Dataset Construction and Analysis of Screenshot Malware

Hugo Sbai
Department of Computer Science
University of Oxford
Oxford, UK
hugo.sbai@cs.ox.ac.uk

Jassim Happa
Information Security Group
Royal Holloway,
University of London
London, UK
jassim.happa@rhul.ac.uk

Michael Goldsmith
Department of Computer Science
University of Oxford
Oxford, UK
michael.goldsmith@cs.ox.ac.uk

Samy Meftali
Université de Lille 1
Lille, France
samy.meftali@univ-lille1.fr

Abstract*—Among the various types of spyware, screenloggers are distinguished by their ability to capture screenshots. This gives them considerable nuisance capacity, giving rise to theft of sensitive data or, failing that, to serious invasions of the privacy of users. Several examples of attacks relying on this screen capture feature have been documented in recent years. However, there is not sufficient empirical and experimental evidence on this topic. Indeed, to the best of our knowledge, there is no dataset dedicated to screenshot-taking malware until today. The lack of datasets or common testbed platforms makes it difficult to analyse and study their behaviour in order to develop effective countermeasures. The screenshot feature is often a smart feature that does not activate automatically once the malware has infected the machine; the activation mechanisms of this function are often more complex. Consequently, a dataset which is completely dedicated to them would make it possible to better understand the subtleties of triggering screenshots and even to learn to distinguish them from the legitimate applications widely present on devices. The main purpose of this paper is to build such a dataset and analyse the behaviour of screenloggers.

Keywords*—Spyware, screenlogger, malware dataset, behaviour analysis, malware detection, screen capture, remote access trojan.

I. INTRODUCTION

Screenlogging is one of the most dangerous functionalities in today's spyware as it can highly contribute to reach the potential goals of a hacker. Cybercriminals take advantage of the fact that people are using internet more to carry out harmful screen capture attacks [59]. MITRE [3] lists 60 examples of widely known malware programs with the screenshot capability.

By closely examining these examples, we observe that screenlogger users can be divided into two categories: financially motivated actors and state-sponsored actors. The first category targets important industrial companies (e.g. Bronze Butler [4]), online banking users (e.g. RTM [9], FIN7 [10], Svpeng [4]), and even banks themselves (e.g. Carbanak [11], Silence [12]). The second category, which is even more problematic, targets critical infrastructures all over the world. For instance, the malware TinyZbot [13], a variant of Zeus [7], targets critical infrastructures in more than 16 countries. More precisely, the targets can be democratic institutions (e.g. Xagent targeting the DCCC and DNC in the US [18] or Regin [8] which has been taking screenshots for at least six years in the IT network of the EU headquarters), diplomatic agencies (e.g. ScarCruft, a North Korean malware targeting diplomatic agencies [14]) or the US's defense contractors (e.g. IronTiger).

Malware authors are also very inventive when it comes to using screen captures in a malicious way. Indeed, screen captures can be used for a wide range of purposes. Some malware only take one screenshot during their whole execution for reconnaissance purposes (see if the victim is worth infecting e.g. Cannon [15], Zebrocy [16]) or to hide what is happening on the victim's screen by displaying a screenshot of their current desktop (e.g. FinFisher [16][17]). Others take numerous screen captures for a close monitoring of the victim's activity. This can allow to steal sensitive intellectual property data (Bronze Butler [4]), banking credentials (RTM [9], FIN7 [10], Xagent [18]), or to monitor the day to day activity of banking clerks to understand the banks' internal mechanisms (Carbanak [11], Silence [12]).

The screenshot functionality is sometimes the unique functionality used in some phases of an attack in order to make observations while remaining stealthy. It is for example the case of the Carbanak attack targeting banking employees [5] [6]. The attackers used the screengrabs to create a video recording of daily activity on employees' computers, amassing knowledge about internal processes before stealing money by impersonating legitimate local users during the next phase of the attack.

In addition to these dangerous capabilities allowed by screen captures, it is even more difficult to fight this threat, as this functionality is used more and more by legitimate applications. Paradoxically, there are few works in the literature about screenloggers, making this threat relatively unknown. To the best of our knowledge, the only studies focusing on it are limited to very specific questions, which are the Android Debug Bridge's vulnerability allowing screenshots taking on Android smartphones [27, 28, 29, 30] and screenshot protection during authentication using virtual keyboards [31, 32, 33]. Thus, there is no general view of the threat represented by screenshot-taking malware and how this functionality is implemented by attackers. Therefore, the aim of this paper is twofold: building a representative screenlogger dataset and analysing it to identify the most promising criteria for screenlogger detection.

After studying the existing datasets (Section II) and briefly describing our threat and system models (Section III), our methodology will be discussed (Section IV). We will then present the first existing screenshot-taking malware and legitimate applications dataset as well as some insights into their behaviour in the results section (Section V).

II. Related work

Although there are many malware datasets available [34-49] containing diverse categories of malware, there is no existing dataset dedicated to gathering diverse forms of screenshot-taking malware. The only screenlogger samples can be found in general malware datasets in the middle of other types of malware. However, such general datasets would not allow us to test our future detection approach nor to gather meaningful insights into screenloggers' behaviour. There are several reasons for that.

First, general datasets authors do not indicate whether their dataset contains screenshot-taking malware or not. They usually just indicate that the dataset contains "spyware", without giving information about their functionalities and particularly the screenshot functionality. To the best of our knowledge, only one dataset explicitly includes screenlogers' network traffic related to well-known malware programs Zeus and Ares [18]. However, the collected network traffic of the aforementioned screenloggers is not representative of the real screenloggers' behaviour because authors only consider a periodic screen recording of 400 seconds, whereas recent screenloggers are very diverse. Thus, existing malware datasets include very few screenloggers samples and they are limited in nature because they only reflect a small number of characteristics describing a very specific screenlogger behaviour.

Second, the screenshot functionality is often not executed immediately when a machine is infected. In particular, for RATs (Remote Access Trojans), screenshot-taking is triggered by a command sent by the remote attacker. There are also some cases where screenshots are triggered by user events (such as mouse clicks), or by the execution of certain programs or specific web pages (online banking, etc.). Consequently, even if existing datasets may contain screenloggers, the screenshot functionality will most probably not be used anyway, preventing us from making observations.

Third, as mentioned before, legitimate applications are increasingly taking screenshots. These application are very diverse: screen sharing (e.g. Skype [19], Screen Leap [20], Join.me [20]), remote control (e.g. TeamViewer [22], Netviewer [23],

GoToMyPC [24]), screen casting (e.g. Camtasia [25], CamStudio [26], Ezvid [27]), parental or employees control (e.g. Verity [28], Kidlogger [29], Norton Online Family7 |68]), or screenshot taking and editing (e.g. Picpick [30], Snipping Tool [31], FastStone Capture [32]). Thus, in order to effectively analyse spyware using the screenshot capture feature, it is critical to be able to understand and identify the similarities and differences between their behaviour and that of legitimate applications. However, there is currently no dataset of legitimate applications taking screenshots and again, even when some of the legitimate applications contained in the da- taset can take screenshots, it is important to make sure that this functionality is trig- gered at the time of execution (for example by enabling screensharing during a Skype call).

Because of the small amount of works on screenloggers in the literature, existing malware datasets are not suitable for creating an efficient detection method. They do not mention whether screenloggers are included or not, and when they do, only basic and naïve behaviours are covered due to the specificities of screenshot triggering. The same problems apply to legitimate screenshot taking applications.

III. Threat model and system model

A. System model

The targeted systems are desktop environments. The main reason why our work focuses on computer operating systems is that the screenshot functionality is a legitmate functionality offered to any application. In contrast, on smartphones the principle is that apps cannot take screenshots of other apps, and, the only way of doing this is to exploit specific vulnerabilities or to divert some libraries but with many limitations (permission asked to the user at the beginning of each session, icon in the notification bar). Therefore, the architecture designs of mobile systems and computer systems are fundamentally different, which may lead to different solutions.

Another assumption is that the victims are not particularly security-aware: they are not necessarily aware of the existing threats and will not install a specific protection against screenshots, such as a specific viewer to open documents in a secure environment preventing screenshots (by using the setWindowDisplayAffinity function [33]). Moreover, the targeted victims may be any individual or organisation, ranging from typical laptop users to small companies or powerful institutions.

B. Threat model

Our threat model is composed of the victim, the attacker, and a spyware with the screenshot functionality. In this model, a screenshot is defined as an exact reproduction in an image format of what is displayed on the screen. Therefore, we exclude the case of malicious web extension taking screenshots by reconstructing the display from the DOM [34, 35]. Moreover, we do not cover privacy leaks due to applications taking screenshots of themselves for commercial purposes, e.g. session replay [36]. Shoulder surfing is also out of the scope of this work.

The attacker's main goal is to extract sensitive data (personal information, business information, or classified information). They may infect the system using common methods such as Trojans or social engineering, or through a malicious insider. The adversary has no physical access to the victim's device. They have no knowledge about the system and tools installed on it before infection. The attacker does not require any advanced hardware. They have no interaction with the victim's device except his capability of receiving files transmitted by the spyware via the network and optionally the possibility to send commands to the spyware. The adversary may be a simple individual acting alone or a more powerful entity, such as an institutional or a cybercriminal group.

IV. Samples collecting and analysis approach

A. Spyware and legitimate applications samples collecting

In order to effectively analyse spyware using the screen capture feature, it is essential to be able to understand and identify the similarities and differences between their

behaviour and that of legitimate applications. Thus, the first step of our approach aims to constitute a significant sample of legitimate applications and screenlogger type spyware, with varied and representative behaviour.

The names and hashcodes of our samples were collected from MITRE [3], which does not contain malware samples but lists 60 malware with screenshot functionality. The hashcodes can be found in the Indicators Of Compromise (IOCs) section of the security reports. Based on these haschodes, we were able to collect 600 spyware samples from VirusShare [37], AVCaesar [38], Malshare [39] and VirusSign [40]. We also collected some source codes on open archives such as Github.

For the collection of legitimate applications, 94 software using the screenshot function were identified and collected. These applications come from different representative categories of screenshots uses among the most popular. These categories are: screen sharing, remote control, screen casting, parental control or employee control and capture and edit screenshots.

B. Samples execution and reports generation

The execution of spyware and legitimate applications is an essential step to understand their behaviour, in particular the identification of actions that trigger the screen capture function. Given the nature of the studied software, their execution was carried out in a secure environment. Spyware are executed on a Windows 10 virtual machine (Inetsim internet configuration) [43]. After each execution, the machine is reinitialised from a snapshot of the non-infected initial state. Then the behaviour was analysed using dedicated software: Wireshark [41] and API Monitor [42].

C. Reports analysis

In order to make sure that the collected samples were actually taking screenshots, it was necessary to find a sequence of API calls that characterises screenshot-taking. This sequence must be precise enough that non-screenshot cases would be excluded and at the same time not too precise to avoid false negatives. The difficulty lies in the fact that there is no unique screenshot function proposed by Windows APIs but rather a sequence of 5-6 functions that can be used alternatively with other functions and that can individually be used in other contexts. Thus, it is impossible to deduce that a screenshot was taken by only looking at a unique function.

To define the criteria characterising a screen-capture, thirty legitimate screenshot-taking applications were analysed. It shown that different API calls sequences can be used for the purpose of screenshot taking. Two main libraries are used: the Windows Graphics Device API (GDI) and the Desktop Duplication API which replaces mirror drivers. We therefore had to make flowcharts with different alternatives at each step, as illustrated in Figure 1, and transcribe these into a script that takes as an input the API calls reports and returns information such as the number of screen-captures taken and their frequency (if the screenshots are taken at a regular time interval).

For the GDI library, using the flowsharts we identify a screen capture by the call of one function that retrieves the content of the screen (GetDC, GetWindowsDC, CreateDCA or CreateDCW), followed by the call of the Bitblt or the StretchBlt function. The script ensures that BitBlt's hdcSource parameter is the value that was returned by the function capturing the screen's content.

A rarer sequence using GetDIBits, CreateDIBitmap, SetDIBits and CreateBitmap functions was found in some applications (e.g. JoinMe [21] and AnyDesk). When this sequence is used, BitBlt is called with small size parameters values whereas in the above sequence it is called with the size of the screen.

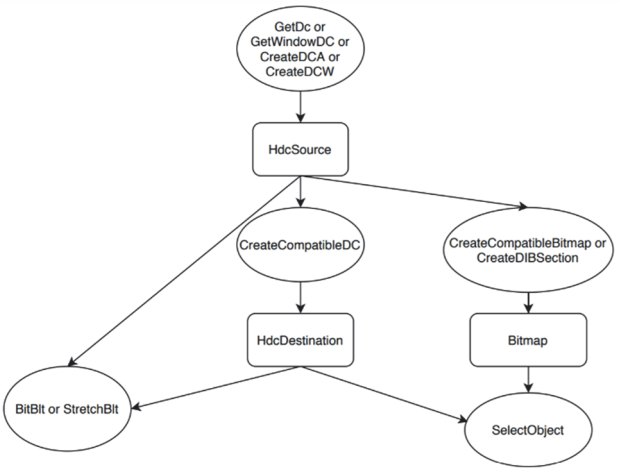

Fig. 1: GDI API calls flowchart

For the Desktop Duplication API, a screenshot is identified by the use of the AcquireNextFrame, GetFrameMoveRects, and GetFrameDirtyRects functions, which allow to capture only the part of the screen that changed compared to the previous frame. This library is mainly used by the applications that capture the screen continuously (screen sharing, remote control, screen casting).

Javaheri et al. [44] proposed a method for screenloggers detection and they also identified criteria based on Windows API calls to characterise a screen capture operation. Their solution was to look at the GetWindowsDC and Bitblt functions. This is similar to the criteria we identified, but it does not take into account the other equivalents functions (StretchBlt, GetDC, …) nor the other libraries that can be used (Desktop Duplication API). Thus, it does not allow to identify all the screen capturing applications. Moreover, the parameters of the function are not verified especially the hdcSource parameter of the Bitblt function which must refer to the screen content. Otherwise, it means that Bitblt was used for a different purpose, and some applications that do not capture the screen can be identified as taking screenshots, which is problematic too.

By running our scripts on the found samples, we realised that no sample but one (over 600 samples) were actually taking screenshots. Different reasons can be put forward to explain this unexpected result:

- Many malware programs infect their targets through different steps, the screenshot module is often not present from the beginning and is rather downloaded from the server afterwards. However, today, almost all the malware servers are dead and even if they aren't, we do not allow the samples to connect to the network for obvious security reasons. For example, the Prikormka malware has a "downloader module", which function is to download other modules which are not present at the time of infection [65]. Silence downloads a dropper which will communicate with the server to obtain the screenshot plugin [12].

- Some malware programs are fitted with anti-sandbox /anti-analysis capabilities. For example, ZeusPanda [45] looks for files showing the presence of Virtual Box, VMware, Wireshark among many others on the infected device. It also looks for these tools in the list of running processes. If it realises that it is within an analysis environment, it does not run.
- Sometimes, the screenshot settings (Should screenshots be taken? When? Targeting what applications? …) are defined by the attacker before infection and impossible to modify afterwards with only the sample at our disposal (particularly KeyBoys [46] and AgentTesla malware).
- In most cases, the screenshot functionality is triggered by a command coming from the attacker. However, analysis reports rarely indicate this command. Moreover, the samples available in current datasets only include the client part of malware.

Thus, as samples that are available on current malware datasets are not taking screenshots during their execution, it was necessary to do a more extensive research to find malware whose screenshot function can be enabled.

D. *Collecting widely used and available malware screenlogging tools*

Often, malware authors do not bother implementing the screenshot functionality themselves and, instead, reuse RATs (Remote Access Trojans) or "Pentest tools" widely available on the internet. The only element that varies between different malware is therefore the infection medium which is not relevant for this paper. Several well-known cybercriminals, having realised significant attacks, use this kind of tools. By way of illustration and in a non-exhaustive manner, one could cite: the FIN7 group (particularly the GRIFFON and Halfbaked malware) using Carbanak for screen capture [10], the Group5 group using njRAT and Nano Core RAT [57] and the CopyKittens group using CobaltStrike and Meterpreter [58].

Thus, by analysing these tools, it seems reasonable to hope covering a majority of screenshot taking malware. As they are widely available, we were able to get working samples (with the client and server parts) and even source codes for the available ones. We constructed Screeminals, a dataset completely specific to screenshot taking malware, containing 118 samples. Most of them were source codes that we had to compile and make working. We ran each of them to ensure that the screen capturing function was triggered. Given the dangerous nature of the studied software, their execution was carried out in a secure environment, with the client and server parts being ran on two different virtual machines.

E. *Ensuring the completeness of the dataset*

A major issue in our work was to ensure the completeness and the representativeness of our screenlogger dataset, a complete and representative dataset being defined as a dataset in which all the behaviours displayed by well-known screenshot taking spyware are included with real proportions. Imposing this constraint is particularly important in the case of screenloggers. Indeed, as we will see in Section V.A, the analysis of the security reports of the existing in-the-wild screenloger shows the existence of various behaviours, while, as we will see it in Section V.B, most screenloggers samples that are available display quite identical and fairly basic behaviour (continuous high-frequency screenshot taking triggered by a command). However, to develop an effective detection approach, it is necessary to take into account all the types of behaviours that exist in the wild. Including these rarer behaviours could help detect screenloggers which remain undetected otherwise. To ensure this completeness criterion, we therefore proceeded as follows: 1/analysis of the malware security reports listed on MITRE [3], and, depending on the results of this analysis, determination of the list of behaviours that must be present in the dataset to ensure its completeness (Section V.A), 2/ analysis of the screenlogger samples we were able to collect (Section V.B) and 3/ implementation of a tool to generate the behaviours of the list that are not present in the collected dataset (Section V.C).

V. RESULTS

A. *Analysis of security reports of screenloggers and characterisation of the completeness criteria*

The first step in analysing the behaviour of screenshot taking malware was to gather high level information from one hundred security reports recovered on MITRE [3]. After that, a set of criteria that can discriminate screenshot taking malware from each other has been identified. These criteria cover all the main operating actions of screenloggers. Some of them are about the way by which the screenshot feature is trigged and executed, others are about storage and compression, and other ones regard network communication and data transmission. These results imply several observations:

1) *Screen capture*

As described in Section IV.C, two main libraries can be used, GDI and Desktop Duplication API. Our analysis showed that existing malware do not seem to use the functionalities offered by Desktop Duplication API.

Regarding the triggering of the screen-capture functionality, the first observation is that, a vast majority of screenloggers (67%) wait for a command from the C2 server to start capturing the screen. This imposes an important constraint on the dataset: we must gather both the server and the client parts of the screenloggers in order to be able to trigger the screenshot functionality ourselves.

Independently of the need of a command to start taking screenshots, different events can be the cause of the screen-capture triggering. These events can be pre-configured in the malware compiled file, or they can be set on-the-fly upon the reception of a command once a victim has been reached. For instance, Remexi is pre-configured to take screenshots when some applications of interest are opened, after a configurable number of mouse clicks, and does not need to receive any command to specify these screenshot-taking parameters [47]. On the other hand, Biscuit [51] and Powersploit [49, 50] take screenshots at a configurable frequency which is set in a command received from the C2 server. Other parameters can be set in the command, such as a time slot.

As shown in Figure 2, the different screenshot triggering events that were identified are: frequency (35%), punctual command (38%), application of interest (9%), mouse clicks (3%) and unique screenshot upon infection (5%).

Regarding frequency, the observed screen-capturing frequencies range from 2 sec (Cross RAT [69]) to 15 min (Prikormka [65]). Malware offering remote desktop functionality have a higher frequency, which may reach 30 frames/sec (Xtreme RAT [66]). The screenshot frequency may vary over time: for example, Prikormka has a normal

frequency of 15 min but it increases to reach 5 sec when VoIP applications such as Skype or Viber are open.

Some other screenloggers allow on-demand screenshots: they have a specific command to take one screenshot at a time (this is the category "ponctual command" on Figure 2). Most malware offering remote desktop also offer an independent screenshot command (e.g. Azorult [67], Carbanak [11], NetWire [68]).

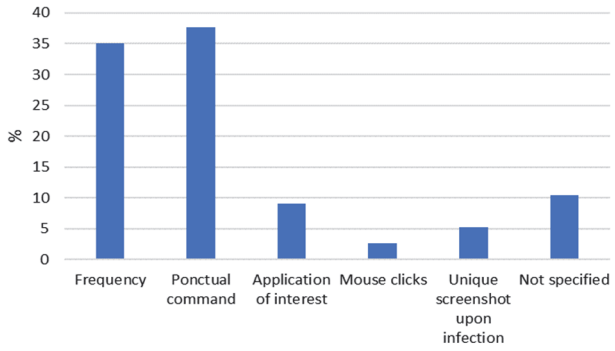

Fig. 2: Screenshots trigerring in the malware security reports

Some few malwares have different behaviours as they capture screenshots in response to mouse clicks (e.g. Remexi [47]) or to the launching of a target application (e. g. Catchamas [48]). This last category of screenlogger waits for a specific application to be open by looking windows titles (Remexi [47], RTM [9], T9000 [52]) or the list of running processes (Biscuit [51], Flame for instant messengers [60]). Finally, some malwares only take one screenshot during their whole execution for reconnaissance purposes (see if the victim is worth infecting e.g. Cannon [15]).

2) Screenshots format and storage

This criterion may determine to a certain extent the possible congestion of the file storage system, which can be an indication for detection. Moreover, it has a considerable influence on the use of network resources.

Regarding the screenshots size, even if this information is often not available (no information for 55% of the security reports), it follows from our study that more than 37% of screenloggers capture the entire screen without targeting a particular area or window. Nevertheless, three other operating modes are represented, even if it is in small proportions. These are the capture of a target window (T9000 [52]), of all overlapping windows (Invisimole [53]) or of a delimited area of the screen (Zeus Panda [45]). Interestingly, the Remexi malware [47] proposes a parameter for full screen or only active window screenshots.

Like the image size, the chosen format for images representation has a direct impact on memory / disk and network usage. 43% of screenloggers for which information is available use the JPG format and 32% the PNG format. Some other malwares opt for other formats such as BMP (12%), and to a lesser extent AVI, WCRT and RAR.

The information of whether malware use memory representations or disk persistent storage of image files is unfortunately not available for almost half of considered screenloggers. However, hard drives seem to be the most used storage mean (90% of malware for which the information is available).

3) Exfiltration of captured data

All the malwares of this study transmit the captured screenshots to remote and malicious servers. 40% of screenloggers for which the protocol is identified use HTTP, which increases their chances of going unnoticed in the large HTTP flow passing through almost all machines. HTTPS, FTP, SMTP and even SOAP complete the list of used protocols. We can also notice that a non-negligible proportion of malware (20%) does not have an application layer network protocol, and simply use the transport layer by sending TCP packets.

Regarding the event triggering screenshot sending, four possible behaviours have been identified. The first one, which is also the most common, is immediate sending of the captured image directly after the screen capture (e.g. Magic Hound [61], RTM [9]). The explanation could be that many of these malware programs use the screenshots for a real-time purpose such as remote control on the victim's machine or real time observation of the victim's activity. The three other behaviours are the screenshots sending at a regular frequency (e.g. Micropsia [54], Flame [60], Rover [62]), when a specific command is received from the C2 server (e.g. Biscuit [51], Powruner [55]), or each time a pre-defined number of screenshots is taken (e.g. RTM after 6 screenshots [9], Pteranodon after a configurable number of screenshots [56]). This criterion is important because it determines the network traffic characteristics (size of the packets, frequency of sending, etc.).

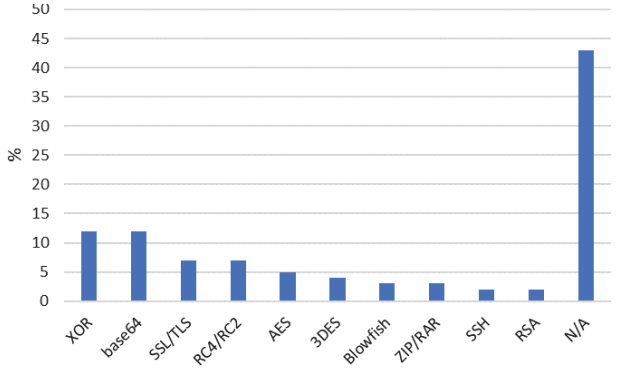

Fig. 3: Image files encryption in the malware security reports

For the malware for which this information was available, we observed that the encryption techniques are very diverse, each malware designing its own encryption method often by combing several techniques. As illustrated in Figure 3, the encryption methods are more or less sophisticated, some malware only using XOR operations (e.g. T9000 [52]) or base64 encoding (e.g. Powruner [55]), and other ones combining several encryption algorithms such as Chopstick. Bronze Buttler uses two alternative encryption techniques: RC4+base 64 and AES [4]. Knowing the encryption methods can be important to detect the exfiltration of image type files, which will be more easily discovered if there is no or very simple encoding (base 64, zip, rar).

4) Completeness requirements for a screenlogger dataset

Based on these results, we were able to define criteria ensuring completeness of the dataset. The different behaviours our dataset must include are summed up in Table 1.

Table 1. Completeness requirements

Windows lib. used to take screenshots	GDI (with all the alternative functions) / Desktop duplication API
Screenshot triggering	Frequency / Punctual command / Application of interest / Mouse clicks / Keyboard presses / Unique screenshot upon infection
Captured aera	Whole screen / Window of interest / All overlapping windows / Around the mouse pointer / Configurable area
Storage	HDD / Memory
Image file format	JPG / PNG/ BMP
Encryption	Yes / No
Communication protocol	TCP / HTTP(S) / FTP(S) / SMTP(S)

B. *Analysis of the malware dataset (collected in Section IV.D)*

1) Screen capture

The malwares of our dataset exclusively use the GDI library to capture screenshots. Therefore, this does not include the Desktop Duplication API. Moreover, all malware composing the dataset need to receive a command to start taking screenshots. The opposite behaviour (start taking screenshots automatically) is not represented.

Regarding the event triggering the screen capturing functionality, the two most current behaviours are the capture of the screen at a given frequency (38%) or on-demand upon reception of a specific command allowing to take one screenshot (59%). Several screenloggers, such as Xtreme RAT, Spynet and NetWire, offer the two possibilities. Remcos constitutes a particular case because its screenshots are triggered by the occurrences of some target keywords in the titles of the opened windows. Two behaviours that were observed in the previous section are not represented here: screenshots triggered by mouse clicks and unique screen capture during the whole execution for reconnaissance purpose.

Regarding the values of the screen-capturing frequencies, they range from 17 ms to 60 s, and several screenloggers have a configurable frequency (e.g. Powershell-RAT [49], Powersploit [50]). Interestingly, thanks to the source codes we were able to observe that some malware use a random number between each screen capture and therefore don't take the screenshots at exactly equals intervals of time. For instance, Carbanak draws a random number between 15000 ms and 30000 ms at each capture, and CtOSRAT draws a random number between 17 ms and 31 ms.

2) Screenshots format and storage

In the constituted dataset, 57% of malware use JPG coding to represent images. The remaining ones use BMP and PNG formats in equal proportions. These observations appear to correspond to the theoretical ones of section B.

Regarding the screenshot size, a majority of the selected malware (86%) captures the entire screen at each shot. However, 14% of malware have smarter and more optimized behaviours as they capture either a specified zone using its coordinates (7% e.g. Spynet, Xtreme RAT) or only difference (changes) between 2 successive screens (Gh0st). Moreover, we found a behaviour that we did not observe in Section B, which is the capture of the zone around the mouse click, which is an option proposed by Pupy. This enables sending smaller packets, and this can be useful in attacks targeting online banking users which enter their password using a virtual keyboard, as proposed by many banks [63].

In our dataset 61% of malware only use a memory representation of the captured image, while 39% have persistent storage strategies on disks. On this criterion, it is quite clear that the results are the exact opposite of those concluded in section B, where most of the malware were using disk storage.

3) Exfiltration of captured data

A very large proportion of the studied malware does not have an application network layer and therefore 94% of them use only the transport layer to exfiltrate data in the form of TCP packets. 6% of our samples use the HTTP protocol. Regarding the screenshots sending triggering, all malwares of the dataset send captured images to a remote server immediately after capturing the screen. This is unfortunately not sufficiently representative of the potential behaviours presented in section B.

Regarding encryption, the major part of the analysed malwares do not encrypt the sent data. Only 25% of them do it, all using different encryption algorithms.

This dataset is, to the best of our knowledge, the first one dedicated to screenshot-taking malware and it is available on Github [64]. Even if 118 is a significant size, it is not sufficient to represent all the possible behaviours of such malware. Indeed, the analysis results shown that some behaviours and characteristics are missing whereas they were observed in the security reports analysed in the previous part.

C. *Implementation of a "screenlogger generator"*

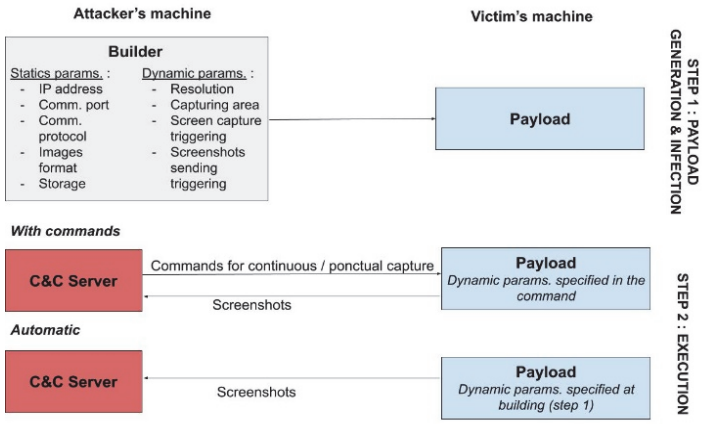

Fig. 4: Screenlogger generator (abstract architecture)

To make our dataset complete, our solution consists in developing a configurable tool that enriches the constructed dataset with all the missing behaviours mentioned in the analysed security reports but not found while analysing our dataset. Although not ideal, this solution will allow to have a complete toolset for studying the screenshot-taking behaviours. It takes the form of a "screenlogger generator" which generates a payload according to the specified parameters. Then, this payload is run on the victim's machine

whereas the server part is waiting for connections on the attacker's machine (Figure 4). Measures were taken to prevent this tool from being used for malicious purposes.

If the option "without command" is selected, all the parameters, including the dynamic ones, will be set at the building time and the payload will start taking screenshots according to these parameters directly upon infection of a new victim.

On the contrary, if the option "with command" is selected, the dynamic parameters cannot be specified at the building time and the screenlogger will wait for a command from the C2 server to start capturing the screen (the dynamic parameters are indicated in the command). This command may be used to take isolated or continuous screenshots.

The generated screenloggers which make the dataset complete can be found along with the tools mentioned in part IV.D on the following github repository: [64]. Again, for security reasons, the executable is not provided.

D. Analysis of legitimate screenshot-taking applications

We have assembled a dataset of 94 applications taking screenshots, among the most commonly used. These applications are analysed with the same criteria and metrics as malware.

1) Screen capture

Legitimate applications use the two main available libraries to take screenshots: GDI (76%) and Desktop Duplication API (17%). The Desktop Duplication API is more represented than for the malware case because it is used by several real-time applications. Indeed, this API is adapted to this type of applications: it allows to take screenshots in a fast and robust way by sending only the difference between two consecutive screens. However, this is a quite recent API (Windows 8 and above) compared to GDI, and this could explain why it is not used in malware.

Regarding screen capture triggering, screen captures are triggered only while the application is executed, and the command usually does not come from distant servers. However, this observation is unfortunately not systematic as some types of legitimate applications such as remote control respond to distant commands to establish the connection and start the screensharing session.

Some applications may have a similar behaviour to malware in the way screenshots are triggered. For example, Spyrix Free Keylogger takes screenshots each time the user switches window or opens a new window. Hubstaff, an employee control application, takes screenshots in an irregular way.

2) Screenshots format and storage

We can notice that legitimate applications do not seem to target overlapping windows, as opposed to some screenloggers (Figure 5).

Regarding screenshots storage, we observe different behaviours depending on the type of application. 57% of the selected applications only use memory representations of the captured screenshots and do not generate disk persistent storage. These are the screen sharing and remote-control applications This is due to the fact that these applications use temporary files which are erased after a real-time use of the data. On the contrary, the screen casting, screenshot editing and employee control applications store images in the disk.

An important part of legitimate applications for which obtaining the information was possible, use png and jpg formats for saving captured screenshots. Other applications use video formats such as AVI, MPEG4 and VMW to store images.

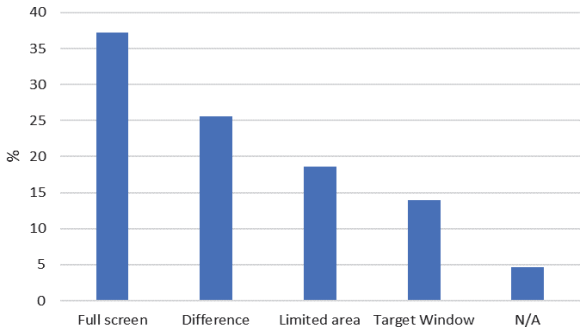

Fig. 5: Screen capture size (legitimate applications)

3) Exfiltration of captured data

An important part of the legitimate applications (45%) does not send the screenshots over the network because this is not their purpose (screenshots editing, screen casting). The other ones often use proprietary protocols (31%). Few ones use standard known protocols such as RFB (Remote Frame Buffer protocol) or HTTPS. Applications of our dataset mainly use SSL or TLS encryptions before files transmission (44%). Few of them, such as TightVNC and UltraVNC do not encrypt files.

E. Differences between legitimate and malicious applications

1) Screen capture

Malware only use the GDI library to capture screenshots while 17% of legitimate applications use Desktop Duplication API. Frequency is not an interesting criterion because, as malwares, legitimate applications display various behaviours: irregular screenshots (screenshot capture and editing), high frequency (screenshare, remote control), low frequency (employee/children monitoring). Regarding screenshots triggering, 67% of malware start captures in response to C2 server commands, whereas in legitimate applications screen capturing is strongly correlated to the use of the application and most of the time cannot occur in response to distant commands or events unrelated to the current application. However, this cannot be generalised because of children/employee control programs which have a behaviour akin to screenloggers.

2) Screenshot format and storage

An important difference that we noticed is the screen capturing area: a vast majority of malware take the full screen whereas it is more diverse for legitimate applications: only 37% of them capture the full screen, and the other ones either capture only part of the screen, a target window or only the difference between two successive screens. This contrast may be explained by the fact that most screenshot-taking malware do not look for a specific information on the infected device, but rather spy the user's activity in general.

Even if the format used by many legitimate applications for screenshots compression and storage remains unknown, it is possible to notice that the same formats are used both by malware programs and by legitimate applications to store and

send the images, the most commons being JPG and PNG. Video formats such as AVI may be used in both cases. Moreover, many of these applications use memory representations, for immediate local processing on the host machine. This trend is much less evident in the case of malware. Indeed, the security reports study shows a strong inclination towards the use of the hard drive for storage, but this observation does not occur in our dataset where memory storage takes the upper hand.

3) Exfiltration of captured data

An important difference lies in the fact that a large part of legitimate applications only uses screenshots to process them locally without transmitting them unlike malware which systematically send captured files over the network. Applications sending screenshots do it directly after the capture as they are real-time applications. A majority of malware have the same behaviour except few ones which may exhibit different sending patterns, such as sending the image files after a certain number of captures or upon reception of a command. Regarding the communication protocol, only one of the malware programs we analysed took the trouble to implement and integrate a proprietary protocol [51], as is the case of many legitimate applications. Thus, even if this cannot be considered as the only differentiation criterion, the use of a proprietary protocol by a software reduces the probability that it is malware. Encryption protocols used by malware and legitimate applications to encrypt images are quite different. Indeed, if a minority of legitimate applications do not encrypt the data they send, most of them encrypt it and for that they use the well-known standards SSL and TLS. On the other hand, 75% of malwares of the dataset do not even encrypt their data, and in the theoretical study it was observed that most of them use "home-made" and rudimentary encryption techniques combining XOR operations, base 64 encoding and symmetric encoding algorithms such as RC2 and RC4. Few of them use RSA and SSL/TLS. One of the possible explanations may be that asymmetric encryption often requires more computation and that's why it is less used by malware.

4) Synthesis

We mainly distinguish 3 classes of criteria according to their degree of differentiation between malware and legitimate applications (Table 2). Criteria like compression format or screenshot sending triggering don't seem to highlight enough differences between legitimate applications and malware to be used effectively in a detection methodology. In the second class we find criteria such as screenshot function library, storage or communication protocol. These are criteria presenting quite significant differentiation but with some specific cases and exceptions making generalisation difficult. The third and final class includes criteria, with significant degrees of differentiation, such as screenshot files sending and files encryption. However, a detection methodology limited only to these latter criteria cannot be effective because of the potential false positives. Thus, an effective malware screenshot detection approach must not only be based on the criteria of this third class but also integrate those of the second class with possibly different weights. Indeed, a software that uses a proprietary protocol with TLS/SSL encryption for network communication or which does not send the screenshots over the network is very likely to be a legitimate application, whereas one that does not encrypt data or just uses base64 encoding and that use a standard protocol such as TCP, FTP or SMTP might be a malware.

VI. Conclusion and Future Work

This work is the first one presenting a study of the screen capture functionality of malware. We proposed an in-depth analysis of a functionality which is quite unknown whereas it is one of the most prejudicial to privacy.

This study showed that there are several reasons explaining this lack of knowledge on screenloggers. The main one is that in the majority of cases, a specific event such as the reception of a command, the opening of an application of interest, or a number of mouse clicks, is needed to trigger the screen capture, which explains why it is neglected in the malware detection works. Studying the screenshot functionality implies being able to detect when a program captures the screen, which requires a meticulous analysis of the API calls reports, as many different sequences of functions can be called to take a screenshot.

We thus built a dataset of working screenloggers, ensuring that the screen capture functionality is enabled. More precisely, this work was the first to:

- Analyse API calls reports of more than ninety screenshot- taking programs to determine screen capturing API call sequences
- Analyse a hundred security reports to extract statistics about key aspects of the screnloggers' behaviour
- Propose a screenlogger classification according to several criteria
- Define completeness criteria for a screenlogger dataset
- Construct a dataset meeting those criteria and including legitimate screenshot-taking applications
- Implement a 'screenlogger generator' which corresponds to the completeness criteria
- Use the analysis results to give insights for screenloggers detection and differentiation from legitimate screenshot applications

This work is a first step towards the implementation of a detection technique for screen-capturing malware, which represents an important part of modern spyware. Since this functionality is ignored by current detection techniques, focusing on it could allow to detect powerful malware which are not detected otherwise.

Thus, more generally, this work proposes a new detection approach which, instead of mechanically running thousands of samples in a sandbox and relying on machine learning to find discriminative features, focuses on one specific functionality by analysing it more deeply both in malware programs and in legitimate applications to find differences and guidelines for detection. We deeply believe that the only effective detection method against screenloggers must be multi-criteria and give different weights to each criterion depending on whether it constitutes an important differentiation or not. This constitutes the objective of our future work.

Table 2: Ease / difficulty in differentiating between screenloggers and legitimate applications according to different criteria

		Used solution	Legitimate applications (%[11])	Malware [Mitre] (%[1])	Malware [Dataset] (%[1])	Diff. degree
Screen capture	Screenshot function library	GDI	76	59	100	++
		DD API	17	-	-	
	Screenshot capture triggering	Frequency	63	35	38	++
		App. of interest	3	9	3	
		Mouse clicks / User triggering	23	3	-	
		Command	-	38	59	
		Unique capture upon infection	-	5	-	
	Frequency		9ms to 1h	2s to 15mn	17ms to 60s	+
Format and storage	Format	JPG	7	25	56	+
		PNG	17	19	24	
		BMP	-	7	21	
		Video	17	1	-	
		Other	-	6	-	
	Storage	Memory	57	5	66	++
		Disk	43	50	34	
	Captured area	Full screen	37	37	87	++
		Coordinates	19	2	6	
		Difference	26	-	3	
		Other	14	6	3	
Exfiltration	Screenshot sending triggering	Real time flow	43	25	73	+
		Remote cmd	-	6	-	
		Scheduled	3	10	7	
		Other	-	13	-	
	Encryption?	No	7	-	75	+++
	Files sending?	No	50	-	-	+++
	Comm. protocol	HTTP	-	36	6	++
		HTTPS/FTP/SMTP/RFB	24	36	-	
		Proprietary	31	1	-	
		TCP	-	18	94	

REFERENCES

[1] *Raja Khurram Shazhad., Syed Imran Haider., Niklas Lavesson.: Detection of Spyware by Mining Executable Files. IEEE International Conference on Availability, Reliability and Security (ARES), pp. 295-302, Sweden (2010).*

[2] *G, Zhao., K. Xu., L. Xu., B. Wu.: Detecting APT malware infections based on malicious DNS and traffic analysis,. IEEE Access Journal, vol. 3, (2015).*

[3] *Mitre, Screen Capture, https://attack.mitre.org/techniques/T1113/ (Retrieved 18/02/2019).*

[4] *Counter Threat Unit Research Team. (2017, October 12). BRONZE BUTLER Targets Japanese Enterprises. Retrieved January 4, 2018.*

[5] *GReAT (Kaspersky Lab's Global Research and Analysis Team), Carbanak, The crime : The billion-dollar bank heist, (Retrieved 12/03/19)*

[6] *David E. Sanger., Nicole Perlroth.: Bank Hackers Steal Millions via Malware. The New York Times, 14 Feb. 2015 (Retrieved 12/03/19)*

[7] *New Jersey Cybersecurity & Communications Integration Cell, Zbot/Zeus, 19 October 2016, https://www.cyber.nj.gov/threat-profiles/trojan-variants/zbot (Retrieved 13/03/19)*

[8] *Symantec Security Response, Regin: Top-tier espionage tool enables stealthy surveillance, 27 August 2015*

[9] *Faou, M., Boutin, J.: (2017, February). Read The Manual: A Guide to the RTM Banking Trojan. Retrieved March 9, 2017.*

[10] *Department of Justice. (2018, August 01). HOW FIN7 ATTACKED AND STOLE DATA. Retrieved August 24, 2018.*

[11] *Bennett, J., Vengerik, B., (2017, June 12). Behind the CARBANAK Backdoor. Retrieved June 11, 2018.*

[12] *GReAT. (2017, November 1). Silence – a new Trojan attacking financial organizations. Retrieved May 24, 2019.*

[13] *Cylance. (2014, December). Operation Cleaver. Retrieved September 14, 2017.*

[14] *GReAT. (2019, May 13). ScarCruft continues to evolve, introduces Bluetooth harvester. Retrieved June 4, 2019.*

[15] *Falcone, R., Lee, B. (2018, November 20). Sofacy Continues Global Attacks and Wheels Out New 'Cannon' Trojan. Retrieved November 26, 2018.*

[16] *FinFisher. (n.d.). Retrieved December 20, (2017).*

[17] *Allievi, A.,Flori, E. (2018, March 01). FinFisher exposed: A researcher's tale of defeating traps, tricks, and complex virtual machines. Retrieved July 9, 2018.*

[18] *Iman Sharafaldin., Arash Habibi Lashkari, and Ali A. Ghorbani.: Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January (2018).*

[19] *Skype, https://www.skype.com/en/, Retrieved April 3, 2020.*

[20] *Screenleap, https://www.screenleap.com/, Retrieved April 3, 2020.*

[21] *Join.me, https://www.join.me/fr, Retrieved April 3, 2020.*

[22] *Teamviewer, https://www.teamviewer.com/fr/, Retrieved April 3, 2020.*

[23] *Netwiewer, http://www.tucows.com/preview/613992/Netviewer-Support, Retrieved April 3, 2020.*

[24] *GoToMyPC, https://get.gotomypc.com/, Retrieved April 3, 2020.*

[25] *Camtasia, https://www.techsmith.fr/camtasia.html, Retrieved April 3, 2020.*

[26] *CamStudio, https://camstudio.org/, Retrieved April 3, 2020.*

[27] *Ezvid, https://www.ezvid.com/, Retrieved April 3, 2020.*

[28] *Verity, https://www.nchsoftware.com/childmonitoring/index.html, Retrieved April 3, 2020.*

[29] *Kidlogger, https://kidlogger.net/, Retrieved April 3, 2020.*

[30] *Picpick, https://picpick.app/fr/, Retrieved April 3, 2020.*

[31] *Snipping Tool, https://support.microsoft.com/en-us/help/13776/windows-10-use- snipping-tool-to-capture-screenshots, Retrieved April 3, 2020*

[32] *FastStone Capture, https://www.faststone.org/FSCaptureDetail.htm, Retrieved April 3, 2020.*

[33] *Microsoft Documentation, SetWindowDisplayAffinity function, https://docs.microsoft.com/en-us/windows/desktop/api/winuser/nf-winuser-setwindowdisplayaffinity, Retrieved April 3, 2020.*

[34] *Chrome Developer Documentation, captureVisibleTab, https://developer.chrome.com/extensions/tabs#method-captureVisibleTab (Retrieved 18/02/19)*

[35] *html2canvas documentation, http://html2canvas.hertzen.com/documentation/ (Re- trieved 19/02/19)*

[36] *PandaSecurity, Watch out for Chrome and Firefox web extensions that access browser history and rob passwords, August 2016, https://www.pandasecurity.com/mediacenter/malware/malicious-web-extensions/ (Retrieved 05/02/2019)*

[37] *Virusshare, https://virusshare.com/, Retrieved April 3, 2019.*

[38] *AVCaesar, https://avcaesar.malware.lu/, Retrieved April 3, 2019.*

[39] *Malshare, https://malshare.com/, Retrieved April 3, 2019.*

[40] *VirusSign, https://www.virussign.com/, Retrieved April 3, 2019.*

[41] *Wireshark, https://www.wireshark.org/, Retrieved April 3, 2019.*

[42] *API Monitor, http://www.rohitab.com/apimonitor, Retrieved April 3, 2020.*

[43] *Inetsim, https://www.inetsim.org/, Retrieved April 3, 2020.*

[44] *Javaheri, D., Hosseinzadeh, M., & Rahmani, A. M. (2018). Detection and Elimination of Spyware and Ransomware by Intercepting Kernel-Level System Rou- tines. IEEE Access, 6, 78321-78332.*

[45] *Ebach, L. (2017, June 22). Analysis Results of Zeus.Variant.Panda. Retrieved November 5, 2018.*

[46] *Parys, B. (2017, February 11). The KeyBoys are back in town. Retrieved June 13, 2019.*

[47] *Legezo, D. (2019, January 30). Chafer used Remexi malware to spy on Iran- based foreign diplomatic entities. Retrieved April 17, 2019.*

[48] *Balanza, M. (2018, April 02). Infostealer.Catchamas. Retrieved July 10, 2018.*

[49] *PowerShellMafia. (2012, May 26). PowerSploit - A PowerShell Post-Exploitation Framework. Retrieved February 6, 2018.*

[50] *PowerSploit. (n.d.). PowerSploit. Retrieved February 6, 2018.*

[51] *Mandiant. (n.d.). Appendix C (Digital) - The Malware Arsenal. Retrieved July 18, 2016.*

[52] *Grunzweig, J. and Miller-Osborn, J.. (2016, February 4). T9000: Advanced Modular Backdoor Uses Complex Anti-Analysis Techniques. Retrieved April 15, 2016.*

[53] *Hromcová, Z. (2018, June 07). InvisiMole: Surprisingly equipped spyware, undercover since 2013. Retrieved July 10, 2018.*

[54] *Tsarfaty, Y. (2018, July 25). Micropsia Malware. Retrieved November 13, 2018.*

[55] *Sardiwal, M, et al.: New Targeted Attack in the Middle East by APT34, a Sus- pected Iranian Threat Group, Using CVE-2017-11882 Exploit. Retrieved Decem- ber 20, 2017.*

[56] *Kasza, A. and Reichel, D.. (2017, February 27). The Gamaredon Group Toolset Evolution. Retrieved March 1, 2017.*

[57] *Scott-Railton, J., et al. (2016, August 2). Group5: Syria and the Iranian Connection. Retrieved September 26, 2016.*

[58] *ClearSky Cyber Security and Trend Micro. (2017, July). Operation Wilted Tulip: Exposing a cyber espionage apparatus. Retrieved August 21, 2017.*

[59] *Vox, Why coronavirus scammers can send fake emails from the WHO, https://www.youtube.com/watch?v=_CrbHvbvvMw, Retrieved April 14, 2020*

[60] *Gostev, A. (2012, May 28). The Flame: Questions and Answers. Retrieved March 1, 2017.*

[61] *Lee, B. and Falcone, R. (2017, February 15). Magic Hound Campaign Attacks Saudi Targets. Retrieved December 27, 2017.*

[62] *Ray, V., Hayashi, K. (2016, February 29). New Malware 'Rover' Targets Indian Ambassador to Afghanistan. Retrieved February 29, 2016.*

[63] *Hugo Sbai, Michael Goldsmith, Samy Meftali, Jassim Happa: A Survey of Keylogger and Screenlogger Attacks in the Banking Sector and Countermeasures to Them. CSS 2018.*

[64] *https://github.com/sbhugo/Screeminals.git*

[65] *Cherepanov, A.. (2016, May 17). Operation Groundbait: Analysis of a surveillance toolkit. Retrieved May 18, 2016.*

[66] *Nart Villeneuve, James T. Bennett (2014, February 19). XtremeRAT: Nuisance or Threat? Retrieved April 3, 2020.*

[67] *Yan, T., et al. (2018, November 21). New Wine in Old Bottle: New Azorult Variant Found in FindMyName Campaign using Fallout Exploit Kit. Retrieved November 29, 2018.*

[68] *McAfee. (2015, March 2). Netwire RAT Behind Recent Targeted Attacks. Retrieved February 15, 2018.*

[69] *Blaich, A., et al. (2018, January 18). Dark Caracal: Cyber-espionage at a Global Scale. Retrieved April 11, 2018.*