Universiteit van Amsterdam

SSN Student Research Project

Analysing MedusaLocker for cryptographic implementation errors

Authors
Isaac S
Yassir Laâouissi
(yassir.laaouissi@os3.nl)
Jan Laan
(Janpt.Laan@os3.nl)
Nicholas Mastorakis
(Nicholas.Mastorakis@os3.nl)

Supervisor
Vincent Breider

April 18, 2024

Abstract

Ransomware is a type of malicious software that aims to encrypt the data on the machines of a victim. Threatening to permanently lock, or even leak this potentially sensitive information if the demanded ransom fee is not paid, it essentially takes the victim hostage. Ransomware attacks continue to pose a substantial threat to individuals, organizations, and critical infrastructure. MedusaLocker is emerging as a prominent actor in the dynamic landscape of cybercrime. This research paper is dedicated to conducting an in-depth analysis focusing on uncovering vulnerabilities in the encryption schemes employed by MedusaLocker's ransomware. The research begins with establishing a sample set of ransomware, related to MedusaLocker, to analyze. To start this project, a set of ransomware samples related to MedusaLocker first needs to be gathered for analysis, after which previous publicized research on these samples is examined. Based on that, a subset of the samples is selected for further analysis. The research is continued with a comprehensive examination of the encryption techniques and algorithms utilized by MedusaLocker, revealing the intricacies of its cryptographic methods. By dissecting its encryption strategies and identifying potential flaws, we aim to understand the vulnerabilities within the ransomware's encryption process. These findings not only enhance our understanding of MedusaLocker's technical intricacies but also provide valuable insights into the weaknesses that can be exploited for decryption and recovery. To enhance the application of our findings, the research also investigates the potential implications of these encryption vulnerabilities for both the victims and cybersecurity experts. It explores the feasibility of decryption and data recovery with identified encryption flaws, providing critical guidance for affected parties and incident response teams. In conclusion, this research offers a meticulous analysis of the encryption schemes employed by more recent versions of the MedusaLocker ransomware, with a particular emphasis on identifying vulnerabilities within the implementation of these schemes. Uncovering these flaws is crucial in the fight against ransomware threats.

Contents

1 Introduction

The digitization of corporate activities has yielded numerous benefits in terms of operational efficiency. However, it has simultaneously increased the number of security threats, concerning personal and corporate data. Certain threats exploit human behavior, seeking to manipulate, influence, or deceive individuals to gain control over computer systems. Human manipulation of this kind could lead to the execution of software that has a malicious nature. In this research, such malicious software, also known as malware, is analyzed. The work will focus on a type of malware called ransomware. Ransomware aims to encrypt the victim's data, essentially keeping the data of the user hostage. The attacker then demands money, also called a ransom fee, in exchange for a decryption key that will allow the victim to decrypt their files.

This research aims to find implementation errors made by the Ransomware developer, which could serve as the foundation for eventually aiding infected users in decrypting their files without paying the attacker. This is achieved by performing static analysis on the MedusaLocker ransomware strain inside an isolated environment. Thus, the following research question is defined:

- How do implementations of encryption schemes in modern ransomware strains withstand against known encryption implementation bugs?
- **Sub-questions:**
  - Which encryption methods are used in our set of malware samples?
  - Which common implementation errors are known to be found within the encryption schemes used by these samples?

Section 2 will provide a number of related research blogs and papers that aid our research on the MedusaLocker ransomware samples. Section 3 describes the methods used to perform this research. This includes the isolated environment setup used for the malware analysis, and the samples from the MedusaLocker strain chosen for this work. Section 4 will show the results of the analysis. Finally, a conclusion of the research will be provided in Section 5, and potential future work will be suggested in Section 6.

2 Related work

For this work, vx-underground[1] was the source of all the samples from the MedusaLocker ransomware strain. The website of vx-underground is one of the largest collections of malware samples and corresponding research papers. From the selection of samples, various related work was found. Genheimer (2019) describes the inner workings of one of the MedusaLocker samples on their research blog. This includes the workings of the AES key generation functionality and an analysis of the decryptor provided by the attacker after the ransom fee is paid. Unnikrishnan (2022) explains the encryption scheme of MedusaLocker on his research blog to contain an AES key for file encryption, which is then encrypted by a public RSA key. They also explain that a function called sub_535840 is performing the encryption. Cyble (2023) posted a research blog, also analyzing the MedusaLocker ransomware, showing the increases of infected devices, and an overview of how the AES key is generated.

The papers listed on vx-underground are centered around generic research on the various malware samples attributed to the threat actor behind MedusaLocker. One research was found that has a scope of finding vulnerabilities in the encryption schemes of MedusaLocker, which was conducted by Kokado et al. (2020). However, this research dates back to October 2020 and is written in Japanese, which made it harder to correctly translate the exact meaning of the researcher, as no Japanese interpreter was involved during this research. Kokado et al. (2020) uses a sample of MedusaLocker with the MD5 hash-value of 129d3661a7341d3b069868a43714b42, which can not be found in the malware databases listed in Appendix A at the time of writing. The researchers might have made a mistake in their MD5 hash value. The MD5 hash value listed in their research paper is 129d3661a7341d3b069868a43714b42, while querying AlienVault OTX[2] it became clear that this might be 129d3661a7341d3b069868a43714b42**5** instead and therefore a typing error. A sample with this MD5 hash value can be found on VirusTotal.com[3]. Furthermore, the sample in question is detected by various AV vendors on VirusTotal's detection page[4] as MedusaLocker ransomware. VirusTotal first saw this sample in 2019 and has the identifiable characteristics listed in Table 1.

[1] https://vx-underground.org
[2] https://otx.alienvault.com
[3] https://www.virustotal.com/gui/file/3a5b015655f3aad4b4fd647aa34fda4ce784d75a20d12a73f8dc0e0d866e7e01/details
[4] https://www.virustotal.com/gui/file/3a5b015655f3aad4b4fd647aa34fda4ce784d75a20d12a73f8dc0e0d866e7e01/detection

Algorithm	Hash Value
MD5	129d3661a7341d3b069868a43714b425
SHA-256	3a5b015655f3aad4b4fd647aa34fda4ce784d75a20d12a73f8dc0e0d866e7e01
SSDEEP	12288:f+IZ+bobAyYFJPrsU4VwryxjpBx8ajiOhA8tsV1YRbRb7:2++EMyYFJPoUecOh8aWdD1UB7
IMPHASH	817569a2fb4950e9c4c506052cc6f032

Table 1: Used hashes in the research of Kokado et al. (2020)

The samples chosen for this work are part of the same strain, though more recent and different in imports and overall layout as will be described in Section 3.

3 Method

3.1 Environment setup

Ransomware tends to be designed to self-replicate, spread, and infect other devices. To analyze those ransomware samples, it is crucial to use a completely isolated and controlled environment. This ensures that the analysis is conducted safely and accurately.

3.1.1 Requirements

The following requirements are set to ensure a safe ransomware analysis:

- **Network isolation:** The environment should be restricted from internet communication during the presence of Ransomware on the VMs. In case of a dynamic analysis, the ransomware might communicate with a Command and Control server, or attempt to spread to other devices within the current network. Having a non-isolated network could result in significant damage from ransomware to the host computer or other devices on the same network.
- **Internet simulation:** Having a virtual machine that simulates the Internet by sending fake responses in the correct format can be a beneficial method to dynamically analyze ransomware. This could provide insights into the type of requests made by the ransomware during the attack.
- **Analysis tools:** The environment should have all necessary tools, such as disassemblers and debuggers, used for ransomware analysis before retrieving the ransomware.

- **Limit attack surface:** Virtual Machines (VMs) provide virtual devices like the floppy disk, CD-ROM, USB controllers, Serial ports, and parallel ports. These ports should be disabled on all the VMs, as it increases the attack surface, which should be minimized to prevent malware breakouts. For example, a previous vulnerability CVE-2015-3456[5] was exploited to abuse the Floppy Disk Controller (FDC), allowing an attacker to escape the VM and potentially execute malicious code on the host system.

- **Physical precautions:** It is important to keep the server running the VMs protected from accidents caused by other people who have access to the server room. To ensure this protection, all input ports (such as USB, Ethernet, and DisplayPort) are taped off and a note is attached, providing information about the purpose of the research and why the ports are taped off.

3.1.2 Architecture

The host machine used for this research is Fedora Linux[6]. All incoming ports to the host machine, except for RDP, are blocked and RDP is only reachable within the OS3 network. An open-source library, xrdp[7], provides the RDP protocol on Linux devices. Appendix B.1 shows the changes to the default xrdp configuration to restrict the protocol, disabling any additional features like USB, clipboard, and file passthrough. A type 2 hypervisor (QEMU/KVM) virtualizes the isolated environments. Access to the hypervisor is restricted to local connections and can only be used during an RDP session using virt-manager[8].

Three virtual networks are configured for the VMs:

- **(1) Default NAT:** This network is the default NAT, which is forwarded to the real internet and uses an IP range of 192.168.124.1/24. Only the virtual router should use this network, as the job of the router is to filter all the incoming and outgoing traffic.

- **(2) Isolated network:** The isolated network enables communication among VMs without having access to the internet. The main purpose of this network is for one VM to provide an Internet simulation to the VM on which the ransomware analysis is conducted. The IP range used for this network is 10.0.0.1/24.

- **(3) Virtual Router network:** This network connects the isolated VMs to the real internet via the virtual router that filters the traffic. The router firewall can decide which traffic is allowed, and more importantly, which traffic is not. The IP range used for this network is 178.16.0.1/16.

[5] https://nvd.nist.gov/vuln/detail/CVE-2015-34566
[6] https://fedoraproject.org/
[7] https://github.com/neutrinolabs/xrdp
[8] https://virt-manager.org/

The architecture consists of three VMs for this work:

- **FlareVM**[9]**:** This VM is spun up from a stock Windows 10 workstation ISO. FlareVM is a collection of software installation scripts for Windows that installs all the tools needed for reverse engineering. This VM is used as the primary workstation for the ransomware analysis.
  Connected network interfaces: 2, 3

- **Remnux**[10]**:** This VM is a Linux Toolkit for Malware Analysis based on Ubuntu. Remnux is mainly used for this research to simulate the internet for the FlareVM device.
  Connected network interfaces: 2

- **VyOS**[11]**:** VyOS is an open-source router system based on Debian. VyOS provides various features like routing, NAT, and Firewall rules. This VM acts as a failsafe for the other VMs used for ransomware analysis to prevent exposure to the full internet when an error has been made, and for filtering the traffic during the ransomware retrieval stage. The VM also acts as a NAT, as shown in Appendix B.2.2. The interface configuration is noted at Appendix B.2.1
  Connected network interfaces: 1, 3

Figure 1 shows a visualization of the introduced VM architecture and its network interfaces connected to other VMs running on a Fedora OS environment.

[9] https://github.com/mandiant/flare-vm
[10] https://remnux.org/
[11] https://vyos.io/

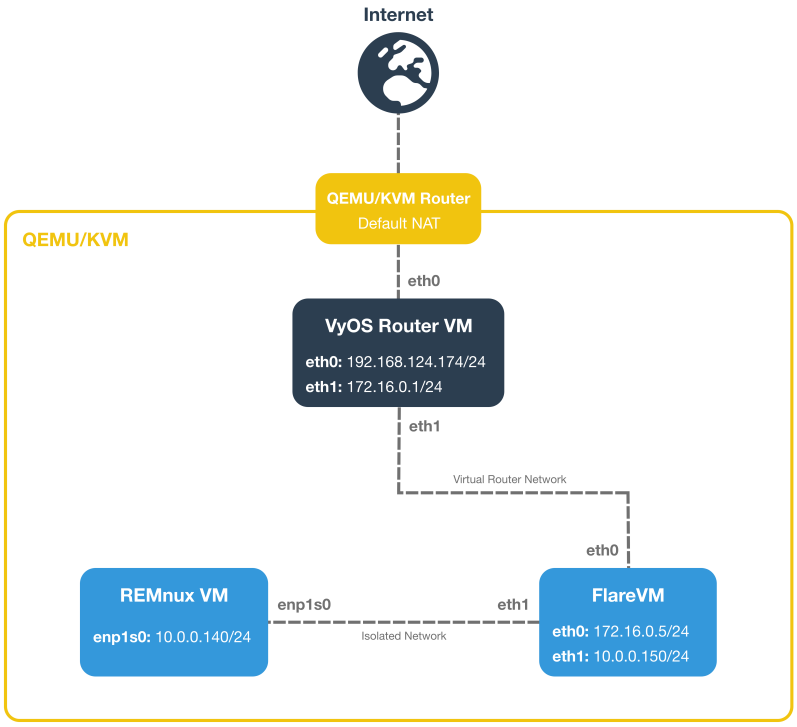

Figure 1: Visualisation of the isolated VM architecture

3.1.3 Stages

Throughout the research process, it was necessary to consider different stages for the environment to be in. To install the tools required for ransomware analysis on the VMs, internet access was mandatory. However, it was also crucial to completely isolate the environment once this phase was completed. Therefore, separating the different stages and tasks is of utmost importance before beginning the research on the ransomware samples.

The following stages were considered during the research:

- **Stage 1:** This stage requires internet access for each VM. During this stage, all the required tools for reverse engineering, static analysis, and dynamic analysis. This stage is important, as every tool needed should be present in the environment, and the internet should be cut off completely when the ransomware is retrieved on the VM.
- **Stage 2:** Stage 2 limits the internet access for all the VMs using the virtual router. A firewall should be set up to exclusively allow incoming/outgoing requests to the server where the Ransomware is stored. Appendix

B.2.3 shows the firewall configuration OUTSIDE-IN and OUTSIDE-OUT used to prevent all traffic except the allowed IP. This acts as an extra safety measure in case the ransomware does ignite. With this in place, it will not be able to connect to the internet apart from the server that hosts it. After retrieving the ransomware, it should remain packed in a compressed format and untouched during this stage.

- **Stage 3:** In Stage 3, the firewall of the virtual router is reconfigured to block all incoming and outgoing packets. This effectively severs all connections between the VMs and the internet, ensuring that the system is safe for analysis. Once it has been confirmed that the internet is completely cut off, the ransomware can be unpacked and analyzed within the isolated environment.

3.2 Analysis methods

This section will describe how the malware samples are analyzed for the project. Due to time constraints, only static analysis was performed on sample ML-14. This consisted of three parts:

- Comparing the sample to other samples that may or may not have been analyzed before, using several hashes.
- Looking at imported functions and their possible uses.
- Decompiling the sample using Ghidra in the FlareVM and looking at the assembly and the C code generated based on the disassembly.

3.2.1 Hashes

The samples are hashed using three different algorithms. The first of these is **SHA256**, but the only thing this tells us is whether or not several samples are identical. The second hashing algorithm, which is more useful for comparing potentially similar files, is **SSDEEP**[12]. SSDEEP is a *Context Triggered Piecewise Hash* (CTPH) algorithm, also known as a fuzzy hash. This hash can be used to detect similarities between files, and quantify how significantly two samples differ. Finally, there is **ImpHash**, short for import hash, used for comparing which standard library imports are used in samples [13]. As the name suggests, it is a hash based on the imports of the Portable Executable (PE) and can be used to find out which samples contain the same imports.

3.2.2 Imports

Looking at the imported standard libraries can tell analysts something about the samples without decompiling them. A library that is not imported, but does seem necessary, might mean that the functionality of that library has been

[12] `https://ssdeep-project.github.io/ssdeep/index.html`
[13] `https://www.mandiant.com/resources/blog/tracking-malware-import-hashing`

written by the developer of the ransomware. This might not be done perfectly, so bugs may be present in that part of the sample. Differences in imports between samples may also be used to quantify the difference between samples. More specifically, the details section in the VirusTotal[14] page of the sample reveals which exact function is actually used in the sample. Certain functions that seem crucial for encrypting files, for example, may not be present in certain samples. Again, this could indicate that the authors of the sample have implemented their own scheme to achieve the same effect. Again, this is a place where analysts should look for potential bugs.

3.2.3 Decompilation

Decompilation can tell analysts the most about a sample, but it is also significantly more time-consuming. The program used for decompilation is Ghidra, included with the FlareVM. It provides a GUI with assembly code and generated pseudocode side by side. Navigating the pseudocode is simple, and the automatically generated names for functions and variables can be manually changed to something that represents their uses.

The decompiled samples allow inspection of how functions are used, which parameters are passed to them, etc. Although all the functions are somewhat readable, the path through the program is difficult to trace and very time-consuming. Because of this, in combination with time constraints, the project is limited to inspecting those parts of the sample that are most relevant to encryption.

4 Results

As described in Section 2, the samples were obtained from vx-underground.org. The retrieved samples are detailed in Appendix C.

4.1 Hashes

From the hashes, several details can be inferred. First of all, the ImpHash of ML-1 does not exist, as it uses no dynamically linked libraries. Instead, the sample has been statically linked. This means that all the functionality that would otherwise be imported, has been compiled into the binary. This makes any analysis much more complex than it would be with imported libraries, as the "imported" functions are not obvious in the disassembled code, whereas a library call with a dynamically linked binary can be read as such. This is because the name of the imported function would still be readable. In contrast, statically linked functions are given a generic name by the decompilation program.

[14] https://www.virustotal.com/gui/file/dbac4f2fffcb4e09aad772895647e8f161b1ac713592fe47c5e8207c85722f13/details

Looking at the ImpHashes of the remaining samples, as listed in appendix C, it turns out that those of samples ML-2 through ML-17 are identical. This implies that they all import the same libraries. The ImpHashes of ML-18 through ML-20 all differ, and must therefore use different libraries than the others. Given the similarities between ML-2 and ML-17, it was decided to start analyzing one of those samples, as any results from this would probably apply to the other samples with the same imports, too. To reduce the risk of previous research into the sample already existing, the sample most recently uploaded to VirusTotal was chosen: ML-14.

4.2 Imports

The first part of the static analysis of ML-14 consisted of finding the specific functions imported, specifically those related to cryptography. VirusTotal provides a list, which we reduced to the following functions from **ADVAPI32.dll**:

- CryptAcquireContextW
- CryptDestroyKey
- CryptDuplicateKey
- CryptEncrypt
- CryptExportKey
- CryptGenKey
- CryptImportKey
- CryptReleaseContext

ML-18 uses a function named **CryptGenRandom** from the same DLL, which can be used to fill a buffer with cryptographically random bytes. (Microsoft Corporation, 2021b) This raises the questions: why doesn't ML-14? And what does ML-14 use instead? These would be answered with further analysis, be it static or dynamic.

Another function that arouses interest is **CryptDestroyKey**, especially in combination with **CryptDuplicateKey**, which is not used by any of the samples analyzed this far. Furthermore, the function only destroys the key handle, but it does not overwrite the memory space that actually stores the key (Microsoft Corporation, 2021a). This might mean that the key itself, or at least the struct containing the key, is left in memory. However, proving this would require dynamic analysis.

4.3 Decompilation

The encryption scheme used to encrypt the files of the victim is AES-256. This was found during the static analysis of ML-14 in one of the functions shown in Figure 2, which calls the CryptGenKey with an *algorithm ID* parameter of 0x6610. This algorithm ID corresponds to *CALG_AES_256*, according to the WinCrypt documentation[15].

```
1
2 bool __thiscall GenerateAESKey(void *this,HCRYPTKEY *AESEncryptionKey)
3
4 {
5   BOOL Status_CryptGenKey;
6
7   Status_CryptGenKey = CryptGenKey(*(HCRYPTPROV *)((int)this + 8),0x6610,1,AESEncryptionKey);
8   return Status_CryptGenKey != 0;
9 }
10
```

Figure 2: The code responsible for generating the AES-256 key

ML-14 uses the Windows built-in function **CryptDestroyKey**. Despite what its name suggests, this function does not actually destroy the key in memory in certain cases, only the **handle**, or pointer, to the key if it is a so-called public/private keypair. This implies the possibility of the public/private key is still being somewhere in memory. The way this function is used is also quite interesting in regard to so-called session keys.
Before the actual encrypting, a session key is generated using the built-in **CryptGenKey**. This is stored in memory, with the handle being visible in Ghidra. However, another variable is visible in Ghidra, also specific for storing key handles. The created key is copied to this second handle, and the copied key is used for encrypting files. Interestingly, the **CryptDestroyKey** function is called on the copied key after every encrypted file. Shortly after, before the next file, the original key is re-copied into the handler variable that is used to store the key handle that was just destroyed. In short, the key is copied and destroyed for every file that the ransomware encrypts. During the analysis, no proof has been found that the original key being duplicated was actually destroyed.

Additionally, ML-14 has not been found to use the pseudorandom number generator (PRNG) source of the WinCrypt library. Not using a PRNG correctly might imply that the seed used for generating the AES-256 key could be reproducible. The developer either implements this from scratch, or no PRNG is used at all.

[15] https://learn.microsoft.com/en-us/windows/win32/seccrypto/alg-id

5 Conclusion

In this study, we conducted a static analysis of the MedusaLocker ransomware to identify errors in its cryptography implementation. The encryption scheme used for encrypting the victim's files on the computer was found to be AES-256, a symmetric encryption algorithm.

Various potential implementation errors were found during the static analysis. The ransomware sample uses the WinCrypt library of Windows, which has been deprecated since the 12th of October 2021, where the sample analyzed was first reported on the 9th of March 2022. Using deprecated libraries is bad practice, as security flaws are usually no longer patched. Additionally, no pseudorandom number generator (PRNG) source of the WinCrypt library was found during the analysis, meaning that the developer either used their own PRNG source or no source was used. Implementation errors in the PRNG source potentially cause a flaw in the seed generation of the AES-256 key, meaning that the generated key could have a predictable seed, ultimately allowing its reproducibility. Another flaw found is the use of CryptDestroyKey. CryptDestroyKey destroys the key and frees the memory that the key used for session keys. However, for public/private key pairs, this function does not destroy the key, just the handle. This means that the actual key is not guaranteed to be overwritten in memory. The MedusaLocker also uses a public/private key pair at a particular stage that could potentially leave traces of the key. The MedusaLocker duplicates the generated AES-256 key for every file to be encrypted; this duplicate key is destroyed after every file encryption. However, no proof has been found that the MedusaLocker destroys the original generated key, potentially leaving the AES-256 key in memory.

This study lays the foundation for ongoing research on the MedusaLocker ransomware strain, offering valuable insights that could aid in the future recovery of victims' files. However, achieving this recovery, requires future research.

6 Future work

This work analyzed the MedusaLocker for cryptographic implementation errors. This study provides insights into a few implementation errors found during the analysis. However, there is still future work that could improve upon this.

The analysis of this work was limited to one of the samples from the MedusaLocker strain due to time constraints. Other samples, especially the ones with different structural properties, like ML-1 which uses statically linked libraries, might provide other insights. Furthermore, the sample ML-20, is written in C# and completely obfuscated, whereas the other MedusaLocker samples are written in C. Finally, samples ML-18 and ML-19 have a significantly larger SSDEEP difference, which could imply another cryptographic implementation. Analyz-

ing these samples might provide further insights.

The hypothesis that the used function **CryptDestroyKey** might not completely destroy the key is yet to be proven. Theoretically, the function only deletes the pointer, suggesting that the key data is still in memory. To confirm this hypothesis, dynamic analysis should be performed to prove that the key is indeed not completely removed, and still readable after calling the CryptDestroyKey function. Since a symmetric AES-256 key is used, retrieving this key could potentially be the basis for a recovery tool for recently infected machines.

Finally, this work found that the random seed generator function from the WinCrypt library is not used by this ransomware sample. This could imply the use of a proprietary implementation to generate the keys, or the use of the default random seed generated during **CryptKeyGen**, provided by the WinCrypt library. Further research is needed to determine the seed used during generation, potentially leading to insights into whether the seed is predictable.

References

Cyble. (2023, Mar). Unmasking medusalocker ransomware. *Cyble*. Retrieved from https://samples.vx-underground.org/root/Papers/Malware%20Defense/Malware%20Analysis/2023-03-15%20-%20Unmasking%20MedusaLocker%20Ransomware.pdf

Genheimer, M. (2019, Nov). *dissectingmalwa.re*. Retrieved from https://dissectingmalwa.re/try-not-to-stare-medusalocker-at-a-glance.html

Kokado, R., Ikegami, M., Hasegawa, T., Harada, T., Kitani, H., & Morii, M. (2020). Development of a recovery tool for a ransomware infected pc. *IEICE Technical Report; IEICE Tech. Rep.*, *119*(437), 55–60. Retrieved from https://ipsj.ixsq.nii.ac.jp/ej/?action=repository_action_common_download&item_id=208557&item_no=1&attribute_id=1&file_no=1

Microsoft Corporation, I. (2021a). *Cryptdestroykey function (wincrypt.h)*. Retrieved from https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptdestroykey

Microsoft Corporation, I. (2021b). *Cryptgenrandom function (wincrypt.h)*. Retrieved from https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptgenrandom

Unnikrishnan, A. (2022, Sep). Technical analysis of medusalocker ransomware. *Cloudsek*. Retrieved from https://www.cloudsek.com/blog/technical-analysis-of-medusalocker-ransomware

Appendices

A Malware databases

virustotal.com

virusshare.com

virus.exchange by vx-underground.org

MalwareBazaar

Inquest DFI

Inquest IOCDB

B Configurations

B.1 xrdp

```
[Channels]
rdpdr=false
rdpsnd=false
drdynvc=false
cliprdr=false
rail=false
xrdpvr=true
tcutils=false
```

B.2 VyOS

B.2.1 Interfaces

```
ethernet eth0 {
    address dhcp
    description EXTERNAL-INTERNET
    duplex auto
    firewall {
        in {
            name OUTSIDE-IN
        }
        out {
            name OUTSIDE-OUT
        }
    }
    hw-id 52:54:00:cb:c2:dd
    smp-affinity auto
```

```
    speed auto
}
ethernet eth1 {
    address 172.16.0.1/24
    description INTERNAL-ISOLATED
    duplex auto
    firewall {
        local {
            name DROP-LOCAL
        }
        in {
            name DROP-ALL # AT STAGE 3
        }
        out {
            name DROP-ALL # AT STAGE 3
        }
    }
    hw-id 52:54:00:02:71:ba
    smp-affinity auto
    speed auto
}
```

B.2.2 NAT

```
source {
    rule 100 {
        outbound-interface eth0
        source {
            address 172.16.0.0/24
        }
        translation {
            address masquerade
        }
    }
}
```

B.2.3 Firewall

```
name DROP-LOCAL {
    default-action drop
}
name OUTSIDE-IN {
    default-action drop
    rule 10 {
```

```
        action accept
        source {
            address ALLOWED_IP_FOR_RETRIEVAL
        }
    }
}
name OUTSIDE-OUT {
    default-action drop
    rule 10 {
        action accept
        destination {
            address ALLOWED_IP_FOR_RETRIEVAL
        }
    }
}
name DROP-ALL {
    default-action drop
}
```

C Sample Details

ML-1	
SHA256	0c4a2efa9863f531e06628e80727a593fae11942d35563fc071043f0a024c0b7
ssdeep	24576:p3JcpHjHzo9qz4543wa2SZ6Ey9/fNm0x:7ci9q85umE+/f
ImpHash	Statically linked, no imports
URL	https://www.virustotal.com/gui/file/ 0c4a2efa9863f531e06628e80727a593fae11942d35563fc071043f0a024c0b7/details
First Submission	2022-01-25 12:15:34 UTC

ML-2	
SHA256	ec2ec1c316045d5e2e43cc0f1df738e6367b520310a4b7a644717d3aebda43f4
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DWKD/KeXI:Tuf4wTuV2Ux3uIZeUBi2Te6HW7KrKeY
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ ec2ec1c316045d5e2e43cc0f1df738e6367b520310a4b7a644717d3aebda43f4/details
First Submission	2021-11-30 10:15:36 UTC

ML-3	
SHA256	6c51f28a6ab35c91e789a4b1a05032c87a3f03006019ba4997dc092ad1c8a625
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DyKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWnKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ 6c51f28a6ab35c91e789a4b1a05032c87a3f03006019ba4997dc092ad1c8a625/details
First Submission	2021-12-01 05:04:13 UTC

ML-4	
SHA256	19e31469f150f69bda363c8a3454113236620aa44155dbe845e7689522724b0b
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DCKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWjKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ 19e31469f150f69bda363c8a3454113236620aa44155dbe845e7689522724b0b/details
First Submission	2022-02-09 12:33:24 UTC

ML-5	
SHA256	26af2222204fca27c0fdabf9eefbfdb638a8a9322b297119f85cce3c708090f0
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8D4KD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWdKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ 26af2222204fca27c0fdabf9eefbfdb638a8a9322b297119f85cce3c708090f0/details
First Submission	2023-01-24 13:52:20 UTC

ML-6	
SHA256	58a0db1ae0d7d8c5cb5db5e5a24fd1088b8029a4e51c02e7b77d400c17bcb39a
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DSKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWfKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ 58a0db1ae0d7d8c5cb5db5e5a24fd1088b8029a4e51c02e7b77d400c17bcb39a/details
First Submission	2022-02-19 11:15:27 UTC

ML-7	
SHA256	66a13e8102f809e23e0ad0ba88ced5eecfa319797c9f709d090994a7143d858a
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DEKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWdKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ 66a13e8102f809e23e0ad0ba88ced5eecfa319797c9f709d090994a7143d858a/details
First Submission	2022-02-17 17:57:38 UTC

ML-8	
SHA256	99a72b56725196298391f3d52b8536b018aa8b60d97c443161e912430079ed30
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DUKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWBKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ 99a72b56725196298391f3d52b8536b018aa8b60d97c443161e912430079ed30/details
First Submission	2022-01-31 14:22:43 UTC

ML-9	
SHA256	8939141fb565c044895627bbeb522d840d24899dec53545e4a925012dbf83230
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DCKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWLKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ 8939141fb565c044895627bbeb522d840d24899dec53545e4a925012dbf83230/details
First Submission	2021-11-30 00:15:54 UTC

ML-10	
SHA256	a3fe92224060ec183a25296999c18d4f86149649f1a701ac91b04d73e8678495
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DxKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HW0KrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ a3fe92224060ec183a25296999c18d4f86149649f1a701ac91b04d73e8678495/details
First Submission	2022-02-03 14:21:39 UTC

ML-11	
SHA256	b15840fb0547fc774f371166adb89cd7a58647d4e379256a2f9806dd5a338627
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8D9KD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HW4KrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ b15840fb0547fc774f371166adb89cd7a58647d4e379256a2f9806dd5a338627/details
First Submission	2022-02-01 02:03:52 UTC

ML-12	
SHA256	c79c6b680a2caa71b3ad052f60ce6da463eb576b8196bb3bbdccd003853769d4
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8D3KD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWWKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ c79c6b680a2caa71b3ad052f60ce6da463eb576b8196bb3bbdccd003853769d4/details
First Submission	2022-02-15 14:02:14 UTC

ML-13	
SHA256	cb12325d13acb03ad4f9977f426baf8b4688af04d4ffe23aa5f1bbd747a147c0
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DEKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWpKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ cb12325d13acb03ad4f9977f426baf8b4688af04d4ffe23aa5f1bbd747a147c0/details
First Submission	2021-12-10 07:38:23 UTC

ML-14	
SHA256	dbac4f2fffcb4e09aad772895647e8f161b1ac713592fe47c5e8207c85722f13
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DdKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWAKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ dbac4f2fffcb4e09aad772895647e8f161b1ac713592fe47c5e8207c85722f13/details
First Submission	2022-03-09 23:23:41 UTC

ML-15	
SHA256	f1c361bb3b649918bc5b3ad3fc5cbd1bbd7c585fbe2557410e267d161d3bb998
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DgKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWdKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ f1c361bb3b649918bc5b3ad3fc5cbd1bbd7c585fbe2557410e267d161d3bb998/details
First Submission	2021-12-06 12:41:01 UTC

ML-16	
SHA256	fbe10da8d483a0db6686b1f03f18b00dbc60c69fb9a9f4a941764c2c3426367c
ssdeep	12288:dQA0FfTcwpBuV2UxqDmuiLZeUaoFi2XZWfGe615HhAZV8DKKD/KeX:Tuf4wTuV2Ux3uIZeUBi2Te6HWnKrKe
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ fbe10da8d483a0db6686b1f03f18b00dbc60c69fb9a9f4a941764c2c3426367c/details
First Submission	2021-12-28 10:16:15 UTC

ML-17	
SHA256	465ab4311a7db9f0bc10921cf6a0da7a746c4023dd78fdcec1c253eee69e5b9d
ssdeep	12288:ZYk1LNT35lDbK/LIVaN8+T7vwqyqhYMhWt918vulAxwQ/EC9+m:Pd35lDbKDIwWUDyqS5omijEC9+
ImpHash	1a395bd10b20c116b11c2db5ee44c225
URL	https://www.virustotal.com/gui/file/ 465ab4311a7db9f0bc10921cf6a0da7a746c4023dd78fdcec1c253eee69e5b9d/details
First Submission	2021-11-21 06:39:14 UTC

ML-18	
SHA256	634d84758d8d922bbfb0ad3c904c38fc7989f11503877acf02ad5dad3775df7a
ssdeep	12288:PyX/Gc6mBG6Lg9PNPjvrdY0YZa2+OeO+OeNhBBhhBBKfsDENv5c+QgMJu1sssn:PyXuc6mE609PNPjDdYpyrPguy
ImpHash	94a05edd536770f9e1ec8c773654f3a1
URL	https://www.virustotal.com/gui/file/ 634d84758d8d922bbfb0ad3c904c38fc7989f11503877acf02ad5dad3775df7a/details
First Submission	2022-02-13 17:26:35 UTC

ML-19	
SHA256	c41926a4e667a38bd712cd8fff2c555c51d7f719a949c9be8c1f74232100444b
ssdeep	6144:vE+MVrAQU3V1jWEaFt7Ka5ZrP0qwTA39RNxCjRp/FGyZ91DHtdTBJZEdjDMr:vE+I3UFVza5ZLMTERa7N5Z9ttdTly0r
ImpHash	34d3cfe59cf0e09bc867b43bab6fa814
URL	https://www.virustotal.com/gui/file/ c41926a4e667a38bd712cd8fff2c555c51d7f719a949c9be8c1f74232100444b
First Submission	2022-02-15 10:22:16 UTC

ML-20	
SHA256	98c9e56cba271bf7b32fc17d7966d067d9b549594f8dc60c941f93346e376c00
ssdeep	6144:Cy6EFVP/UnTqSjgYgTERaj7xuoxxDseXQ1Iq2uN1A3JdCBeQnqNlLj7EPC5frryh:/FoTqSpgFNTrjAGq2amJGeQa7qD
ImpHash	f34d5f2d4577ed6d9ceec516c1f5a744
URL	https://www.virustotal.com/gui/file/ 98c9e56cba271bf7b32fc17d7966d067d9b549594f8dc60c941f93346e376c00/details
First Submission	2022-02-19 22:16:47 UTC