UNIVERSITATEA “ALEXANDRU IOAN CUZA” IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE DISERTAȚIE

File clustering based on EULA

propusă de

Ciofu Vlad Teodor

Sesiunea: februarie, 2020

Coordonator științific

Conf. Dr. Dragoș Teodor Gavriluț

UNIVERSITATEA ALEXANDRU IOAN CUZA IAȘI

FACULTATEA DE INFORMATICĂ

File clustering based on EULA

Ciofu Vlad Teodor

Sesiunea: februarie, 2020

Coordonator științific

Conf. Dr. Dragoș Teodor Gavriluț

Avizat,

Îndrumător Lucrare de Licenţă

Titlul, Numele şi prenumele _____

Data _____ Semnătura _____

DECLARAȚIE

privind originalitatea conţinutului lucrării de licenţă

Subsemntatul(a) ……

domiciliul în ……..

născut(ă) la data de ……., identificat prin CNP ……., absolvent(a) al(a) Universităţii „Alexandru Ioan Cuza" din Iaşi, Facultatea de …… specializarea ……., promoţia ……., declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul:

_____

_____

_____elaborată sub îndrumarea dl. / d-na _____, pe care urmează să o susţină în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice in vederea facilitării fasificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, …… Semnătură student ……

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*File clustering based on EULA*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*

Absolvent *Prenume Nume*

_____

(semnătura în original)

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil şi sursă, să aparţină autorului prezentei lucrări, *Ciofu Vlad Teodor.*

Încheierea acestui acord este necesară din următoarele motive:

Lucrare de față este realizată folosind date confidențiale din cadrul companiei Bitdefender. Incheierea acestui acord este necesară deoarece codul sursă se află sub o clauză de confidențialitate între autorul lucrării și compania Bitdefender SRL

Iaşi, *data*

Decan

_____

(semnătura în original)

Absolvent

_____

(semnătura în original)

Table of contents

1. Malware

Malware is simply defined as malicious code, software designed with the purpose of damaging the system, or the effect of which has harming effects. This is a program that may affect or interrupt the system's functionality, making way for an attacker to get acces to confindential or sensitive information, as well as the ability to spy on personal or private systems.

An intrusive and vicious malware tries to invade, disable or even destroy computers, mobile phones, system networks and tablets, often times through partial controll over certain operations of a device. Much like the human flu, the malware infects the host and makes it behave unnaturaly.

The main purpose of malware is ilicitly making money off it's target. Although in most cases malware cannot alter the physical parts of the computer or of the network, the hardware, it can encrypt, steal or even delete data, spy on the system's activity or change basic functions of the computer without the computer user's knowledge.

Cyber-criminals often program malware in a specific way so that it stays persistent, that way it can stay on the targeted system for long strands of time without the user knowing or consenting to it. The maliciously intended software disguises itself, most of the time as clean programs, that belong to the system.

While the effects of said type of malicious software are damaging for a user, they can have devastating effects for companies. If spread through a network, the malware can provoke damage and large scale downtime, which may require great effort on the part of the company to come back from.

The malware spectrum is wide and gets wider by the minute.

1.1. The history of malware

The history of malware goes a long way back, as you may imagine. The term of malware was introduced by Yisrael Radai back in 1990[4], but those type of threats existed decades before, called computer viruses. It all started as experiments or jokes, but evolved into much more. Nowadays, hackers are using malware to steal personal, financial or business information. Even worse, it is believed that government agencies use those practices to have acces to secrets.

Each and every day, new malware appears, and the worst part is that it changes almost every time. Every time a new one appears, it is more complex.

A full history of malware would be too long a list, taking into consideration the massive number of types and variants that are launched every day. Those being said, an outlook over malware tendencies in the last couple of decades is much easier to manage. Those are the main tendencies in malware development [5].

Starting with 1980 and the following years: The theoretical basics of automatic reproduction software, also known as viruses, date to an article published in 1949, and the viruses of the time made an appearance on the computerized platforms in the 1970s. The history of modern viruses starts with a program named Elk Cloner, which started infecting Apple II systems in 1982. The method of propagation being infected floppy disks, the virus itself was harmless, but would spread onto every floppy disk attached to the system. Because of it's virulent break, it could be considered the first widespread infection in history. Remember that this was before any malware on Windows PC. From then on, viruses and worms became widespread.

The 1990s: In this decade, the platform Microsoft Windows came to be, along with the flexible macros of it's applications, which determined malware authors to write infectious code using Microsoft Word macros. What those macros do is infect documents more than executable applications, although all in all, Word macros are executable code

2002 – 2007: instant messaging worms, malitious code that self replicates, spreading

through an instant messaging network, are taking advantage in the lack of security to spread on a large scale, infecting AOL AIM, MSN Messenger and Yahoo Messenger.

2005 – 2009: Adware attacks became more prevalent, showing unwanted ads on the screen, somtimes as a pop-up or in a windows that couldn't be closed by the users. Those advertisements exploit in most cases legitimate software as a means of spreading, but around 2008, software developers have started taking adware companies to court for fraud. This resulted in millions of dollars in fines, which prompted the adware companies to close down.

2007 – 2009: Malware creators took to MySpace, using it as a platform for social engineering, to spread their code. They started creating false advertisements, fake antivirus and security solutions and spreading them to the users. After MySpace's popularity went out, the prefered platforms became Facebook and Twitter. The main methods were promoting facebook aplications with malware extensions and presenting false links to phishing pages. As the companies took action to solve those problems, the tendency went out and migrated onto other things.

2013: A new face of malware appears named ransomware, which launced an attack under the name of CryptoLocker, which started in september 2013 and up untill the end of may 2014, targeting Windows computers. The attack managed to make 27 million USD from ransoms paid by the victims, the figure coming in the last trimester of 2013. The succes of CryptoLocker spawned other ransomware, named slightly similar. Another variant of the malware has made over 18 million USD from around 1000 victims in around a year.

2014 – 2017: The top ranked malware are ransomwares, trojans and exploits, with ransomware taking the crown, having great outbreakes in 2017 which have affected businesses of all kinds, draining them of their money.

2017 – present: Cryptocurrency came into the attention of malware developers, and like that cryptojacking was invented, a word used for taking control of a computer and using it to mine diverse cryptocurrency without the user's consent.

1.2. Spreading techniques

When it comes to malware, there are a lot of spreading techniques. The most popular methods being the following.

1.2.1. Drive-by download

Drive-by download has 2 main definitions:

The first one says that a drive-by download is a download authorised by the user but did not fully comprehend what will be downloaded automatically.

The second one refers to any download happening without the user knowing of it, and most of the time this definition refers to malware of any kind.

Drive-by downloads could happen when clicking a link or popup, or even when browsing a website or clicking a mail attachment. Some sites may even try to trick the user by impersonating operating system errors, and when clicked the website pretends that the user has approved of the dolwnload, and shall start the process of downloading the malicious, or at the least unwanted file. Quite similarly, the download could happen when a visited website, browser or plugins have exploitable vulnerabilities that allow running malicious code without the user's consent.

One of the reasons for the increase in drive-by downloads is the increased number of exploit packs released. Those are easier to use than writing the code or finding the exploit, allowing almost anyone to create this kind of attack. Having the exploit kit, the next step would be to have the malicious code ready to be downloaded, but that requires a host. While the host could be a self made server, in most cases it is a legitimate website that either has vulnerabilities or is already compromised, or even trough ads on such a website. By the time the conted of the compromised website is loaded by the client, the attacker will fingerprint the client to adapt the

exploitation code for the specific vulnerabilities found on the client's browser or machine.

Once all of the preparations are complete, the attack follows one of two main ways of running the code. Either it uses a component of an installer that insecurely downloads other components from the internet, allowing the attacker to make his own file to be downloaded, or loads a shell code into memory through abuse of browser or plugin vulnerabilities and redirecting the code flow to the malicious shell code. After the code has been executed, it has most like left a backdoor, and from there the attacker can install more malware or even start stealing information from the target.

On top of all this, whitch is already quite hard to detect, the atacker can obfuscate it's code, which is sometimes done using IFrame. The obfuscation impedes detection, but there is also encryption. Sometimes the malicious code is encrypted and the decryption code is left at the end of the malicious file, do extract the payload and run it.

1.2.2. Homogeneity

To make homogeneity easy to understand, it is better to think of a network of computers as a chain of systems. If all of the computers have the same os, and that os has a vulnerability, the attacker can easily compromise the whole network, since it can pass trough each link of the chain. This can be combated by having other operating systems in your so called chain. Having even one different operating system in the chain prevents the attack to propagate with the same vulnerability.

1.2.3. Vulnerability

Vulnerabilities are security overlooks in software, which can further be exploited by an attacker or malware. It could be a coding error or overlook, or any kind of inherrent weakness in software implementation, in an application or operating system.

1.2.4. Backdoor

A backdoor is a way to bypass authentication or encryption in an informatic system, be that a system or a network. Backdoor are often used for remote controlling a system over the internet and accessing sensitive information like passwords or personal files on a victim's computer, transfering them to a cloud or similar service.

A backdoor can be either a piece of code hidden inside some software, in the firmware of a hardware or even in the operating system like Windows. Backdoors are usually created by trojan horses, software that pretends to be regular software untill it is executed, moment when it triggers malicious code and installs a backdoor and what other payload it has. Although backdoors are mostly installed in secred, other backdoors are delibaerate and well knows, just like remote connection clients that allow acces onto a user's computer through the network.

Predefined passwords could work as backdoors if they remain unchanged by the user, as those are the first that are tried in bruteforce attacks

.

1.3. Malware terminology

Botnet: A number of devices connected to the Internet that have been infected and partake without the user's consent in distributed denial of service attacks and spam mail sending.

Containment: The process of stoping the spread of malware and preventing the further deterioration of a host.

Payload: The part of the malware that is ran on the target's system to do the intended damage.

Privilege: The limit of a user's ability to modify the system or network it is part of .

Signature: Specific signs either for a certain behaviour or an element of certain malware.

Threat:. A system from a network becomes a threat the moment it has a persistent software vulnerability that can be exploited, increasing the risk of an attack.

Track: Proof of an intrusion in a system or network. Malware can clear directories, remove event logs and hide network traffic to cover it's tracks.

Zombie: A computer connected to the internet that has been compromised by a hacker or trojan horse and can now be used for malicious actions.

1.4. Different types of malware

The word malware has a bad connotation by design, most proeminently it starts with the prefix „mal", which means bad. Also, the term malware is explained easily in 3 words: badly intended software. The definition is wide, and that is mainly because of it's many types it can be categorised into, more and more appearing exponentially as time goes on and humanity is more and more reliant on technology.

Recent studies show that most of the mawlare that is in the wild nowadays are worms and trojans, with viruses having a diminishing number. A study from 2011 showed that 69.99% of tracked malware were trojan horses, while viruses made up only 16.82%.

One of the more recent studies, made in 2017, observed that malware designed for mobile devices like smartphones and tablets have skyrocketed lately, and even come preinstalled on some devices.

1.4.1. Viruses

Viruses came to be around the same time as computers. The first academic paper on the theory of self replicating software was made in 1949 by John von Neumann. The first examples of what could be classified as a virus have been detected around the 70s.

One of the main caracteristics of viruses would be reproduction, meaning that it has the ability to self replicate it's malicious code by any means. Another would be that they are hidden, making finding them on the system difficult for the user, without a dedicated security solution called antivirus. All in all, they appear uninvited and burrow into the system, working without the user's knowledge and that's what makes them so dangerous.

It's hard to even call a virus a file, since the virus infects a host file, injecting it's code into it, so mainly the virus is the malicious code hidden in all of the files. The downside to this approach to propagation is that the malicious code needs to be ran by the user so it continues to spread. Viruses cand be further classified by the region of the file or the specific files they infect.

One of the more rare types of viruses are the ones that load themselves into the RAM memory, installing themselves as part of the operating system, from the moment the system is started untill it is shut down. The rarity comes from the security measures already built in the newer operating systems, that would not allow such behaviour.

The better part of viruses are the ones that hide in generic executable files with the purpose of having greater chances of being executed by the user. Into the same category fall the viruses that infect script and batch files, for the same main reason.

Although less prevalent nowadays than a few years ago, there are viruses that infect Microsoft Office macros. The decrease in prevalence comes from the way macros start, previously being ran when the document was opened, but nowadays must be activated manually by the user.

Certain viruses have been the cause of large scale destruction, some of which being Concept, Cernobyl, also known as CIH or Spacefiller, Anna Kournikova, Brain and RavMonE.exe

1.4.2. Worms

The second type of infectious malware is the worm. A worm disregards infecting any type of file and simply replicates itself over and over, while destroying files and other data on the system. Most of the time worms target operating system files and run untill the drive it is present on is emptied.

Worms often spread through email and instant messages, and can't do much untill executed on a computer. They use the network to spread locally after being ran, exploiting the network's or the target computer's vulnerabilities and then accees and delete all the user data.

Some worms are not created with spreading over the network in mind, but even those are nefarious by using up computer resources and network traffic.

Some examples of worms include Morris, Sasser, Blaster, Melissa, Mydoom and Mylife.

1.4.3. Trojan Horses

Using the term for the wooden structure used by the greeks to invade Troy, the trojan horse, or simply the trojan, is a type of malware that disguises itself as being usefull so it will be installed by the user on their computer.

The payload of the trojan could be anything, but most of the time it is some sort of backdoor that allows unauthorised acces or remote controlling of the infected system to the attacker. After the backdoor is in place, the attacker then has acces to personal data like the IP address, passwords and bank accounts the user has previously saved or will use in the future while the trojan is present. That can be accomplished by installing keyloggers that capture every keystroke of the user, some of which even sending screenshots taken and are virtually invisible.

Among current malware, trojans are considered to be the most dangerous out of all, especially those made to steal the banking information of a user. Some trojans even pretend to remove malware from the system while downloading and running them on the system.

Some notable examples of trojan horses being Magic Lantern, FinFisher, WARRIOR PRIDE, Netbus, Beast, Tiny Banker and Zeus have been some of the most notable.

1.4.4. Rootkits

A rootkit is a bundle of software specially designed to allow malware to gather information from a system in the background, without making the user aware of it. While running, the rootkit will allow many more types of malware to make way into the system.

Much like the backdoor, their effects are similar, allowing attackers acces to a computer without the user's acces, and just like a backdoor, it's instalation could be manual when a system is compromised or automatic as a payload of a trojan.

What makes this type of malware different is how deep it burrows itself into the victim's computer. The removal of a rootkit is extremely complicated, hardly possible at times, especially if the rootkit is inside the kernel of an operating system. Reinstalling the operating system is often times the best solution to fully get rid of an advanced rootkit.

The first famous Windows rootkit was NTRootkit, back in 1999, but the most famous was the Sony BGM rootkit that has caused issues for the company in 2005.

1.4.5. Ransomware

Ransomware is the most devastating and prevalent type of malware nowadays. Also being one of the most advanced, ransomware blocks the user from accessing it's own files by encrypting them and promising the decryption is done after a price has been payed. The worst part is that there is no guarantee the files will be decrypted after the price has been paid.

Thid kind of malware infects the system from the inside, locking the computer and making it useless. The more rudimentary versions of ransomware simply blocks the computer untill the ransom is paid while the more advanced searches for files that are relevant to the user and specifically encrypts them.

Ransomware became prevalent in Russia, but has grown in popularity worldwide. It usually spreads as a payload from a trojan.

This kind of digital extorsion has existed since the end of the 80s, it has resurfaced at the end of 2013 because of the aparition of cryptocurrency, which is used for paying the ransom and is untrackable. Detecting and removing ransomware is a complicated process, it being classified as the most dangerous cyberthreat. Also, besides being prevalent on computers, it is also growing on the mobile market.

Some big ransomware names are Reveton, CryptoLocker, CryptoWall, and the more recent 2017 WannaCry. Also, for the mobile, the biggest name is Fusob, which has used scare tactics to get it’s victims to pay the ransom in exchange for decryption.

1.5. Financial tendencies

Although malware started out as a proof of concept and without any previous motive, people soon realized it’s money making capabilities and started putting it to use. Without a doubt, malware should be illegal, but creating laws that require specific wording and the agreement of lots of people is extremely slow in comparison to the speed at wich a new malware can appear.

As a consensus, the creation of malware is not illegal, but spreading it is. In the United States, the US Patriot Act (sec 814) offers punishment for those who damage or gain unauthorized access to a protected computer, causing financial or medical damages.

In the past, there have been cases of a conviction related to an infected computer used

in a larger scale attack, which simply goes to show just how hard it is to track a malware attack. Alas, if the actual creator of the virus is caught and convicted, most of the time said person will not have the necessary money to pay for all of the damage the malware he created and spread did, which in cases amount to billions of dollars.

Taking into consideration how easy it is to avoid the law, here are some of the creative ways malware creators have thought up to make money.

1.5.1. Cryptominers

To understand cryptomining we must first define and understand what cryptocurrency is. Cryptocurrency is defined by the following principles.

Digital: the existence of cryptocurrency is only on the internet, there are no physical notes or coins.

Decentralized: cryptocurrencies must be distributed over a network of computers that are numbered in the thousands at least. Such network without a central server is called a decentralized network.

Peer-to-Peer: cryptocurrencies must be transferred from one person to another, without a having to deal with a trusted third party, like a bank when it comes to regular money. It is called a trusted third party because you have to trust it with personal information in order to use their services.

Pseudonymous: there is no need to give personal information, as there are no rules as to who may own cryptocurrency.

Trustless: there are no trusted third parties, so the users are in complete control of their information and money at all times.

Encrypted: by the use of cryptography, every user has a special access code that allows them access to their information, and that other users don’t have access to

Global: cryptocurrency can be sent around the globe easily, much easier than flat currency

Cryptocurrency appeared in the early 1990s, created by a group called Cypherpunks. Their motivation was the lack of trust in the government and corporations and their opinion that said entities hold too much power.

The group wanted to give internet users more control over their money and information. Their attempts at this were DigiCash and Cybercash, two digital money systems that failed by the end of the 90s. Both of those had almost all of the principles listed above, but neither had all of them.

The big break in cryptocurrency happened in 2009, when the alias Satoshi Nakamoto, whose identity remains a mystery up to this day, although speculations have been made, created Bitcoin. Bitcoin succeeded where the other 2 cryptocurrencies failed, and its success came from using the blockchain technology.

The blockchain technology is what bitcoin uses to keep track of all the transactions that have ever been made. All the transactions are public for the whole world to see, and new blocks are added to the chain the more transactions are made.

The blockchain database is stored on a network with multiple systems, the bigger the number the safer the blockchain. All the computers in the network have the database, so that in the event one gets compromised by an attacker, the data would only be modified on the one system, and when compared to the data from the other computers in the network, the problem will be detected and the previous unaltered database will remain.

The way payments are made coincide with the malware to be further discussed, cryptomining. Mining cryptocurrency is, if extremely simplifying it, checking if transactions are correct, eliminating the risk of double payment, a big problem in cyber trading. This process is more or less guessing for a correct hash, which requires a lot of computing power. The computing power is not used for guessing one single hash, which is quite easy to do, it is used for guessing a great amount of hashes. The winning miner that has found a hash that works, receives an amount of coin.

Although miners could work by themselves and hope to manage to add a block to the chain, and some do, what they prefer is to attach themselves to a network. This mining network distributes the workload between the miners, increasing the chances of attaching the block to

the chain and collecting the reward. After collecting the reward, the network splits the earnings between the miners, proportionally to the amount of work they have put in for the block.

Cryptominers, after being left on a target's computer by a trojan payload or a previously left backdoor, start using the computing power of the system to mine in a network. Although a regular user may not have a system with a lot of processing power, the purpose of cryptominers is to spread on as many computers as possible.

The user infected with a cryptominer may find the system running very slowly, as that is a dead giveaway of the miner, the high usage of CPU. But since their first appearance cryptominers have evolved.

The malware creators have realized how easy a miner is to spot, so they limited the CPU consumption, so the whole mining process goes unnoticed. By sacrificing computing power, they extended the ability to hide, useful for the malware, less so for the user who unknowingly generates money for the attacker.

1.5.2. Ransomware

Ransomware's popularity and effectiveness has been brought up by the apparition of Bitcoin. The reason behind it is in the basic principles of cryptocurrency, it being pseudonymous. What that means is that everyone can own and pay with the cryptocurrency, without having to provide personal information.

The malware creators use this principle to collect money without being tracked by the police. If the process would use flat money, the risk would exponentially increase, as opening an account would require more than enough personal information to catch and convict someone.

As previously described, what ransomware does is it makes its way onto a target's system, often times as a trojan's payload, then locks the user out of his computer, promising to unlock the computer only if a certain price is paid. The newer ransomware doesn't lock the user out, but seeks the user's personal files and encrypts them, leaving a ransom note behind.

The ransom note usually contains a set of instructions on how to pay the price for unencrypting the files. Along with those, a code to the attacker's bitcoin wallet, which as

previously said, is untraceable. The user is told or directed to a site where he may learn what bitcoin is and how to purchase it, only to then send it to the attacker and wait for his response.

Further attempts at finding what has caused the issue after the ransomware has been ran are met with confusion, as it usually deletes itself after the encryption is done.

1.5.3. Clickers

Clickers are another kind of malicious program that are usually spread as trojan payloads. They run silently in the background, just like most regular malware, but their purpose isn't to damage the computer, but to access certain sites once in a while, producing advertisement revenue to said sites, most of the time owned by the creator of the clicker.

Only one of this type of malware is barely noticeable, since all it does is consume minimal resources and network traffic, but the problem arises when there is more than one or the clicker is aggressive. With the computing power everyone has nowadays, having more than one program that simply accesses a webpage, or multiple ones, is not exactly the end of the world, just a mild inconvenience.

The real problem appears in the consumption of network bandwidth. In countries with better internet infrastructure, in which internet speed could reach up to Gigabit speed, this problem is obsolete, but having in mind that some countries have very low internet speeds, and getting faster internet may not be the best financial decision for the common user, every little extra connection may slow down regular internet browsing.

Even if the user is mostly unharmed in such an attack, the ones at a loss are the advertising companies whose ads are not seen, only mindlessly accessed by the malware, providing money to the creator of the clicker and thinking regular users see their ads.

2. Grayware

The concept of grayware refers to the borderline between clean software and malicious software. Grayware tries to convince the user that it is honest and trustworthy, most of the time through its end user license agreement, abbreviated EULA, or by marketing strategies.

The main drive of grayware, just like malware, is to illicitly make money off the user or steal personal data. The difference between malware and grayware money making is that malware does it without the user's knowledge, like clickers, or consuming resources, like cryptomining, but grayware tells the user everything, or in the shadier grayware, just parts of what it does, but mostly in fine print or in subtle ways that may easily be overlooked.

Grayware is usually distributed by bundling with other legal software packages including freeware and shareware, which obtain users' consent for installation via an EULA. Unfortunately, the lengthy EULAs presented by such packages are typically vague and deceptive on functionality descriptions.

The reason they tell everything they do is to avoid legal action. The moment a user would want to sue a company for the effect their product had on his computer, the company's legal team would simply point out that he agreed with their terms in the first place, which the user was most likely not even aware of, but also didn't bother to find out even if it was before his eyes.

2.1. Grayware types

Because of the fuzzy line between clean and malicious software, some grayware behavior is simply malware with extra steps, while other are way more on the clean side. The categorization of grayware is not made by separating clean from malware, or at least placing

them on the spectrum. The categorization is made by grouping them based on the way they make money or get data off the user.

2.1.1. Deceptors

Deceptors are software that try to get the user to buy the full product by offering a very stripped down version of itself as trial. The moment the user installs the trial, he is prompted scary messages, like having the program detect a great amount of errors with the system, and telling him all of those errors could be fixed by buying the full product.

The problem is, most of the time the problems the deceptor finds are exaggerated and made to look like errors, so the regular user would be deceived into thinking his computer has problems, when the only problem is the deceptor itself. After paying for the full program, the software removes all the pretended errors from the system and the lied to user feels safe again.

As you may have noticed, there is nothing malicious actually done to the user's computer, maybe actually some optimizations are made, so why is this grayware? It's because of the overemphasis on buying the product and less on actual issues and fixing them. The only one having to suffer is the user falling for a malicious marketing strategy, which is making him believe he has a problem only to be sold a solution.

Deceptors vary in how hard they try to scare the users into buying the software. The lighter deceptors may simply emphasize issues, while the more aggressive ones could employ other tactics.

One technique used by deceptors would be making the uninstallation more difficult. The moment a user would click the uninstaller, uninstallation would seem to proceed as normal, but if paying close attention, the text would be slightly changed.

The user would be prompted to press the continuation button if he wants to keep the software on the system, and to press a not so obvious button, sometimes disguised and with writing almost the same color as the background, to continue with the uninstallation. After running the uninstall, the user would forget about the program, only to be greeted by it at the next computer startup, or when the deceptor scheduled itself to run.

In some cases, the user may get more than what he expected, as when installing the app and more than one software gets installed. The bundling together of deceptor apps is very common, the more aggressive of those without even asking the user for permission.

In the ones that do ask for permission, there are cases where the offer to install extra software is in fine print, or looks organically like the installer which prompts the user to think it is part of the installer and glance over it.

The movement against deceptor applications has been on the rise recently. Sites like AppEsteem do everything in their power to separate the good from the bad when it comes to deceptors.

AppEsteem researches cleaner or optimizer applications, calling their support lines and checking every nook and cranny of the app in order to see if it can be categorized as a deceptor or not. Since grayware has a component of subjectivity inherent in it, the site has developed a set of rules to try and minimize the more personal component in identifying deceptors.

Beside identifying if an application is a deceptor or not, AppEsteem contacts the company in an attempt to certify the app, for a price. The contacted company is presented the rules it violates that may classify its app as deceptor and requests that the software be changed.

If the company decides that it wants to change the app, then after the newest version of the software is released it is evaluated again, and if no deceptor rules are found present, then the app gets certified by them.

Of course, some companies use this strategy willingly, and knowing their product has no real use whatsoever and it will not be bought they refuse to change when contacted about the rules they break. Sometimes, those companies opt to rebrand the product or even make a completely new one, hoping that the user will not notice the same behavior pattern, and that it will go beneath AppEsteem's radar long enough to make a profit off unknowing users. In those cases, the user should also have in consideration the company name that released the application and check its reputation.

Some grayware companies even choose to fully rebrand, name and everything, to further deceive users and escape their bad reputation. This mask easily falls off when checking the

company name on the internet, but a lot of users do not have the time or experience to do their research themselves, falling in the trap the company set.

As there can be no good thing without the bad, there are a few issues when it comes with exposing a set of fixed rules to define deceptors. The issue is the same when it comes to any set rule, corner cases. As long as a deceptor creator knows the rules, he can try and model his app in a way that respects all of the rules but still is able to scare the user.

The only way to combat this is to make the rules as well and as restraining as possible, at times maybe a little bit less reasonable to the company in favor of the user. Otherwise, it's just a matter of time before the deceptor creators find other loopholes, and AppEsteem covers them, which has no gain for the user who is simply trying to not spend his money fruitlessly.

2.1.2. Keyloggers

Some eyebrows may be raised to see keyloggers classified as grayware and not as malware to begin with, but there are numerous reasons for this placement.

Keyloggers are software that capture every keypress of the user and save it on the disk. Some loggers may even be configurable to send the data over the internet or through email. Besides capturing keypresses, some of the newer keyloggers take screenshots and record mouse clicks.

Without a doubt, there are keyloggers that are easily classified as malware. Malware keyloggers are apps that are spread through trojan horse payloads or by exploiting system vulnerabilities. The loggers are set to run at startup, in the background without a way for the user to interact with them, or even know anything about them. They are preconfigured by the attacker so that the software sends the captured information right back to him, be it through email or some other way. Of course, most of the time the email is a throwaway so there is not that much to go on to track the attacker, leaving the user with his data compromised and vulnerable to being blackmailed.

Besides being an invasion of privacy, the attacker may further use the gathered data for his own monetary gain. The more usual cases are blackmailing the user, by sending a mail with

enough proof to convince him that he has some information that the user may not want to be made public. From then on, the user sends money to the attacker, and as it goes with blackmailing, the user has no way of being sure that data will remain a secret or not.

In the less common cases, the maliciously obtained data can be used for monetary gains in other more complex ways, like stock investment. If a keylogger finds its way onto the system of a bigger company, this provides the attacker a great deal of information, not only on the company itself but also in the field of work of said company, allowing him to invest and make money without ever contacting the company, as his best interest is to stay hidden and collect the information in real time.

Although there are malicious keyloggers, that does not mean that the concept of keylogging can only be used for destructive purposes. A big lot of keyloggers are legitimate and used in big corporations, or on regular user's systems.

The first step into legal keylogging, would be the upfront installation. Unlike it's malware counterpart, that gets installed without the user's knowledge, the installation process with a clear interface for a clean keylogger would be paramount.

One of the things the installation process should contain is the EULA, so the user knows what he is getting himself into when installing the product. Although a strong tool for the grayware creators to hide behind, also as great for the patient user that reads through it. A good EULA is worded in such a way that a regular user would understand it, as grayware usually hides in complex terminology and twisted sentences.

Another thing to add to the clean keylogger recipe would be that the user is always aware of the running software. Malware keyloggers are hidden and have no intention of making themselves known. A clean one should at least leave a tray icon present to notify the user, and give details of what is happening when clicked.

The other very important part of clean keylogging would be uninstallation. A user should first be able to find the keylogger in the list of installed programs on the system, then to be able to remove said program from the system if required. The malicious counterpart does not appear in the installed software list, and would regularly be hidden in a system folder or somewhere the regular user would not have any idea how to get to.

Besides being able to find it, the uninstallation process is also important. The uninstall should not leave any traces of the previously present software, like running services or others that persist after a startup of the system, if required for the uninstall process to be complete.

Having presented both clean and malicious keyloggers, it is time to look at the elephant in the room, grayware keylogging. Just like regular grayware, those keyloggers exist in the space between the clean and malware ones, and usually having certain traits from both.

Often times, grayware keyloggers have a hidden mode, a way to run or even to install the keylogger in a way that it will be hidden after installation, having every configuration made at the beginning.

2.1.3. Adware

Adware is software made for the purpose of displaying advertisements on the user's system, and it is the most iconic grayware software. Besides displaying ads, it also may collect browsing data and click behavior, override other ads and create popups.

It is also found among malware, being dropped by trojan horses or other less common spreading means and creating unwanted ads without the user's consent. Although annoying, the adware that fall under grayware are much more of a pest.

Adware makes money off the ads the user watches. Depending on what the installed adware does, it could create popups in the browser with ads, replace existing ads or even display them in the operating system as tray popups.

Adware that falls under the umbrella of grayware does not spread the same way as malicious files, as it would further hurt their image and bring unwanted attention. The way it spreads is through bundling or by being included in certain freeware or shareware downloads. As certain adware software are legitimate, they have their own website, and can be downloaded directly from there.

2.1.4. Bundlers

Bundling is the most popular choice in spreading adware. It works by adding offers to programs into another software's installation. This happens when users try to download software, most of the time well known, from sites other than the official one. The installer from those sites is often not the official installer, but a modified version, with added offers.

At this point, it all depends on how aggressive the so called bundler is. The lighter bundlers may simply offer one or two other software, making the offers easy to see and distinguish from the normal installation process. Also, the lighter bundler will always make the buttons clear and easy for the user to understand what to press to make his decision, and not the bundler's.

The more aggressive bundlers employ certain tactics to confuse the user into installing the software they are offering. One of those techniques is to change the default installation buttons. If a regular installation would require the user to press a button labeled refuse in order to refuse the offer, the bundler would ask the user to press the refuse button to accept the offer.

Another way bundlers confuse the user using buttons is to recolor them. By making the refuse button almost the same color as the installer background, the bundler attempts to hide the option to refuse from the user, while still making it available and arguing that it is not with malicious intent.

One more way to deceive is to hide the offers they make from plain sight. The offers are hidden under a button tagged with custom install, and only then the user can see and decline the offers.

The most aggressive bundlers do not even ask for consent and simply install all the bundled software, sometimes even omitting the app the user was installing in the first place. This behavior is not even grayware, it is outright malicious.

2.2. Financial tendencies

As grayware encompasses a lot of programs and behaviors, some of the ways it makes money is similar not only to malware, but also to legitimate software. That being said, please keep in mind that the more aggressive grayware may use malware making techniques, these are some of the characteristic grayware making techniques.

2.2.1. Advertisement injections

Advertisement injections are byproducts of adware, software that changes the browser or even the operating system to show targeted advertisements to the user. Although it might not sound so bad at first, the more aggressive adware may multiply the amount of ads you see on a page by a big factor.

Adware can also take the form of browser extensions, which allow some innate control over the browser and what it's pages display, changing the html code of the loaded page.

After it is executed, the changes may be subtle, as the lighter adware might make only small changes. Such changes could be changing the ads that would have already appeared to the user with others. By doing that, the adware redirects the monetary gain for those specific ads to the attacker. Although in those cases the user does not have much to lose, the bigger loss comes for the visited websites that lose the advertisement revenue from the infected user.

Light as they may be, sometimes the user can be inconvenienced by replaced ads. Regular ads can be closed and reported if the content is inappropriate, a feature that most adware do not provide, for the obvious reason that it is not in their interest to lose money for the user's pleasant time on the internet. Such content may be of pornographic or unsavory nature, making browsing the internet an unpleasant endeavor, even less so for children left unattended with the computer.

For the more aggressive adware, waiting for the user to browse and replace the regular ads is way to slow a way to make money. They fix that by increasing the number of ads

exponentially, so much so that the user can barely navigate the internet without having to close popups every few seconds. And it doesn't limit itself to popups, it may even open new tabs with phishing messages. For example, a tab that impersonates an operating system error, which a user may be confused about, and then redirect the user to a fake support that remote connect to their computer and install malware like backdoors for further access and malware attacks.

The problem with the lighter adware is that the line between malware and legitimate software becomes blurry. Some users may actually opt for software that change the ads towards something they like or have previously agreed to. The problem with this is that most users look over one of the most important step in the installation of a product, which is reading the End User License Agreement (EULA for short).

Most of the time, adware companies explain the effects of their software in the EULA to be free of any legal charges pressed against them, and reading it would make the user aware of what unwanted effects it may have on the computer, prompting him to abort the installation.

2.2.2. In app advertising

As previously stated, ads can be injected into places they don't belong. But now to talk about where they do belong.

Grayware, in most cases, have an interface, which is the perfect place to advertise. This is quite comparable to how clean software gains revenue, by introducing ads on their app. The technique is harder to abuse, since an interface cluttered with ads will drive the user away, but not impossible.

Advertisements earn the developer money when viewed, but more so when clicked, as when an add is clicked it produces at least twice the amount of money than when only viewed. The price is variable, depending on the API used to show the add, but this is valid in most cases.

What grayware applications do to get more revenue is place ads right next to relevant buttons that the user may need to press. That way, if the user misclicks, the add is opened and the developer gets more money. This technique is more prevalent on mobile devices, where it is easier to misclick, or mistouch in this case.

3. EULA clustering

The concept of EULA has been used throughout this work, especially when talking about grayware. Although it has been talked about in general, this next part will go through the details of what EULA is and what it's purpose is.

3.1. Definition of EULA

The End User License Agreement, EULA for short, is a legal contract between the user and the software developer in which the terms and conditions for using the software are established.

Most of the time when it comes to software, the EULA is presented at the beginning of the installation process. Said installation process will halt if the user is not in agreement with the EULA, and will only continue if he agrees with the terms. Some EULA only allow the user to agree if he scrolled through the entire document, as a measure to be sure it will be read.

The EULA is used by the software developers to hold themselves harmless in the event that the software causes problems to the computer or data. Also the contract can specify that the developer is not liable to legal action if the software is misused by the user, causing problems of any sort.

As the EULA is a legal document, the words used by it are complicated, as most words in legal language are. The reason for that is that the document must be extremely specific about use cases and other issues, otherwise it may be liable to legal action.

Although the document is really specific, the complicated language is not easy to understand for the regular user. In a study, 150 EULAs from popular websites have been given a readability score using the Lexile test. The vast majority of those scores exceeded the college reading level.

Keep in mind that the EULAs were from popular websites. Now take into consideration if a grayware app would want to obfuscate its own terms, the reading score would skyrocket. It would seem that in almost all cases, the EULA is made in favor of the software developer, and not as much in favor of the user, who may not even understand what he reads, if he does read it that is.

Thankfully, because of the introduction of the General Data Protection Regulation, GDPR for short, in 2018, EULA writers are forced to write clearly and plainly by a clause included in the regulation. Since then, most companies have removed a lot of technical terms to make their EULA easier to understand.

The problem with having clearer EULAs is the increase in size. To fully explain what the software does, the developer would need to explain a lot more terms that are currently used and expected of the user to understand. By doing this, the size would grow substantially.

Most of those terms are from the domain of informatics, terms like internet protocol address, IP address for short, and cookies. So the user would need to be somewhat technologically literate to understand said terms without having the EULA become a full sized course in informatics.

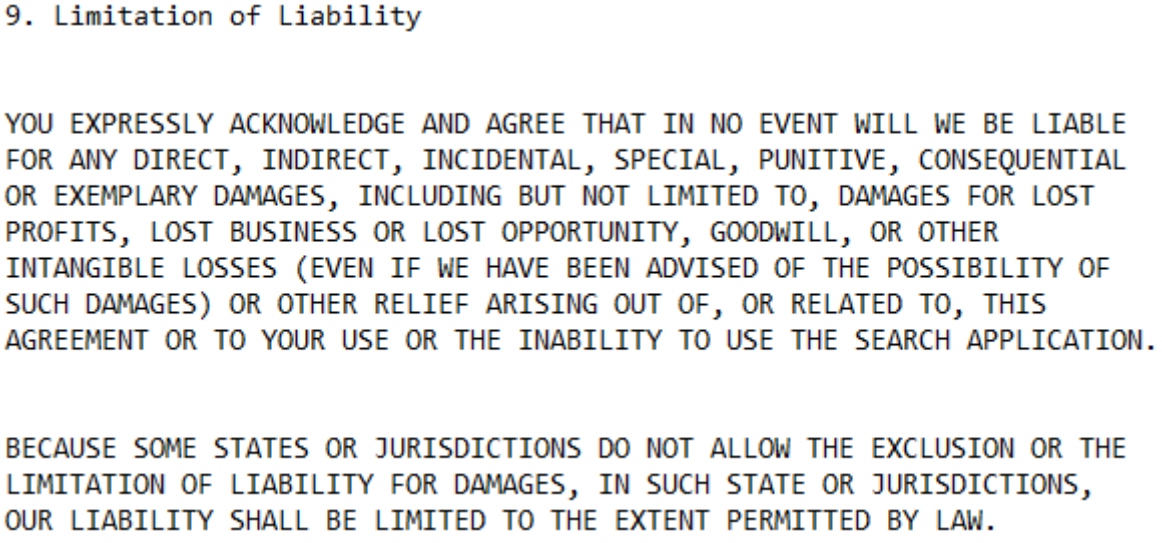
Another factor in the size of the agreement is the ever increasing technology. As time passes there are more and more ways to collect data from the user, and as those become more complex, the more the developer would have to explain. From this point of view, the simplification of EULAs in the future is not looking bright.

3.2. Grayware EULA

Having established that EULA used to be heavily balanced toward the developer's side, when it comes to legal action that is, and currently it is trying to improve but having problems in doing so, it is time to talk about how grayware uses EULA to its advantage, maybe even more so than regular companies.

Grayware is following the line of malware in trying to make money off the user illicitly, but threading the clean path of telling the user everything it does, in an attempt to gain trust. Trust is a powerful word to use in this context, so it is best to call it permission, and the best way to be granted permission, is none other than the EULA, which is a legal contract accepted by the user.

Since there used to be very few laws to actually address the protection of user data in the past, the EULA was used as a cover for anything that would happen to a user's computer or data after the usage of a software. In other words, grayware developers do not use the EULA as intended, to inform the user of what to expect when using their software and to make sure they agree to it. Instead, it is used to hide behind when, or most importantly if, the user realizes what the program does.


9. Limitation of Liability

YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT IN NO EVENT WILL WE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, PUNITIVE, CONSEQUENTIAL OR EXEMPLARY DAMAGES, INCLUDING BUT NOT LIMITED TO, DAMAGES FOR LOST PROFITS, LOST BUSINESS OR LOST OPPORTUNITY, GOODWILL, OR OTHER INTANGIBLE LOSSES (EVEN IF WE HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES) OR OTHER RELIEF ARISING OUT OF, OR RELATED TO, THIS AGREEMENT OR TO YOUR USE OR THE INABILITY TO USE THE SEARCH APPLICATION.

BECAUSE SOME STATES OR JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR THE LIMITATION OF LIABILITY FOR DAMAGES, IN SUCH STATE OR JURISDICTIONS, OUR LIABILITY SHALL BE LIMITED TO THE EXTENT PERMITTED BY LAW.

Figure 1 Limitation of Liability from Mysearch EULA

Most of these problems can be avoided by reading and comprehending the EULA for each software. But then why is it a problem, and such a prevalent one at that? Think about how many programs you have installed on your personal computer. At least 10, and that's being generous, as most users have way more programs installed. Now taking into account that the average EULA has around 2500 words [14], and the average reading speed in academic literature is 250 words a minute, it would take about 10 minutes for each new install, totaling at around one hour and 40 minutes for 10 pieces of software.

Now, 10 minutes for an install does not sound so bad, but it's not only software that uses the EULA. Every site that works with data, and that is about all of them, require the user to agree to its terms. Since there are no good estimates on how many new websites a user visits each year, a study done by Lorrie Faith Cranor and Aleecia McDonald estimate around 1400 a year. 1400 EULAs to read, each needing 10 minutes, would amount up to approximately 230 hours, almost 30 days having 8 hours each a year to read.

Thinking of that number, we start to understand why the regular user tends to skip reading an EULA, but it's not just that. In the ever increasing number of EULAs, the same phrases are found over and over again. After the first few reads, the process becomes boring and annoying, considering the amount of ground that needs covering when it comes to how many EULAs are to read.

And that's not the end of the problem. The numbers from before can be applied to regular EULAs from popular websites and software, but when it comes to grayware, things get quite worse.

Grayware does not want the EULA to be easily read and understood. The more complicated, longer, and looser terms the better. That way, it will turn the reader away when trying to read it, accomplishing the goal of the user not reading it and agreeing with the imposed terms.

The user may get past being turned away by the length, but it's not over. The greater barrier to overcome is the difficulty of understanding the grayware EULA, which may be obfuscated. The sense of the word obfuscation is not that the user needs to decrypt the EULA in a regular sense, but to decode it's meaning, which will be hidden behind vague terms and legal speech. This is accentuated if the user is not good with technology or has no background in computers, which would make it a bit easier.

More recently, with the GDPR in place, these obfuscations may not happen anymore, as it would be illegal to do such a thing. The regulation dictates that "The controller shall take appropriate measures to provide any information... and any communication... relating to processing to the data subject in a concise, transparent, intelligible and easily accessible form, using clear and plain language, in particular for any information addressed specifically to a child" [16].

To simplify, the article interdicts the use of complicated phrases to mislead the user. They have to be clear as day, otherwise they are liable to being sued. Therefore, the future of EULA simplification is looking bright, but only if the user starts reading it, or the effort was all for nothing.

In theory it is easy to understand why the EULA is so important. It is even better understood with practical uses and examples. The following pages are going to be used to describe a real life examples of when EULA is used to the advantage of the developer, who is trying to gain as much money as possible and dodge any lawsuit coming his way [13].

The company to be analyzed is Gator Corporation, and its main product, Gator, and the more silent OfferCompanion. Their main way of making money was through targeted ads, but in a more unconventional way.

Gator Corporation is responsible for creating behavioral marketing, giving its users ads based on certain algorithms, only known by them at the time. Gator claims that their way of showing ads gain way more clicks than regular ads, with a clickthrough rate of more than 10%, more than 35 times higher than average websites.

The company says it is because of the well selected ads on their part, "deliver[ing] special offers precisely related to the users' interest at the time they are making purchasing decisions or shopping for relevant goods and services."

Having the numbers to back them up, Gator started charging a lot of money to advertise on their platform, beginning at $25000 for a promotion of an unspecified timeframe. Although articles do not specifically say it, it is implied that the money needed to place ads on a website is a number lower than the rates Gator sets.

Right now, the software seems it is doing quite well for itself, and it surely does, but at the cost of the user. Gator has a lot of issues that not only cause problems for the user, but for the websites the user visits as well.

Regular users do not have the necessary funds to take the company to court, but other companies that own websites, whose permission was not given to Gator to advertise over, have said capital. And so, Gator has been sued by The Washington Post, The New York Times, Dow

Jones and seven other publishers, which allege the company's ads violate their copyrights and steal revenue.

The companies suing Gator Corporation are displeased that the ads on their page are getting covered by Gator's own ads, almost immediately after loading. Besides hiding the actual ads from the user, it stands in his way to click them, which would get the website owner advertisement revenue.

Another problem the companies had with Gator is that the ads of competitors would get on their website. The website should have control over what ads are displayed on their own website, but that is up to the judge to decide.

The following images come from the declaration of Benjamin G. Edelman, a specialist who analyzed the behavior of the products released by Gator Corporation. He left the conclusion to the judges, but pointed out the so called features of the programs, that make it grayware.

8. Software written and distributed by Gator Corp. causes the display of pop-up advertisements upon a user's attempt to view certain websites. Gator causes the display of these pop-up advertisements without the permission of or payment to such websites.

9. Gator Corp. distributes a free software application called "Gator." Gator is essentially a "digital wallet." Gator provides users with a mechanism for storing personal information about themselves, passwords, user identification numbers and names, and other data that consumers routinely need to input on electronic forms when they shop for goods or services on the Internet. Gator assists users in filling out such forms without having to retype the previously stored information.

10. Gator Corp. bundles a software program that it calls "OfferCompanion" together with its Gator digital wallet software program, so that persons who download the Gator application onto their personal computers automatically have OfferCompanion downloaded and installed onto their personal computers. In addition, when a person downloads certain popular free software programs, such as KaZaa or AudioGalaxy, OfferCompanion is automatically downloaded and installed onto her personal computer. Because OfferCompanion is bundled with other software programs and downloaded automatically with those other software programs, even sophisticated computer users frequently do not know that OfferCompanion has been installed on their personal computers.

Figure 2 Part 1 of Edelman's declaration [13]

11. Once OfferCompanion is installed on a personal computer, OfferCompanion automatically launches and operates whenever a user invokes her computer's web browser. OfferCompanion communicates frequently with Gator Corp.'s computer servers, monitoring the user's activities on the World Wide Web and transmitting that information over the Internet to Gator Corp.

12. When a computer user visits certain websites Gator Corp.'s remote computer systems transmit to the user's computer one or more advertisements to be displayed directly over the content that the owner of the website intended to be displayed.

13. Gator Corp.'s pop-up advertisements typically appear at approximately the same time that the web page that the user has requested is downloading onto the user's computer and opening on the user's computer screen. As a result of Gator Corp.'s pop-up advertisements, users ordinarily do not see the requested web page in the manner that the website owner intended to display it. Instead, users see the Gator Corp. pop-up advertisement concealing some of the content the website owner intended to be displayed on the requested web page. In order for a user to see the requested web page displayed as intended by the website owner, the user must move her mouse to each pop-up advertisement and click the mouse to close each advertisement, thus delaying access to the site's content.

14. The design of the Gator Corp. software allows Gator to cause advertisements to be displayed on any website desired. The Gator software is capable of placing advertisements even on the websites of Gator's competitors, and it is equally capable of placing advertisements on websites that do not sell advertising or that refuse to permit certain types of advertising.

15. Gator Corp. does not prominently advise users who have downloaded OfferCompanion software that Gator Corp.'s pop-up advertisements will change the display of content on particular websites. Furthermore, the Gator Corp. pop-up advertisements fail to suggest that they are not authorized and supplied by the underlying website.

16. Because Gator Corp.'s advertisements appear on a user's screen simultaneously, or nearly simultaneously, with the downloading and opening of the requested web page of the targeted website, the Gator Corp. pop-up advertisements appear to be an integral and fully authorized part of the original underlying web page.

17. Users may find it difficult or confusing to remove Gator software from their computers. Even if a user invokes the ordinary Windows "uninstall" feature to remove the software program that provided OfferCompanion as a part of a bundle, OfferCompanion remains on the user's computer and continues to function in support of Gator Corp.'s pop-up advertising system.

Figure 3 Part 2 of Edelman's declaration [13]

The terms of the lawsuit are confidential, but we do know that the court ordered Gator to stop displaying ads over the plaintiffs' websites, so that may give some insight into what the judge and jury thought about the case.

After the lawsuit settled, Gator started putting effort into looking like a clean application, supporting anti-spyware laws and hiring a former member of the Federal Trade Commission, FTC for short, as chief privacy officer.

Another change Gator made was to rework its EULA. Remembering that the average EULA is around 2500 words, Gator's EULA has been reworked to be around 5900 words, so twice and a bit more than the average.

In the reworked EULA, Gator tries to cover certain loose ends that were factors in its lawsuit previously. Not only that, but try to make itself harder to remove through legal means.

The first notable thing to encounter in the EULA after about 3000 words in is the following line: "You agree that you will not use, or encourage others to use, any unauthorized means for the removal of the GAIN AdServer, or any GAIN-Supported Software from a computer." It is worth noting that GAIN is the name of the network Gator created, and it is mostly synonymous to it.

To understand the importance of that line means to first understand how the uninstallation process for OfferCompanion works, since that is the program that caters all the ads. The software does not have an entry in the Windows add or remove programs tab, instead it uninstalls the moment there are no more partnered apps installed on the computer.

What the software actually does when being removed is try to reinstall itself through the partnered apps. So, when the user tries to remove the OfferCompanion files, the moment a partner app would run, the files would be downloaded again and reinstalled on the user's computer.

Another part of this problem is that the user may not even know which apps are partnered with OfferCompanion. The partnership may have been disclosed in the EULA, but since the user has most likely skipped through it, he probably doesn't know which apps to first uninstall to then delete the adware.

Having that whole process of uninstallation in mind, try reading the quoted line again and understanding that it is not put in the EULA for the user's benefit, but to make the app harder to uninstall, even harder than it currently is.

Another interesting line found later in the EULA states that "Any use of a packet sniffer or other device to intercept or access communications between GP and the GAIN AdServer is strictly prohibited."

The analysis in the declaration made by Ben Edelman was made using a packet sniffer, helping him see what the software would send and receive to and from the network. It would not have been possible without it, or at least extremely time consuming through reverse engineering.

Once again, it is obvious that this line was introduced in the EULA with the sole purpose of impeding analysis.

3.3. The problem

Finally, the most interesting change, the name. After all of the reworks to its EULA and public image, Gator decided to switch to the name Claria. The reason given by the company was to "better communicate the expanding breadth of offerings that [they] provide to consumers and advertisers". Of course, the big reason behind the public speakers is a bit more enticing.

By this point, the name Gator made news headlines, and a lot of people knew about the things they had done, and that they had been sued. Users started pressing decline the moment they saw the name Gator, and that did not do them well for installs.

As their numbers started plummeting they taught up a solution to the problem: a name change. Since the users usually don't take the time to do research, they will most likely be fooled for a click, and since the product is as hard to uninstall as it is, that is mostly all that they need. And because of that, Claria was born.

The name change did not bring any more changes with it, everything staying the same. The most important part that remained, the EULA, still had the same badly intended terms as before, expressed in the same words.

This practice is used a lot in grayware. A software developer with a bad reputation, or whose files are detected by an antivirus solution may opt to change the name of the product, or create a new product as a clone of a previous one with a changed name. That will most of the time clear up it's bad name to the users who do not research or read the EULA, to see for themselves the negative effects of the software itself.

Not only will the name change, but if the file that was detected is modified, depending on the type of detection, the newly created grayware product could remain undetected until it is discovered, and that may take some time.

Currently, if that were to happen, there would be no way to automatically identify the previous product it came from. The process would require human input, searching through a company's timeline and legal documents, if any are available to the public, behavioral analysis and maybe even reverse engineering.

Doing all of those things would mean not learning from the mistakes of the common user, which is looking at the EULA.

4. EULA clustering

As previously stated, the EULA is one of the only ways to automate the process of identifying companies that modified their name but not their product behavior. Since the terms will still contain all the necessary lines to keep the company out of being sued, the EULA will remain mostly unchanged.

Some companies may try to change the EULA, but if the behavior is not changed, then the relevant part of the EULA will either stay the same, or be changed slightly. That problem can be taken care of programmatically.

The end goal is to have software grouped by original company, and implicitly by behavior. This will not only allow for the easy detection of multiple files at once without the need for in depth analysis, but also help with gathering training or testing data for future algorithms to be implemented in this direction.

The programming language of the project is Python. The main reason behind it is the readability of the language. As the project will be used, other people may need more features from it, and may add at their leisure. Since the code is easy to understand and well modularized, changes do not take a full understanding of the whole project, allowing everyone to add what they need to the final product. Some changes may even be integrated in future updates of the tool.

The most relevant python module used for the project is the Natural Language Toolkit module, NLTK for short. This module is used for the analysis of the EULA, finding it's relevant parts and stripping down sentences to their core words, to further be used in clusterization.

Another important part of the project is the PostgreSQL database system. It was chosen because it runs on all major operating systems and has full support for foreign keys, joins, views, triggers, and stored procedures.

4.1. Extracting the EULA

When analyzing a file to further group, the first step is identifying the EULA, if the file has one. To check this an internal unpacking tool is utilized, which parses the executable file and maps its headers and resources while saving them as separate files.

Going through the saved resources, we are looking for a file that contains the EULA, or a link that points us to said agreement. The files that are dropped by the unpacker could be of a few types. Some dropped resources could be other executable files, that can be identified by a valid MZPE header, while others could be other types of binary files, like archives that are further unpacked, picture files like .jpg or .png or even sound files like .mp3.

The resource we are looking for that may contain the EULA fully is a text file. A file that does not contain any other characters than readable ones. In the next image there is an example of unreadable characters and how they look in a hex file viewer.

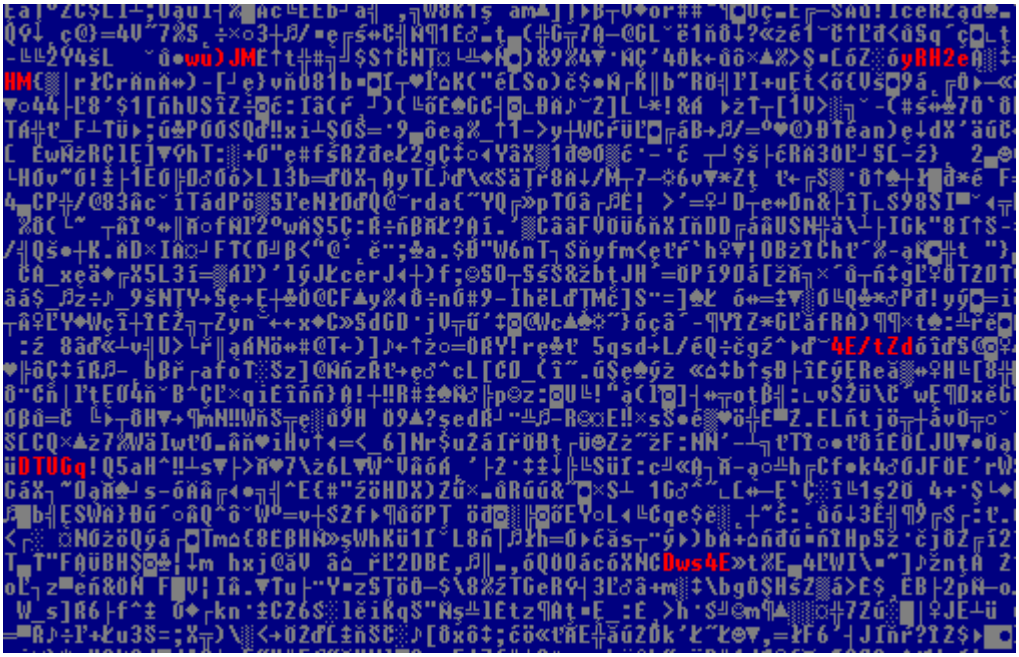


Figure 4Non-executable binary file

To differentiate a regular text file from one that contains an EULA, it must contain at least 3 of the following words: policy, privacy, EULA, terms, legal, agreement. A regular EULA may contain all the words, but some don't, so in order to not miss any agreement, a minimum of 3 out of 6 should be in order.

The more complicated part comes when there is no EULA in the file. Normally we could assume the file has none and stop the process, but that may leave some cases uncovered.

Most of the time the executable file has links in its resources to the software's webpage and EULA, so that when the program is loaded and the user clicks the button to see the agreement on the official website, he is redirected there.

As the links could be found in any resource, be it an executable, regular binary or text file, we look for them in every file the unpacker dropped from the original file and in the original file itself.

Most executables have other links inside of them, especially ones that have a security certificate. To be sure that the process is as optimized as possible, a whitelist of domains has been created so that whenever we find a new link, we can search for it in the whitelist and discard it if it is there. The whitelist will only contain domains that are safe and could never intentionally host files that we are interested in.

The whitelist system can also work as an exception mechanism, in case there could be certain software or developer that has been previously analyzed and has a certain good reputation.

After filtering all of the URLs found, there is another selection in place to find the specific URLs that host EULAs. The check is quite simple: if the URL contains one of the words in the list described a few paragraphs ago, it passes, otherwise it is discarded.

The content of the few remaining links is downloaded and checked with the same process as the other files up until now, if 3 or more words from the list are present, it is accepted as a valid EULA and sent further to be processed.

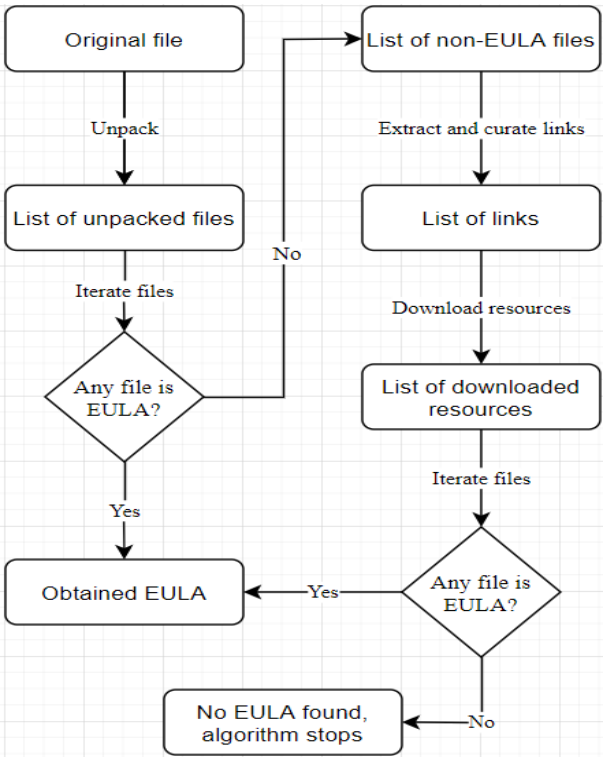


Figure 5Process of extracting the EULA

4.2. Tagging the EULA

If this step has been reached, it means that an EULA in some form has been obtained. To get relevant information from the agreement, it requires some processing.

The first step is getting the text into a better form for processing. In case the EULA is found in a clear text file, this step it skipped, since it is in its normal form. The problem comes when the agreement is found in an HTML file, which has parts that have to be removed before analyzing the syntax.

In the case that the file is HTML, then the removal of its specific traits is in place. To actually remove the tags and paragraphs is a more complicated process, so the easiest way is to extract the EULA text from the file.

Thankfully, there is a python module that does solve that exact problem, called html2text. As the name implies, it extracts the text from an HTML file. It is easily usable and configurable, making the extraction a breeze.

Now that the EULA is in a more appropriate format, it is time to split it into sentences and even further into words. The process is called tokenization, and it attributes every word in the text a certain part of speech.

To further clean the EULA, the stop words are removed. The stop words are the most common words from a language. There is no set list of stop words for any language, but the NLTK provides us with such a curated list, so that will be used.

Most of the words in the list are pronouns, adverbs and conjunctions. The reason for that is to keep the important parts of the sentence so valid combinations can be collected, and not be affected by unspecific words.

The next step of the text preparation is to replace all the nouns with synonyms. Since we care about the semantics of the sentence, and not the syntactic, the nouns are replaced with a set synonym, so every apparition of a concept will be tagged the same, despite the multitude of words used to express it. The synonym will always be singular, to have a more general and correct tag.

Lastly, all of the verbs are brought to the same tense, the present tense. This is also a measure to remove as much clout as possible, making the tagged words easier to group and process.

Now that the EULA is in a tokenized state, the problem becomes apparent. There are still too many common tokens present. If the clusterization were to take them into account, it would be useless for our purpose. The task now is to identify which tokens are more relevant to our clusterization, keeping in mind it's the behavior of the software that should be the main focus point.

To see which part of the EULA are more relevant, we need to look at other software's agreements. A few agreements from random applications sufficed, software that has been tested and classified as neither malware or grayware. It is important that the files are clean, so that the extracted tokens are relevant for clean files.

The clean EULAs are then tokenized and applied the same treatment as the other files, to remove any unwanted tokens. Currently we have a list of sentences which are composed of tokens for every EULA, including the one that should be processed.

We can consider two sentences being the same if 85% of the tokens are the same. In this comparison we ignore order, because it is a small chance that two sentences will ever be the same if we take that into account.

Now having defined the equality between two sentences, we remove any sentence from the EULA that should be currently processed that also appears in the clean EULAs, that way removing a lot of sentences that are generic to any agreement, and keeping the specific ones to further help in the clusterization.

To further increase the precision, something similar was done with grayware files. A few random detected files were collected, had their EULA extracted and then tokenized. This time there was no elimination process, but a more in depth analysis of the sentences.

In the detected files, we needed to search for specific phrases that refer to the software's behavior. Besides that, if we find the exact reason for the file's malicious behavior, those keywords could be extra useful.

After the analysis, a list of about 50 elements was created with the most used combination in the detected files, some of them including: [[relevant], include, software, thirdparty], [[ad], advertiser, thirdparty], [[offer, relevant], software, use].

The combinations above are represented as lists of strings. It can be observed that a group contains more than just strings, but also another list. The list inside the group represents the keywords selected from phrases that are specific to detected files and their behavior.

The final part before the actual clustering is the tagging of the EULA being processed. Having at our disposal the extracted list of the most used combinations, we mark down every apparition of any of the combinations described before, as well as count how many appearances they make.

4.3. Grouping the files

Now that the tagging is complete, and we have the data to all of the files, it is time to group them. The process only involves moving the file around in directories, but the more difficult part is in which directory to place said file.

The file is currently tagged with the apparition of relevant keyword combinations found in its EULA. The tags are saved in a file named with the same name as the file, which is usually the file's md5 hash, and adding the extension eula.

The clusterization is then applied to a list of files that have a .eula file present along with them in the directory given as the first parameter to the script. The algorithm groups the files depending on the second parameter given at runtime, which can have the following values.

The first value can be –most_files, which will try to prioritize placing files in tags that have a lot of files. The use of this parameter will most often lead to files being separated under a tag composed of only one group of tokens, meaning a specific sentence. This way of grouping is useful for finding files relevant to the specific sentence.

The other value is –most_tags, which prioritizes the clusters defined by the most tags possible. When using this parameter, the files will be more spread than when using the previous option. This prioritization leads to the most precise behavioral clusterization, since if files have the same used sentences. While using this option, the chance of two files having the same EULA, and implicitly the same company, is very high.

The last value of the parameter is the –multiple_files parameter, which indicates to the algorithm to place the file in all the possible clusters, as before the file would only be placed in the first category found for it, depending on the preference.

4.4. Results

The clusterization has been run on a set of 200 files, 100 clean and 100 detected, the conditions being that the files are software installers that have an EULA associated with them, and said EULA contains at least one tagged sentence. The clean files have not been picked at random this time, but have been chosen under the criteria that they have a certain feature that has to do with ads, or has a behavior that is slightly suspicious.

As expected, when running the clusterization with the first parameter, only a few groups are created, with multiple files attached to them. 12 groups have been created, the biggest group having 47 files and the smallest having 8, but all of the groups were defined by a single tag.

When running the clusterization with the second parameter, the results are way more specific. 44 groups are created this time, the biggest having 15, and the smallest having just one. The good part about this, is that the group defined by the most tags was standing at 4 phrases, and the least were just one. Although at first it may seem pointless to have a lot of small clusters, and most of the ones with lots of tags have just one file, this type of clusterization shines when it finds 2 or more files with more than 3 common tags, those files being almost identical in behavior.

The final value of the parameter revealed a bit more information than expected. By creating every possible present tag combination, 335 clusters were created. The most number of tags per group was once again 4 tags, and of course, since the files can be put in multiple groups this time, the least tags was 1. Now that the groups of 4 tags did not consume the file, there were more 3 tag groups with more than 1 file in them. The largest 3 tag group was a group of 4 files.

Out of the 4 files in the biggest 3 tag group, 3 of them shared the same company, while the fourth did not. Behaviorally, all of them are the same, and the EULA for the software is extremely similar.

Conclusion & Future work

This paper was created to present a way of grouping files by the behavior described in the end user license agreement extracted through natural language processing.

The clusterization works wonders in reducing the time required to analyze multiple files. Also, it has the added benefit of giving a hint into what companies may have changed their name in order to avoid bad publicity.

In the future, work will go into improving the natural language processing, trying to identify similar clusters and see what changes could be made, if any should be.

Increasing the starting set of clean EULAs may be beneficial, since the more common phrases are eliminated, the more relevant data remains. The problem is that these agreements should be curated manually, since if one of them would contain relevant keywords, the clusterization would lose precision.

Also, more research could be made on confirming or infirming the results when it comes to companies. Most of the time companies are not so popular as the presented case, and all their data isn't public, so the moment more data becomes public, the result will be verified.

Bibliography

[1] https://www.rapid7.com/fundamentals/malware-attacks/

[2] https://www.malwarefox.com/malware-types/

[3] https://www.malwarebytes.com/malware/

[4] https://www.coursehero.com/file/phpusa/Before-the-term-malware-was-coined-by-Yisrael-Radai-in-1990-malicious-software/

[5] https://encyclopedia.kaspersky.com/knowledge/history-of-malicious-programs/

[6] https://www.bitdegree.org/tutorials/what-is-cryptocurrency

[7] https://www.buybitcoinworldwide.com/mining

[8] https://www.investopedia.com/terms/b/bitcoin-mining.asp

[9] https://www.opswat.com/blog/most-destructive-malware-all-time

[10] https://www.csoonline.com/article/3212260/the-5-biggest-ransomware-attacks-of-the-last-5-years.html

[11] https://www.hindawi.com/journals/jcnc/2011/569829

[12] https://arstechnica.com/information-technology/2008/04/detecting-and-fighting-greyware

[13] https://www.benedelman.org/spyware

[14] https://www.theatlantic.com/technology/archive/2012/03/reading-the-privacy-policies-you-encounter-in-a-year-would-take-76-work-days/253851/

[15] https://www.nytimes.com/interactive/2019/06/12/opinion/facebook-google-privacy-policies.html

[16] https://www.privacy-regulation.eu/en/article-12-transparent-information-communication-and-modalities-for-the-exercise-of-the-rights-of-the-data-subject-GDPR.htm