Extracting Actionable Data from Banking Malware[1]

Authors

James Wyke, MSc (Royal Holloway, 2015)

Frederik Mennes, ISG, Royal Holloway

Abstract

This article will open the door on modern banking malware, giving readers an understanding of what this kind of malware can do and how it works. We will help the reader gain a deeper understanding of banking malware based on the forensic artefacts left behind after execution, so that network defenders can better protect against future infections and more thoroughly understand the consequences of a compromise.

We will define actionable data, demonstrate how it can be extracted from banking malware, and describe how the reader can use this data to help defend against highly damaging cyber-attacks from organised criminal gangs.

Introduction

Banking malware has been around for over a decade and in that time it has become more sophisticated and the criminals using it have become more successful at stealing money from victim's bank accounts.

We have seen a change in the calibre of actor deploying this kind of malware, from teenagers working alone out of their bedrooms, to organised criminal groups with large resources and highly skilled personnel. Many gangs target the more lucrative accounts of businesses and other organisations with large balances, rather than accounts belonging to individuals, so as to maximise the return a compromise can yield.

Groups operating high profile banking malware such as *Dridex, Zeus, Dyreza* and *Vawtrak* have defrauded victims all around the world to the tune of many millions of dollars. Modern banking malware is sophisticated and effective and should be a serious concern for any IT professional charged with defending corporate networks.

[1] This article is to be published online by Computer Weekly as part of the 2016 Royal Holloway information security thesis series. It is based on an MSc dissertation written as part of the MSc in Information Security at the ISG, Royal Holloway, University of London. The full MSc thesis is published on the ISG's website.

An overview of banking malware

Modern banking malware is chiefly concerned with stealing any kind of data that can lead to a financial advantage for the people who are operating the malware. This can include usernames and passwords – not just for online banking but also for other applications such as email, FTP clients, and social networks, but ultimately the criminals who are running the malware want access to your (or your company's) bank account so that they can transfer money out of it and into their own. Here is a breakdown of some of the different types of data that the criminals are after:

- Application credentials – FTP and email clients in particular.
- Web page credentials – banking websites and many more such as social media, customer resource management, shopping.
- Keystrokes – all keystrokes entered, or just keystrokes entered into specific applications.
- Screenshots – still captures of the screen when certain events occur such as mouse button presses.
- Screen video – full video capture of the screen when certain URLs are visited.
- Contacts – email contacts can be used to further spread the malware.

There are a wide variety of techniques that a piece of malware can use to obtain this kind of data. In some cases, data like usernames and passwords can be obtained by simply calling certain Windows APIs. For example, any passwords saved in browsers or email clients can be retrieved by software running on the same system. However, to get the really juicy data from your online banking session and to do something constructive with it – like automatically transfer a certain amount out of your account and into someone else's - the criminals use a technique called Man-in-the-Browser (MitB).

MitB is a process whereby the malicious software injects code into a browser process and takes complete control over it. The injected code will hook certain APIs that the browser uses, intercepting and modifying the data that passes through them. This means the malware can read and modify any web pages that the victim browses to, and can intercept any data that the victim enters such as usernames and passwords.

Naturally, one of the most common pieces of data that the criminals want to intercept is credential information. This can be specifically targeted by concentrating on data that has been entered into 'form' fields on a web page. This is often known as 'form grabbing'.

Criminals are generally pretty greedy and they will try and steal as much as they can from an infected victim. One example of this is attempting to capture more information than the user would normally enter when they login to their online bank account, for example ATM PIN numbers or social security numbers. They achieve this using a feature that most banking malware families include, called "web injects". These are snippets of HTML and JavaScript code that are injected into specific locations in web pages that are of interest to the criminals.

Figure 1 shows an example where code has been injected into a web page that asks the victim for various other items of data that would not normally be asked for at login, such as "name of your first school".

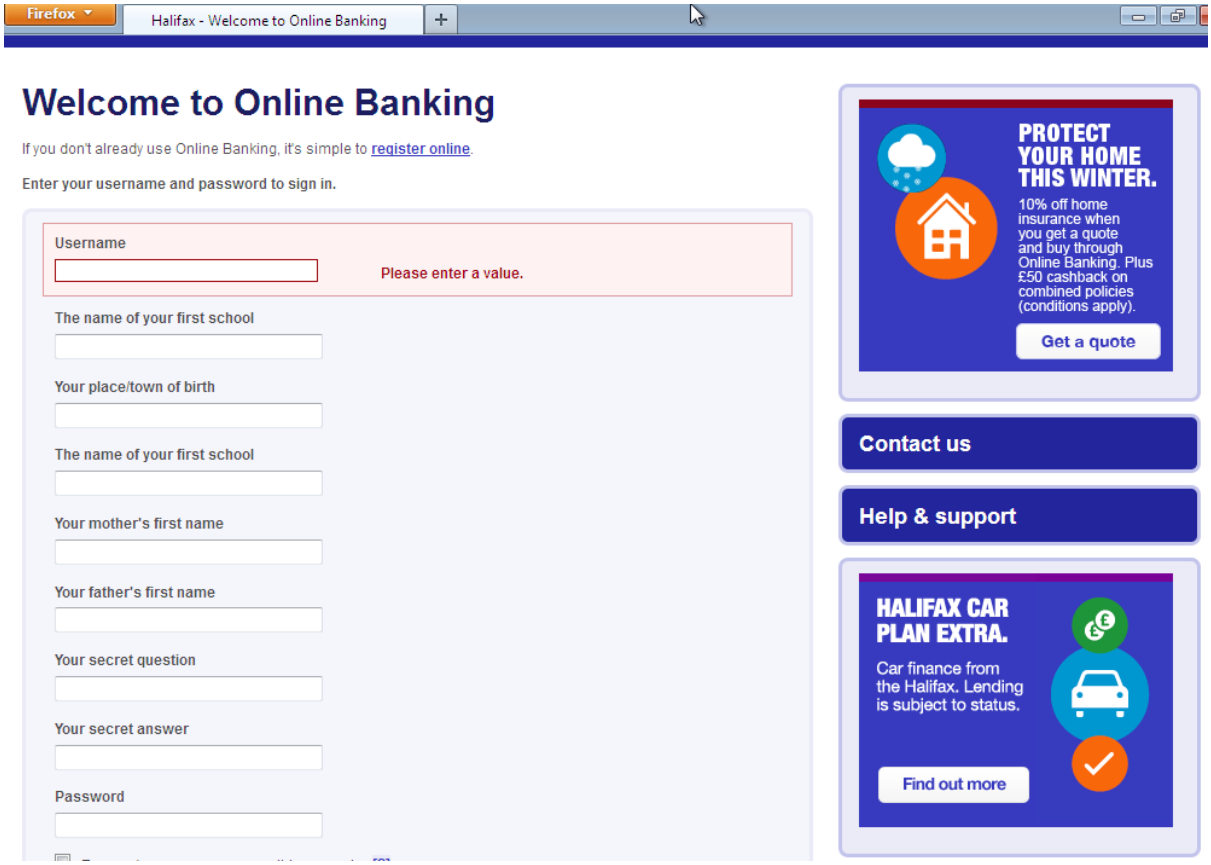

Figure 1 – injected code

If my bank requires 2FA I'm good, right?

Over the years, banks have introduced various measures to try and curb fraud carried out by criminals using banking malware. The criminals have responded by adding features to their malware that add the capability of bypassing these measures.

One notable countermeasure introduced by many banks is to add a second layer of authentication for certain types of functionality, such as initial login or funds transfer to a new recipient. This Two Factor Authentication (2FA) is usually a one-time-password (OTP) that the user accesses in one of several ways, such as a small hardware device, a SMS message, or a soft token generated from an application on their smart phone.

Two Factor Authentication (2FA):

- Something you *know*
- Something you *have*

2FA systems such as these increase the security of the online banking system because they mean that the user requires something that they **know** – their username and regular password – and something that they **have** - the device that gives them the one-time-password.

This presents a problem to criminals using banking malware. They may capture the username and normal password entered by the victim, and they may capture the one-time-

password as the victim logs in. However, this OTP will not be of any use when the criminal tries to log in to the bank account at a later time, as the OTP was only valid for one occasion.

However, the criminals have developed several techniques to get around 2FA and still transfer money out of the victim bank account. One technique uses a combination of clever "web injects" and social engineering. The criminals inject HTML and JavaScript code into the page after the victim has logged in that suggests there has been some sort of problem with the victim's bank account. Maybe the user will be presented with a message that says a mistaken transfer was made into their account and access has been suspended until the error can be resolved. However, the victim can make the problem disappear by approving the transfer error which will give them their access back immediately. The user just needs to enter the OTP and the erroneous transfer will be backed out. In reality, the OTP is used to initiate a fraudulent transfer to a bank account under the criminal's control.

This is just one of the many ways that criminals can be very creative in their efforts to bypass countermeasures introduced by banks.

What is actionable data and what can I do with it?

It's important to note that *actionable data* can mean different things to different people. In the context of Information Security we can borrow the definition from the European Union Agency for Network and Information Security:

"Actionable information is used to take actions that mitigate against future threats, or help address existing compromises."

When we are talking about actionable data and banking malware we must first establish what we might be able to achieve with such data. Broadly speaking, we are looking for data that will help us detect a compromise by this banking malware and for data that will help us during an investigation into the implications of a compromise and into those who have perpetrated the attack.

The following are some of the different types of data that we can extract from banking malware to help us achieve these objectives:

- **Command and control addresses.** These are the primary means by which we can look for evidence of a compromise at the network level. Any machine that is beaconing to one of these addresses may be compromised. Many banking malware families also have multiple backup command and control addresses so we need to make sure we have all of them.
- **System artefacts.** Things like filenames, registry keys and mutexes that can be used as indicators of compromise.
- **Stolen data.** We are very interested in knowing what data may have been stolen so that we can assess what remediation steps need to be taken such as password changes.
- **Attacker issued commands.** Most banking malware families include the ability for the criminals to issue certain commands to the infected machine. These are generally the type of command that typical Bots can receive such as finding and uploading files from the victim machine or downloading and executing further malware. We would like to know the details of any of these commands.

- **Other network artefacts.** These may be further indicators that we use to identify the malware across the network when command and control addresses that we don't know about are used. For example if the malware makes a HTTP request using a specific URL format we can write an IDS rule to flag up this activity.
- **Configuration files and "web injects".** This data will generally be downloaded by the malware during execution. It can give us further network indicators and can be a huge help in understanding the intentions of the criminals behind the malware – which banks are they targeting and what kind of extra information were they attempting to get from the victim.

Many of these data elements are useful to both network defenders and investigators.

How can it be extracted?

Now that we have established the type of data that we want extract from banking malware, we must work out how we can go about actually extracting it.

After a banking malware sample has executed on a victim machine, there are three main areas where we can seek actionable data: **memory, disk** and **network traffic**.

The memory of the infected machine holds all the data used by the malware program. The program must be able to communicate with command and control servers, download and decode configuration files, and act on any received commands. This means that all data such as addresses and cryptographic keys must be stored somewhere in the program's memory.

Trying to get all this information out of system memory can be a tricky business. For example we might try and approach things by looking at all the memory on the victim machine. To start with, we can use one of the various tools that exist to capture a binary image of physical memory. However, this is literally just a very large lump of binary data. We may find some interesting pieces of data by simply *grepping* through the file – such as domains, URLs or pieces of a downloaded configuration file - but if data is encrypted or obfuscated we will not have much luck. Similarly, when we want to find data structures used by the malware, the complexity of memory management systems in modern operating systems means that it is very difficult to locate the regions of memory used by the malware program when all we have is one large binary file.

Another approach might be to attach a debugger to the infected system and inspect the processes that we think are involved in the malware's operations. This can be very powerful as many debuggers can be scripted which means we can automate much of the process. We can also dump specific regions of memory – such as blocks of memory that may have been injected into a system process – for later analysis. With some reverse engineering we can work out exactly where the malware stores its interesting data and then write a script for the debugger that will find that location and grab the data for us. For example, we may reverse engineer a sample and work out that it stores a list of command and control addresses, a campaign ID and an AES key used to encrypt network traffic, encrypted inside the *resource section* of the *PE* file. If we are using the *Immunity Debugger* we could write a script in Python to unpack the sample, locate the section, then decode and extract the contents. One possible downside with this approach is that we need to interact with the sample on a test machine with the debugger, which is a somewhat manual process and may not be possible in all situations.

Another approach we can take is to go back to our full capture of physical memory and use a memory forensics tool to help us make sense of the binary mess. One such tool is called *Volatility*. *Volatility* is a framework written in Python that takes a memory dump and gives the analyst an interactive, scriptable interface into the running state of the system when the memory image was taken. It includes a large number of commands, or plugins that can be used to extract digital artefacts from the image. These include things like running processes, open network connections, file and registry handles and much more besides. We can also write our own custom plugins to extract useful data from malware.

The disk will hold the binary file belonging to the malware and any data files that it may store. Registry hives are also located on the disk and many banking malware families store important information such as configuration files and binary modules in the registry. We can pull files and registry hives straight from the disk of an infected machine, or we can take a forensic image and use tools like *The Sleuth Kit* to extract the data that we consider useful from the image.

Network traffic will contain all communication that the malware has carried out with its command and control servers, including downloaded files and stolen data. If we want to ascertain the full impact of a compromise we will need access to a capture of network traffic from the infected machine and the means to decode it, possibly using cryptographic keys that we obtained from analysis of the memory dump. *Figure 2* shows a representation of the commands that were received and executed by a sample of the *Vawtrak* or *Neverquest* banking malware family.

```
CMD received:

CMD_NAME: UPDATE_NO_REBOOT
HOST: segropa.com
URI: /collection/00000042/00/C7704535
DATA_TYPE: CMD
ARG: http://dedorka.com/upd/66?id=3346023733&o=12&n=15

CMD received:

CMD_NAME: PROCESS_LIST
HOST: segropa.com
URI: /collection/00000042/00/C7704535
DATA_TYPE: CMD
ARG:

CMD received:

CMD_NAME: GRAB_PONY
HOST: segropa.com
URI: /collection/00000042/00/C7704535
DATA_TYPE: CMD
ARG:
finished extracting data
```

Figure 2 – Vawtrak commands

The figure shows that three commands were received during analysis and we can see which command it was each time and any associated data too. The first command is called "UPDATE_NO_REBOOT" which unsurprisingly means an update will be performed without a reboot. We can see the URL and host from which the command was received and the argument to the command which in this case is the URL from which the update will be downloaded. The second command is called "PROCESS_LIST" which will cause the malware to enumerate all processes on the infected machine and send the list back to the command and control server. This is sometimes performed to try and spot interesting software that might be installed on the system, such as Point of Sale software that would indicate the machine is processing payment card data. The third command is called "GRAB_PONY" which grabs the saved passwords and usernames from a wide variety of software such as email clients and FTP software and sends them back to the attacker. This command is called *PONY* because it uses code from another well-known password stealing malware family called *Pony.*

By combining elements of the data extracted from each data source we can build a comprehensive view of the malware's activity and use the data to identify further compromises and aid an investigation.

Conclusion

We have merely touched on some of the aspects of banking malware and how we can gain a more thorough understanding of the actions and intentions of the criminals that use it.

We have given a brief introduction to the capabilities of banking malware and the techniques that are used. We have demonstrated some of the ways that modern examples attempt to bypass sophisticated countermeasures that banks have introduced. We have touched on the types of information we can extract from banking malware samples and how we can use that information to defend our networks.

We hope that this article may act as a starting point for further investigation for those interested in this exciting and rapidly evolving field.

Biographies

James Wyke is a Senior Threat Researcher with Sophos where he has worked for eight years. James has published and presented on a range of research topics at a variety of industry conferences including VB, CARO and SOURCE. His research interests include botnets, banking malware, automated analysis, APTs and data mining.

Frederik Mennes heads VASCO's Security Competence Center, working on the security aspects of VASCO's products and infrastructure. He is a regular speaker at industry events and conferences about security technology, and a contributor to the Initiative for Open Authentication (OATH). Besides his role at VASCO, Frederik has supported the Information Security Group (ISG) at Royal Holloway, University of London in various educational roles. He earned an MBA from Vlerick Business School (Belgium), an M.Sc. in Information Security from Royal Holloway, University of

London, and an M.Sc. in Computer Science Engineering from KU Leuven, Belgium. When not in the office, you will probably find him climbing a mountain or playing the piano.