Mac-OS Ransomware Protection: Prevention and Detection Mechanisms

Ioana BRĂNESCU[1], Emil SIMION[2]
[1]Politehnica University of Bucharest
Faculty of Automatic Control and Computers
ioana.branescu0601@stud.acs.upb.ro

[2]Politehnica University of Bucharest
Faculty of Applied Sciences
Center for Research and Training in Innovative Techniques of Applied Mathematics in Engineering "Traian Lalescu"
emil.simion@upb.ro

Abstract: As we are living in the era of information, the security of computer systems is gaining more and more attention. An attacker needs only one vulnerability, which, once exploited, can have a devastating effect on the targeted system. There are various reasons behind such attacks, but one of the most frequent motivations is the financial one, a well-known example of such an attack being the one which uses ransomware malware. This article analyses different methods of protection against ransomware malware on Mac-OS devices and studies the mechanism which made past ransomware attacks successful.
Keywords: ransomware, Mac-OS, malware, cybersecurity, encryption.

INTRODUCTION

Mac-OS popularity has increased significantly lately, as it can be noticed with regard to Apple's computer shipment growth by 28% last year. With a shipment of over 29 million devices, Macs purchases grew twice as fast as overall PC purchases, making 2021 one of the most successful year for Apple's revenues. (Canalys, 2022)

However, as the popularity of this OS increases, the attention it gets from attackers also increases. AV-TEST Institute (AV-TEST Institute, 2022) reports the number of threats grew almost 20 times from 2015 to 2022, only for Macs (Figure 1). Even if Mac-OS still features a smaller amount of specific malware compared to Windows systems (AV-TEST Institute, 2022), it is important to be aware of the fact that it is not bullet proof and targeted Mac-OS malware still exists. As in the case of other operating systems, there are different types of malware which are found on Apple's OSs, from Adware, Trojans and Potentially unwanted applications (PUAs) to Crypto-miners and Ransomware. Any type of malware is obviously unwanted on any machine, but among the common malware types, ransomware has a bigger impact than other Mac-OS reported viruses and it is considered to be one of the most dangerous.

https://doi.org/10.54851/v5i1y202308

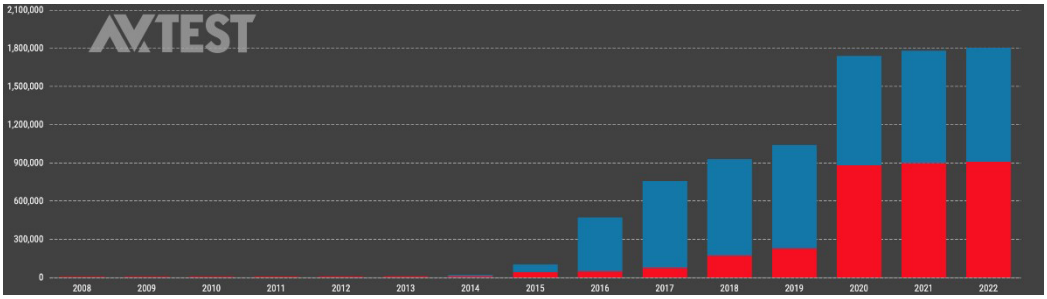

Figure 1. *Evolution of total amount of malware and PUAs under Mac-OS, by year (AV-TEST Institute, 2022)*

ABOUT RANSOMWARE

Ransomware is a type of malware which makes data unavailable to the user, by either locking or encrypting it, and only makes it available again if a payment is made quickly enough. The payment is usually made with cryptocurrency, which makes the attacker untraceable. The European Union Agency for Cybersecurity (ENISA) reports that ransomware is the prime cybersecurity threat across the European Union, with 10 terabytes of data stolen monthly from targeted organizations and a projected cost estimation of more than 10 trillion dollars by 2053 (European Union Agency for Cybersecurity, 2022). There are two main types of ransomware (Kapoor et al., 2022):

- Locker ransomware prevents the infected devices from being accessed, without actually encrypting the stored data. This type of malware is not usually effective, as the data can by recovered by attaching the storage device to another computer.
- Crypto-ransomware encrypts the victims' data using various encryption algorithms. As this kind of malware evolved, it has used four main type of encryption algorithms:
  - Symmetric key encryption is the fastest way to encrypt data. However, this method is not used by modern ransomware as it is vulnerable: the encryption key which is also the decryption key is often stored on the targeted device at some point and finding the key can ruin the attack;
  - Client-side asymmetric encryption uses a public key to encrypt the data and then sends the private key to the attackers' server. This method is also not frequently used today as the attack can fail if the targeted computer goes offline before the private key is deleted from the infected machine and sent to the server;
  - Server-side asymmetric encryption is employed for encrypting the files with a public key generated on the server and for transferring the private key after the ransom is paid. The risk for the attacker is that when the private key is transferred, it can be intercepted and shared with other victims;
  - Hybrid encryption tries to resolve the vulnerabilities of the previous encryption methods and is adopted by modern ransomware.

Cryptographic behavior

Modern ransomware uses a more complex, hybrid encryption-decryption scheme. Firstly, when the malware starts encrypting files, it uses a self-generated symmetric key. Then, this symmetric key is encrypted with a public (asymmetric) key embedded in the malware, which was generated by the attacker who also kept the corresponding private key. When the files are already encrypted, the victim only has the encrypted key which could be used for file decryption and he can only decrypt it by using

the attacker`s private key. For doing that, the victim has to send the encrypted symmetric key to the attacker, together with the requested amount of money. The attacker uses his private key to decrypt the encrypted key and he sends it back to the victim, who should now be able to use it for decrypting the files and restoring the data. An interesting aspect is that an Internet connection isn't needed during encryption, but it is required during decryption.

Attack phases

The main stages of the ransomware kill chain are usually the following:

- Campaign distribution: during this phase the user is tricked to download a malicious file. This is done in various ways, two relevant examples being the use of phishing emails with infected attachments or using Trojans for making the application look legitimate.
- Infection: during this phase the ransomware itself is downloaded, its installation begins and it sets up.
- Staging: in this stage, the malware tries to overcome the security checks. It might get root privilege, start using random extensions which aren't blacklisted by security tools or establish itself as a launch application to establish persistence.
- Scanning: the malware scans for files it can encrypt on the targeted system.
- Encryption: the ransomware encrypts either the whole file-system, specific files or even the backup, if existent.
- Payday: finally, a ransomware note is displayed to the victim with the payment instructions. In some cases, it might also include a deadline from which the software begins deleting the encrypted files or the ransom begins to rise.

Attack phases

Cryptocurrencies have a major role in the recent rise of ransomware attacks. Until digital currency was available, if an attacker wanted to ask for ransom, he had to provide an account which could easily link him to the attack. Nowadays, a crypto-wallet and a Bitcoin payment make tracing the attacker much more difficult.

Even if the transactions on blockchain are transparent and monitoring a transaction is easy, there is no direct link between an account and its holder. That, together with the emergence of dark web and anonymous browsing technology like Tor not only put anonymity and untraceability into practice when it came to ransom payment but also contributed to the rise of Ransomware as a Service. This refers to the possibility of using ransomware attacks crafted by third parties, based on a subscription-like mechanism, under the condition of sharing a percentage of the earnings with those who built the attack.

MAC-OS NATIVE PROTECTION SYSTEM

The low number of successful ransomware attacks on Mac-OS is not a coincidence. The built-in security features make it significantly harder for such attacks to succeed.

X-Protect and MRT

X-Protect is the default Mac-OS built-in antivirus. It is a very light-weight antivirus because it doesn't monitor the whole system all the time. It only checks an application when it is firstly launched, when its signature changes or when X-Protect signatures are updated. Also, when a file downloaded by a File Quarantine-aware application (one that can download files such as Safari, Mail) is opened, X-Protect automatically scans it. The scanning itself is based on YARA signatures which are kept in a special XProtect. yara file, as it can be seen in Figure 2.

Malware Removal Tool (MRT) kicks in after restart or logging in and it is also signature-based. However, it scans significantly more files than X-Protect so it might be responsible for temporarily CPU usage increase. In the context of ransomware protection, these mechanisms are relevant as an infected application with an already known signature should immediately be found and reported.

```
rule XProtect_MACOS_7c241b4
{
    meta:
        description = "MACOS.7c241b4"

    strings:
        $a1 = { 5f 54 72 61 6e 73 66 6f 72 6d 50 72 6f 63 65 73 73 54 79 70 65 }
        $a2 = { 5f 69 6e 66 6c 61 74 65 49 6e 69 74 }
        $b1 = { 90 4? 63 c? 48 8? 0d ?? ?? 00 00 32 14 08 4c 39 fb }
        $b2 = { 49 63 c6 48 8d 0d ?? ?? 00 00 44 32 3c 08 90 48 8b 85 78 ff ff ff 48 3b 45 80 }
        $b3 = { ff cb [0-2] 48 63 c3 48 8b (15 | 0d) ?? ?? 00 (00 | 00 44) 32 ?? ?? 48 8b ?5 [1-4] 48 3b ?5 }

    condition:
        Macho and any of ( $a* ) and any of ( $b* )
}
```

Figure 2. *X-Protect Yara rule example (Dubey, 2022)*

Notarization

Starting with Mac-OS Catalina, all applications should be notarized to run properly. Notarization takes code signing a step further. If by code signing a hash of an application is computed so that it can be further checked if the application wasn't changed or modified, notarization is a kind of two-factor authentication for code signing. After a developer signs a piece of software, a copy of it is sent to Apple along with the developers' Apple ID credentials so that Apple can check if the submitted software isn't malicious, match everything up and produce a cryptographic seal-of-approval for that trusted piece of software. The results of notarization should be attached to the distributed software so that no issue arises even if the application runs offline.

Gatekeeper

A gatekeeper is designed to ensure that only trusted software runs on Mac-OS and it is closely related to the notarization process. All software in Mac-OS is checked the first time it is opened. It should come from either App-Store or a registered developer and should be notarized by Apple. Unnotarized software can still run, but the user must explicitly ignore the security warnings or even disable certain protection layers, like System Integrity Protection.

Time machine backup

The operating system offers the possibility to backup data by essentially copying different files to another hard drive. This functionality, called Time Machine, is more than manually copying specific data to an external location. Once the first backup is done, it will perform routine backups by saving any changes made since the last backup. After a file is saved, it won't be saved again until it's modified again, making consecutive backups quite efficient. From the perspective of ransomware protection, even if a Mac-OS falls victim to such an attack, the data should be recoverable if Time Machine is enabled.

MAC-OS RANSOMWARE

The first known attempt involving a Mac-OS ransomware appeared in July 2013. As detailed in (Wardle, 2016) by Malwarebytes Inc., this malicious code locked the user's browser stating the browser has been blocked due to the user's online activity and a ransom of 300$ must be paid in order for the browser to be unlocked. However, this was not a real ransomware sample as all it did was to lock the browser by creating 150 iFrames via JavaScript. The malware did not and was not capable of encrypting any file and the user could simply kill the browser process to escape this attack. Still, it is notable that the attackers were starting to target Mac-OS users for money extortion via ransomware-like attacks.

Another OSX ransomware attempt appeared in June 2014. According to Kaspersky Lab (Wardle, 2016), FileCoder was rather an uncompleted Mac-OS ransomware malware. It had some of the characteristics of such a virus, like trying to encrypt some files and requesting money from the victims, but it turned out it was only encrypting its own files. Moreover, beside this

obvious weakness, the virus used a static and symmetric encryption key.

In 2015, two security researchers published two proof-of-concept ransomware viruses. One of them, called Mabouia (detected by Symantec as OSX.Ransomcrypt) was developed by a Brazilian cybersecurity researcher, who wrote the malware to highlight the fact that Macs may also be affected by ransomware (Johnson, 2015). The sample was shared with Apple and Symantec and it was confirmed that it could be used to create real, „in the wild" Mac-OS ransomware (obviously the source code wasn't made public).

KeRanger

The first fully functional Mac-OS ransomware appeared in 2016. It was called KeRanger and was distributed along with a piece of legitimate software: Transmission torrent client. The malicious application was signed with a valid certificate, so it passed undetected by the Gatekeeper. If a user installed the apparently legitimate application, an embedded executable file ran on the system. More specifically, the infected installer included a hidden file in the Resources directory. This file was camouflaged by using an icon that made it look like a legitimate Rich Text Format file, when it actually was an executable file. When the user clicked on the infected application, the bundle executable copied the fake RTF file to ~/Library/kernel_ service and executed it before the user interface appeared. When executed for the first time, KeRanger wrote the current time to a hidden file and then slept for three days. When it awakened, it sent the infected Mac's model name and UUID to the control servers, over Tor. The executable kept trying to connect to the servers until a response was received. The response was encoded in Base64 and contained a RSA public key and the instructions for ransom payment. The payment had to be done through a specific Tor network and to a specified address. Next, the malware started to encrypt the files under /Users and /Volumes (including Time Machine backups) directories with the received key.

To stop its spreading, Apple revoked the Gatekeeper signature and updated X-Protect, but several thousand unsuspecting Mac users were still affected (Xiao, 2016).

EvilQuest

The most recent and complex Mac-OS ransomware appeared in 2020 and it was called EvilQuest by researchers at Malwarebytes Inc., who renamed it a few days later as ThiefQuest. It was an evolving hybrid threat that combined ransomware, spyware, and data theft capabilities. This malware was distributed through malicious installer files for pirated applications. Obviously, the original apps were clean, only the applications pirated by Trojans had embedded malware. As the applications were pirated, a valid Apple Developer ID wasn't available, so the affected users received a warning message when trying to run the application but had the option to ignore the warning and launch it anyway. After being launched, it started encrypting files on the targeted system, and eventually directed the victims to a simple ransom note on their Desktop.

Luckily, the encryption was done using a symmetric key encryption algorithm, which is a weak option, so an encryption tool became public within weeks after the malware emerged. It turned out that, when encrypting a file, the virus also appended the encrypted original key and the key used for encrypting it. The binary also contained the function responsible for decryption, so the researchers performed reverse engineering on it and got the original encryption key, which could then be used for decrypting the victims' data.

Even if, in the end, the ransomware capabilities of this piece of malware turned out to be limited, after a closer look, it turned out it also had some other interesting characteristics. The installer contained a binary called ‚patch' which was moved to the /Library/mixednkey directory, set to executable and launched. As the installer requested root privileges during the install, the malicious script would also have root privileges. After analyzing the script by means of different static analysis tools, it turned out

it might also have key-logging capabilities (as _CGEventTapCreate, _CGEventTapEnable symbols were present), anti-analysis capabilities (as _is_debugging, _is_virtual_mchn symbols were present) and spying capabilities (because of __ get_host_identifier, __get_process_list symbols).

After reverse- engineering was performed on it, the researchers found that the malware only performed a sandbox check and not a virtual machine detection check, which made it easier to analyse it. After moving on to the dynamic analysis phase, it was revealed that the malware was also trying to become persistent by establishing itself as a launch item. Moreover, it tried to exfiltrate sensitive data to the command and control servers, while also trying to disable specific third-party security tools (Wardle, 2020).

RANSOMWARE DETECTION

Even if the user respects the general security guidelines (keeping the software updated, not opening attachments from unknown senders, backing up relevant data), there are some third-party security protection tools which are adding an extra layer of protection, especially if the given computer stores sensitive data or if it is linked to sensitive networks. In the context of ransomware protection, it's important to notice how this kind of malware is detected and which are the upsides and downsides of each approach.

Signature-based detection

Signature-based ransomware detection is the one used by X-Protect itself. The security solution holds a list of already known static elements (hashes, domain names, IP) related to the respective malware and compares those to the ones specific to the scanned item. With this approach, it is vital to keep to signature list updated with the latest discovered viruses. The main problem of this type of static detection is that it only works for already known and studied malware, leaving the door open for newly created malware.

Behavioral detection

This type of dynamic detection works by observing how a piece of malware behaves without any other prior specific knowledge about it. In the case of ransomware, it can be monitored if there is any abnormal behavior, like opening and encrypting a significant number of files in the file-system. The inbound and outbound network traffic monitoring can also be useful, as ransomware usually communicates with a remote server.

A behavioral detection-based scanning engine is a very complex software, but a simpler ransomware scanner for Mac-OS could be pictured in the following way: the scanner is informed every time an operation is performed on the monitored files, together with relevant data about the responsible process (PID, signature). If there is a large number of file operations from a given, untrusted process, the process might be a malicious one. Another solution is to use similar data in a machine learning model which is going to further decide if the behavior is malicious or not (van Mieghem, 2016).

Both approaches would need access to operating system specific information. More precisely, the proposed solution should firstly monitor I/O file events. It should also be able to determine if a file operation is in fact an encryption operation and if a process is trusted or not.

File system monitoring

One way to monitor file system events is using the FSEvents API. This API allows to register for receiving notifications regarding changes of a directory tree. When the monitored filesystem is changed, the kernel sends notifications via a device file (/dev/fsevents) to a user-space process called fseventsd. The API allows the client to register with a given list of monitored file operation types and a callback which is going to be performed when an operation of the specified type happens to a monitored file. Still, this method has a relevant weakness. The delivered file events contain information about

the process responsible for that given event, but the information is limited to the PID of the process, which is not always enough, especially in the case of a security tool, which might need the corresponding path and code-signing information for that process. There are ways to determine other information about a given PID, but they are not reliable, especially if the process might be terminated quickly (Wardle, 2016).

Another way to monitor the file system is based on Apple's Endpoint Security Framework. This framework is available starting with Mac-OS Catalina and it aims to keep the Mac-OS security developers out of the kernel. It has replaced the need for kernel extensions, previously used for developing security tools and it added a security plus, as it provides a user-mode interface for monitoring system events, instead of the kernel-mode one used by its ancestor. This framework provides comprehensive process (including code-signing) information for all events, which solves the highlighted weakness of FSEvents and it allows a proactive respond to file events (meaning that the integrator could decide to either block a process or not).

The received information about an event includes: the type of the event (create, write), the path of the affected files, the PID of the process, the process path and any process code-signing information.

Process identity validation

After receiving data about a process which is modifying files, including its path and code-signing information, it is important to check if it is signed by a trusted source. This is doable, based on different Apple Code Signing Services like SecStaticCodeCheckValidity.

Encryption detection

From the perspective of ransomware detection, while the file system is being monitored for changes and there is access to extensive data regarding the process which is causing these changes, it is vital to also be able to detect which changes are encryption operations, as this would be the main red flag regarding ransomware malware.

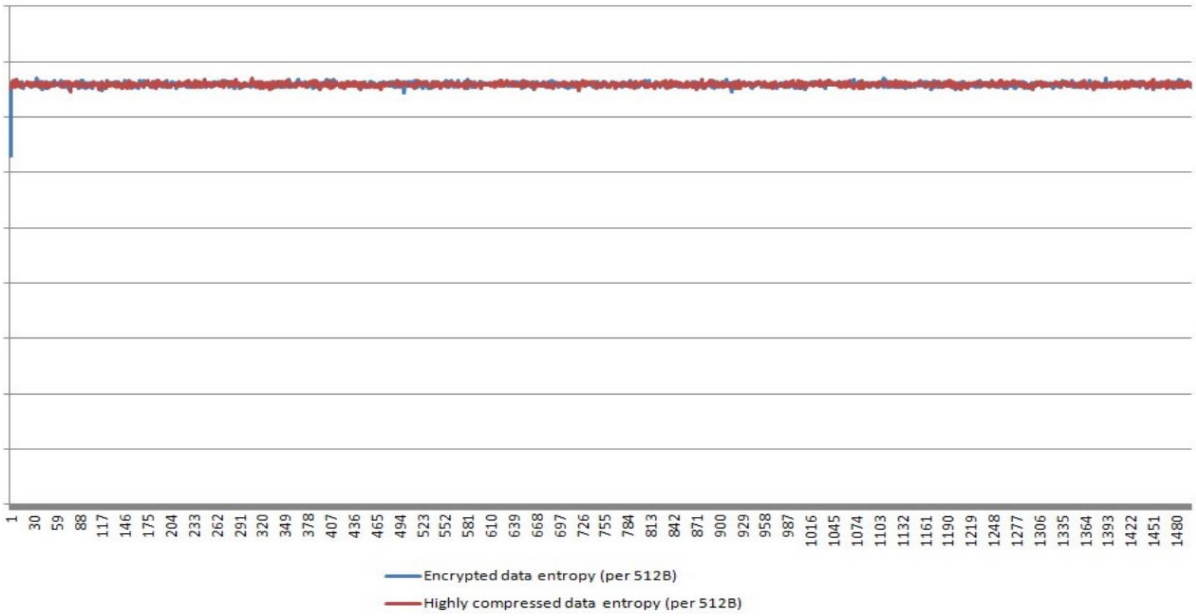

Figure 3: *Information entropy for encrypted and highly compressed data (Tšikul, 2019)*

There are multiple possibilities to detect whether a file was encrypted or not. The first one is based on entropy. This term was first used in statistical thermodynamics and refers to the amount of uncertainty or disorder. In Information Theory however, entropy of data is an average measure of some stochastic system which defines how much information it produces (Shannon, 1948). A basic approach for detecting encrypted files consists in using entropy calculations. However, while this approach successfully distinguishes between encrypted and unencrypted files (Figure 4), it is not able to reliably tell the difference between an encrypted and a compressed file, like a .zip archive, as both feature high levels of apparent randomness (Figure 3) (Wardle, 2016).

In 2013, Craig Heffner, a well-known security specialist and vulnerability researcher highlighted the fact that encrypted data is more stably distributed by entropy while compressed data tends to have some deviations. He researched two randomness tests, namely Chi-Square and Monte Carlo Pi in order to identify the claimed deviations (Tšikul, 2019). The results seemed promising and were confirmed by the Mac-OS security researcher, Patrick Wardle, in his proof-of-concept mac-OS ransomware detector (Wardle, 2016). Still, some weaknesses were identified: one of them is related to the fact that it only took into account the total number of deviations occurring in a file while completely ignoring other factors, such as file size and entropy distribution levels. (Tšikul, 2019)

More recently, other statistical tests, like the NIST suite and HEDGE have been used for distinguishing between encrypted and compressed content, by testing the randomness of its byte structure. The byte structure of an encrypted file is a true pseudo-random sequence, while the byte string of a compressed file maintains a structure and only approximates a randomly generated stream (De Gaspari et al., 2022)

Another promising approach for detecting if a file is encrypted or compressed is by using machine-learning algorithms. Various experiments regarding this problem were carried out. Some are using classical supervised machine learning algorithms, like Naive Bayes or Decision Trees (Ameeno, Sherry & Gagneja, 2019), while others are using deep neural networks architectures (De Gaspari et al., 2022). The latter seems to be the most promising for now. The deep neural classifier proposed by De Gaspari et al., 2022, reached an accuracy of ~90% on a complex dataset containing over 400M sample fragments and 16 possible compression formats (zip, gzip, rar and others). The same paper suggests that a machine learning approach or a hybrid one seems to have a significantly better potential than a purely statistical, entropy-based technique to successfully distinguish between encrypted and compressed data and could successfully be used for a ransomware detection tool.

CONCLUSIONS

This article firstly described the big-picture steps of a ransomware attack and then it discussed the Mac-OS built-in anti-ransomware mechanism, which includes, but is not limited to X-Protect, Malware Removal Tool, Gatekeeper, notarization, the signature system and Time Machine. It also further analyzed the main known Mac-OS ransomware malware and highlighted different ways to detect and protect against such attacks from a third-party point of view. This analysis is relevant, especially in the current context, with Mac-OS devices becoming more and more prevalent, mainly in the enterprise environment. Even if, at present, Mac-OS ransomware attacks are a pretty rare incident, it is important to be aware that such malware exists and that Mac-OS is not immune.

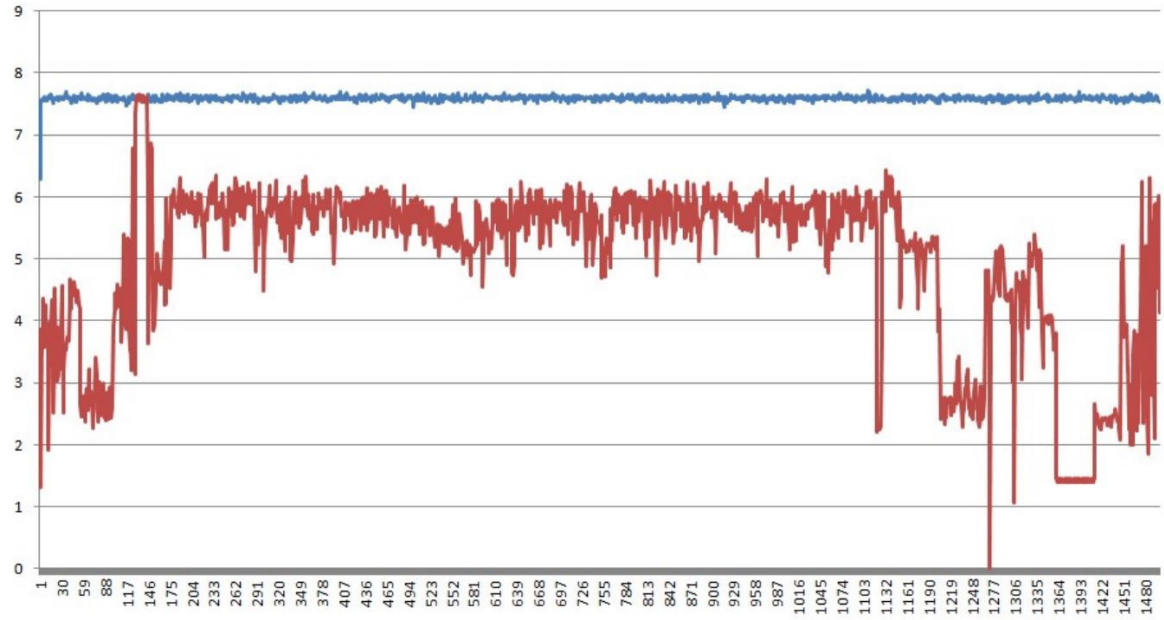

Figure 4. *Information entropy for encrypted and unencrypted data (Tšikul, 2019)*

REFERENCE LIST

AV-TEST Institute. (2022) *Total amount of malware and PUA under mac-OS*. AV-TEST Institute.

Canalys. (2022, January 12) *Global PC shipments pass 340 million in 2021 and 2022 is set to be even stronger.* https://www.canalys.com/newsroom/global-pc-market-Q4-2021 [Accessed 25 January 2023].

European Union Agency for Cybersecurity. (2022) *ENISA threat landscape for ransomware attacks* https://www.enisa.europa.eu/publications/enisa-threat-landscape-for-ransomware-attacks [Accessed 25 January 2023].

Dubey, S. (2022) *Uncovering the security protections in MAC – XProtect and MRT.* https://nixhacker.com/security-protection-in-macos-2 [Accessed 25 January 2023].

Johnson, A. L. (2015) *Proof-of-concept threat is reminder OS X is not immune to crypto ransomware*. https://community.broadcom.com/symantecenterprise/viewdocument/proof-of-concept-threat-is-reminder [Accessed 25 January 2023].

Kapoor, A., Gupta, A., Gupta, R., Tanwar, S., Sharma, G. & Davidson, I. (2022) Ransomware Detection, Avoidance, and Mitigation Scheme: A Review and Future Directions. *Sustainability*. 14(1), 8. doi:10.3390/su14010008.

van Mieghem, V. (2016) Behavioural Detection and Prevention of Malware on OS X. *Virus Buletin.* https://www.virusbulletin.com/virusbulletin/2016/09/behavioural-detection-and-prevention-malware-os-x/ [Accessed 25 January 2023].

Shannon, C. E. (1948) A Mathematical Theory of Communication. *The Bell System Technical Journal*. 27, 379-423.

Tšikul, P. (2019) *Encrypted data identification by information entropy fingerprinting.* Master`s Thesis. Tallinn University of Technology, School of Information Technologies.

Wardle, P. (2016) *Towards Generic Ransomware Detection.* https://objective-see.org/blog/blog_0x0F.html. [Accessed 25 January 2023].

Wardle, P. (2020) *OSX.EvilQuest Uncovered* https://objective-see.org/blog/blog_0x59.html [Accessed 25 January 2023].

Xiao, C. (2016) *New OS X Ransomware KeRanger Infected Transmission BitTorrent Client Installer:* https://unit42.paloaltonetworks.com/new-os-x-ransomware-keranger-infected-transmission-bittorrent-client-installer/ [Accessed 25 January 2023].