encryption, which allows us to anticipate the encryption, and possibly detect and act upon it in time.

Cryptographic Scheme

Finally, we have the core part of this type of threat, i.e., the crypto that comes into play for each ransomware threat. In Windows, we are very used to seeing how in some cases, like **Conti**, this part is delegated to the Windows APIs themselves. In many others, the malware uses many different crypto libraries like CryptoPP (e.g., PYSA Ransomware), mbedtls (e.g., Petya) and libgcrypt (e.g., Moneybird).

Among the samples for Linux systems, it is much simpler as the use of the OpenSSL library to perform all crypto tasks predominates in almost half of the samples. In fact, several of the most well-known families have this library statically linked in the binary itself, representing more than 50% of the threat code. There are still some edge cases where the malware is developed in Golang or Rust, where the native libraries/modules for each language predominate.

In terms of algorithms, on Windows, it gets a bit more difficult to observe patterns since there are many different algorithms used among the huge variety of known families, while ChaCha20 and RSA slightly predominate over the rest. When more uncommon libraries or algorithms are used, it ends up in design flaws on the threat, with its consequent public decryptor.

Unsurprisingly, a smaller number of variants can be found in the Linux world. The majority of these samples primarily rely on AES for encryption, with ChaCha20 being the most common alternative in several families. As for asymmetric encryption algorithms, RSA takes precedence in the vast majority of cases, occupying a secondary role.

As in all the above points, there are exceptions. **ESXiArgs** employs Sosemanuk for symmetric encryption, while the “smartest” of them, **CI0p**, employs RC4 with an embedded key at the point where asymmetric encryption is typically utilized. This approach renders file decryption trivial without the need for payment.

At the end of the day, threat actors, especially in this field, prioritize efficiency because the faster the threat is able to cover all the target files, the less options are available for defense. Reliability is the second consideration, and therefore they use robust libraries and algorithms to reduce the number of design flaws that may allow security researchers to break their encryption. These two factors cause the different actors to create relatively uniform tools, which helps us gain insight into the tools and priorities used, which in turn enables us to more easily detect this type of threat.

Conclusions

Our analysis of various Linux-targeting ransomware families reveals an interesting trend towards simplification, where their core functionalities are often reduced to just basic encryption processes, thereby leaving the rest of the work to scripts and legitimate system tools. This minimalist approach not only renders them heavily reliant on external configurations and scripts but also makes them more difficult to detect.

Our research also showed some of the distinctive strategies among ransomware families, with a clear focus on ESXi systems but with other technologies too. The ransomware main entry vectors are vulnerabilities in exposed services, which in some cases are precisely the most relevant services and, therefore, the main targets for this type of threat.

Comparing the ransomware encryption techniques between Windows systems and Linux, the malware families that target Linux favor OpenSSL as the main library used and AES as a common encryption cornerstone, with RSA serving as the primary asymmetric choice. All of this provides a relative uniformity of tools among the different threat actors.

Check Point customers remain protected against the threats covered by this research while using Check Point Harmony Endpoint , and Threat Emulation – which provide comprehensive coverage of attack tactics and file-types.

- Ransomware.Wins.HelloKitty.ta.D
- Ransomware.Wins.GwisinLocker.ta.A
- Ransomware.Wins.Clop.ta.I
- Ransomware.Wins.Royal.ta.B
- Ransomware.Wins.IceFire.ta.A
- Ransomware.Wins.Monti.ta.A
- Ransomware.Wins.ESXi.ta.B
- Ransomware.Wins.Babuk.ta.A
- Ransomware.Wins.LockBit.ta.AK
- Ransomware.Wins.BlackCat.ta.M
- Ransomware_Linux_BlackCat_A, Ransomware_Linux_BlackCat_B
- Ransomware_Linux_Maori_A, Ransomware_Linux_Maori_B
- Ransomware_Linux_Clop_A, Ransomware_Linux_Clop_B
- Ransomware_Linux_Cylance_A, Ransomware_Linux_Cylance_B
- Ransomware_Linux_Royal_A, Ransomware_Linux_Royal_B
- Ransomware_Linux_ViceSociety_A, Ransomware_Linux_ViceSociety_B
- Ransomware_Linux_IceFire_A, Ransomware_Linux_IceFire_B
- Ransomware_Linux_Esxiargs_A, Ransomware_Linux_Esxiargs_B
- Ransomware_Linux_Monti_A, Ransomware_Linux_Monti_B
- Ransomware_Linux_Lockbit_E, Ransomware_Linux_Lockbit_F
- Ransomware_Linux_GwisinLocker_A, Ransomware_Linux_GwisinLocker_B

- Ransomware_Linux_Babuk_A, Ransomware_Linux_Babuk_B
- Ransomware_Linux_HelloKitty_C, Ransomware_Linux_HelloKitty_D

Yara rules

```
rule linux_Babuk_ransomware {
    meta:
        author = "Marc Salinas @ CheckPoint Research"
        description = "Detects samples of the Linux ransomware family Babuk"
        malware_family = "Babuk"
        date = "09/08/2023"

        hash1 = "b711579e33b0df2143c7cb61246233c7f9b4d53db6a048427a58c0295d8daf1c"
        hash1 = "d1ba6260e2c6bf82be1d6815e19a1128aa0880f162a0691f667061c8fe8f1b2c"

    strings:
        $str1 = "Statistic:"
        $str2 = "Encrypted files: %d"
        $str3 = "Usage: %s /path/to/be/encrypted"

        $bablock1 = ".x1x2x3"
        $bablock2 = "/_r_e_a_d_m_e.txt"

        $cylance1 = ".Cylance"
        $cylance2 = "CYLANCE_README.txt"

        $orig1 = "How To Restore Your Files.txt"
        $orig2 = ".babyk"
    condition:
        uint32(0) == 0x464c457f and (all of ($str*) or all of ($cylance*) or all of
($bablock*) or all of ($orig*))
}

rule linux_ESXi_ransomware {
    meta:
        author = "Marc Salinas @ CheckPoint Research"
        description = "Detects samples of the Linux ransomware family ESXi"
        malware_family = "ESXi"
        date = "09/08/2023"

        hash1 = "11b1b2375d9d840912cfd1f0d0d04d93ed0cddb0ae4ddb550a5b62cd044d6b66"

    strings:
        $usage = "usage: encrypt <public_key> <file_to_encrypt> [<enc_step>]
[<enc_size>] [<file_size>]"

        $coms1 = "init_libssl returned %d\n"
        $coms2 = "encrypt_file"
        $coms3 = "encrypt_simple"
        $coms4 = "lseek [start]"

        $cde1 = {48 8B 85 80 FD FF FF 48 01 85 50 FF FF FF 48 8B 8D 38 FF FF FF C7 85
28 FD FF FF 67 66 66 66 C7 85 2C FD FF FF 66 66 66 66 48 8B 85 28 FD FF FF 48 F7 E9
48 C1 FA 02 48 89 C8 48 C1 F8 3F 48 89 D3 48 29 C3 48 89 9D 40 FD FF FF 48 8B 85 40
FD FF FF 48 C1 E0 02 48 03 85 40 FD FF FF 48 01 C0 48 89 CA 48 29 C2 48 89 95 40 FD
FF FF 48 83 BD 40 FD FF FF 00}
        $cde2 = {48 8B 85 30 FD FF FF 48 D1 E8 48 8B 95 30 FD FF FF 83 E2 01 48 09 D0
```

```
F2 48 0F 2A C0 66 0F 28 C8 F2 0F 58 C8 F2 0F 11 8D 48 FD FF FF}
        $cde3 = {F2 0F 10 05 15 6F 00 00 F2 0F 59 85 48 FD FF FF F2 0F 11 85 28 FF FF
FF 48 8B 85 48 FF FF FF 48 89 85 50 FD FF FF 48 83 BD 50 FD FF FF 00}
    condition:
        uint32(0) == 0x464c457f and ( $usage or 3 of ($coms*) or 1 of ($cde*) )
}

rule linux_Monti_ransomware {
    meta:
        author = "Marc Salinas @ CheckPoint Research"
        description = "Detects samples of the Linux ransomware family Monti"
        malware_family = "Monti"
        date = "09/08/2023"

        hash1 = "edfe81babf50c2506853fd8375f1be0b7bebbefb2e5e9a33eff95ec23e867de1"
    strings:
        $str1 = "Total encrypted: %s\n"
        $str2 = "Encrypting %s\n"
        $str3 = "Cannot rename file %s\n"
        $str4 = "fork() error."

        $cde = {55 48 89 E5 48 83 EC 50 48 89 7D B8 48 89 75 B0 48 C7 45 C0 7F 44 4E
00 48 C7 45 C8 81 44 4E 00 48 C7 45 D0 84 44 4E 00 48 C7 45 D8 87 44 4E 00 48 C7 45
E0 8A 44 4E 00 C6 45 F3 05 C7 45 F4 00 00 00 00 48 8B 45 B8 48 85 C0}
    condition:
        uint32(0) == 0x464c457f and ( $cde or all of ($str*))
}

rule linux_IceFire_ransomware {
    meta:
        author = "Marc Salinas @ CheckPoint Research"
        description = "Detects samples of the Linux ransomware family IceFire"
        malware_family = "IceFire"
        date = "09/08/2023"

        hash1 = "e9cc7fdfa3cf40ff9c3db0248a79f4817b170f2660aa2b2ed6c551eae1c38e0b"
    strings:
        $str1 = "iFire.pid"
        $str2 = "-----BEGIN RSA PUBLIC KEY-----"
        $str3 = "ComSpec=C:\\Windows\\syste"
        $str4 = "./boot./dev./etc./lib./proc./srv./sys./usr./var./run"
        $str5 = "Do not try to recover files yourself"
    condition:
        uint32(0) == 0x464c457f and  3 of ($str*)
}

rule linux_Royal_ransomware {
    meta:
        author = "Marc Salinas @ CheckPoint Research"
        description = "Detects samples of the Linux ransomware family Royal"
        malware_family = "Royal"
        date = "09/08/2023"
```

```
        hash1 = "b57e5f0c857e807a03770feb4d3aa254d2c4c8c8d9e08687796be30e2093286c"

    strings:
        $str1 = "Testing RSA encryption"
        $str2 = "-vmonly"
        $str3 = "Most likely what happened was that you decided to save some money"

        $cde1 = {48 8D 85 30 FF FF FF BA 90 00 00 00 BE 00 00 00 00 48 89 C7 ?? ?? ??
?? ?? 48 8D 85 30 FF FF FF 48 89 C6 BF E8 0D 58 00 ?? ?? ?? ?? ?? 48 8B 85 60 FF FF
FF 48 85 C0}
        $cde2 = {48 8B 85 60 FF FF FF 48 89 C2 48 8B 4D D0 8B 45 CC 48 89 CE 89 C7 ??
?? ?? ?? ?? 83 F0 01 84 C0}
        $cde3 = {48 8D 85 30 FA FF FF 41 B8 00 00 00 00 48 89 C1 BA DD 0D 58 00 BE E0
0D 58 00 BF E0 0D 58 00 B8 00 00 00 00 ?? ?? ?? ?? ?? BF 00 00 00 00 }
    condition:
        uint32(0) == 0x464c457f and ( 2 of ($str*) or 1 of ($cde*))
}

rule linux_BlackCat_ransomware {
    meta:
        author = "Marc Salinas @ CheckPoint Research"
        description = "Detects samples of the Linux ransomware family BlackCat"
        malware_family = "BlackCat"
        date = "09/08/2023"

        hash1 = "e2dcd1eaf59e7e10b9dfeedc6f2b0678efac7907f17ee8b4e8791c39c1fbaa58"

    strings:
        $str1 = "no-vm-kill-names"
        $str2 = "safeboot-network"
        $str3 = "NO_VM_KILL_NAMES"
        $str4 = "Preparing Logger"
        $str5 = "already borrowed"
        $str6 = "/cargo/registry/src/github.com"
    condition:
        uint32(0) == 0x464c457f and 4 of ($str*)
}

rule linux_HelloKitty_ransomware {
    meta:
        author = "Marc Salinas @ CheckPoint Research"
        description = "Detects samples of the Linux ransomware family HelloKitty"
        malware_family = "HelloKitty"
        date = "09/08/2023"

        hash1 = "754f2022b72da704eb8636610c6d2ffcbdae9e8740555030a07c8c147387a537"

    strings:
        $str1 = "cr:%d f:%s\n"
        $str2 = "cr:%d l:%d f:%s\n"
        $str3 = "Done:%s file size:%lu crypt size:%lu \n "
```

```
        $str4 = "All your important documents, photos, databases were stolen"
        $str5 = ".README_TO_RESTORE"
        $str6 = "libcrypto.so not found \n try to find manual and  make link to
libcrypto.so \n"
        $str7 = "Usage:%s [-m (10-20-25-33-50) ] Start Path \n"
        $str8 = "Error InitAPI !!!\nExit\n"
    condition:
        uint32(0) == 0x464c457f and 4 of ($str*)
}

rule linux_Lockbit_ransomware {
    meta:
        author = "Marc Salinas @ CheckPoint Research"
        description = "Detects samples of the Linux ransomware family Lockbit"
        malware_family = "Lockbit"
        date = "09/08/2023"

        hash1 = "472836ed669d3927d50055e801048696375b37fce03b2f046e3e1039fb88e048"
    strings:
        $cde1 = {31 FF 41 BE 01 00 00 00 ?? ?? ?? ?? ?? 4C 89 E7 48 89 44 24 18 ?? ??
?? ?? ?? BA F1 B0 64 00 48 89 C1 BE 14 00 00 00 48 89 E7 ?? ?? ?? ?? ?? ?? ?? ?? ??
?? 48 89 D9 48 89 C2 48 89 E6 BF C0 BB 64 00 31 C0}
        $cde2 = {55 BE 60 BA 64 00 53 48 89 FB 48 81 EC 08 04 00 00 48 89 E7 48 89 E5
?? ?? ?? ?? ?? BE 15 5F 43 00 48 89 E7 ?? ?? ?? ?? ?? BE 6C BA 64 00 48 89 E7 ?? ??
?? ?? ?? BE 15 5F 43 00 48 89 E7 ?? ?? ?? ?? ?? BE 74 BA 64 00 48 89 E7 ?? ?? ?? ??
?? BE 10 5F 43 00 48 89 DF }
        $cde3 = {48 83 C3 01 ?? ?? ?? ?? ?? 48 29 EB 48 98 31 D2 48 F7 F3 48 8B 5C 24
08 48 8D 04 2A 48 8B 6C 24 10 48 83 C4 18 C3}
    condition:
        uint32(0) == 0x464c457f and 1 of ($cde*)
}

rule linux_GwisinLocker_ransomware {
    meta:
        author = "Marc Salinas @ CheckPoint Research"
        description = "Detects samples of the Linux ransomware family GwisinLocker"
        malware_family = "GwisinLocker"
        date = "09/08/2023"

        hash1 = "7594bf1d87d35b489545e283ef1785bb2e04637cc1ff1aca9b666dde70528e2b"

    strings:
        $str1 = "error: option `--%s` %s\n"
        $str2 = "Usage: %s"
        $str3 = "c.d/%s* stop"
        $str4 = "show this help message and exit"

        $ext = ".mcrgnx"
        $hex =  {66 30 66 64 62 33 64 38}

        $cde1 = {48 8B 74 24 08 31 FF ?? ?? ?? ?? ?? 48 85 C0 49 89 C7 0F 94 C1 41 83
FD 01 41 89 DD 0F 94 C0 08 C1}
```

```
        $cde2 = {41 54 55 53 48 81 EC 30 01 00 00 48 89 E5 48 89 EF ?? ?? ?? ?? ?? 48
8D 7C 24 20 B9 21 00 00 00 48 89 EA 48 8D 35 36 8B 00 00 F3 48 A5 8B 06 BE 0E 01 00
00 89 07 0F B7 05 2F 8C 00 00 66 89 47 04 48 8D 7C 24 20}
    condition:
        uint32(0) == 0x464c457f and ( (2 of ($str*) and ($ext or $hex)) or 1 of
($cde*) )
}

rule linux_Cl0p_ransomware {
    meta:
        author = "Marc Salinas @ CheckPoint Research"
        description = "Detects samples of the Linux ransomware family Cl0p"
        malware_family = "Cl0p"
        date = "09/08/2023"

        hash1 = "09d6dab9b70a74f61c41eaa485b37de9a40c86b6d2eae7413db11b4e6a8256ef"

    strings:
        $str1 = "C_I_0P"
        $str3 = "README_C_I_0P.TXT"
        $str4 = "OR WRITE TO THE CHAT AT->"

        $cde1 = {55 89 E5 57 81 EC 24 02 00 00 8D 8D F8 FE FF FF BA A4 83 10 08 B8 FC
00 00 00 89 44 24 08 89 54 24 04 89 0C 24 ?? ?? ?? ?? ?? 8B 45 08 89 44 24 08 C7 44
24 04 8C 83 10 08 8D 85 F8 FD FF FF 89 04 24 ?? ?? ?? ?? ?? 8D 85 F8 FE FF FF B9 FF
FF FF FF 89 85 E8 FD FF FF B8 00 00 00 00 FC 8B BD E8 FD FF FF F2 AE 89 C8 F7 D0 83
E8 01 89 45 F8 C7 44 24 08 B4 01 00 00 C7 44 24 04 42 00 00 00 8D 85 F8 FD FF FF 89
04 24 ?? ?? ?? ?? ?? 89 45 F4 8B 45 F8 89 44 24 08 8D 85 F8 FE FF FF 89 44 24 04 8B
45 F4 89 04 24 ?? ?? ?? ?? ?? 8B 45 F4 89 04 24 ?? ?? ?? ?? ?? 81 C4 24 02 00 00 5F
5D C3}
        $cde2 = {8D 95 0B FF FF FF B8 75 00 00 00 89 44 24 08 C7 44 24 04 00 00 00 00
89 14 24 ?? ?? ?? ?? ?? C7 45 F4 00 00 00 00}
    condition:
        uint32(0) == 0x464c457f and 1 of them
}
```

GO UP
BACK TO ALL POSTS