TLP: WHITE

ENJOY SAFER TECHNOLOGY™

JUMPING THE AIR GAP:

15 years of nation-state effort

Authors:
Alexis Dorais-Joncas
Facundo Munõz

December 2021

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. EXECUTIVE SUMMARY

Warning: This report contains references to material from the Vault7 leak, which may contain classified information.

Air-gapping is used to protect the most sensitive of networks. In the first half of 2020 alone, four previously unknown malicious frameworks designed to breach air-gapped networks were publicly documented. ESET Research decided to revisit each framework known to date and to put them in perspective, side by side.

Here are the key findings stemming from this exhaustive study:

- All the frameworks are designed to perform some form of espionage.
- All the frameworks used USB drives as the physical transmission medium to transfer data in and out of the targeted air-gapped networks.
- We have not found any case of actual or suspected use of covert physical transmission mediums, such as acoustic or electromagnetic signals.
- Over 75% of all the frameworks used malicious LNK or autorun files on USB drives to either perform the initial air-gapped system compromise or to move laterally within the air-gapped network.
- More than 10—critical severity—LNK-related remote code execution vulnerabilities in Windows have been discovered, then patched by Microsoft, in the last 10 years.
- All the frameworks were built to attack Windows systems. We have not found any evidence of actual or suspected malware components built to target other operating systems.

In this white paper, we will describe how malware frameworks targeting air-gapped networks operate, and provide a side-by-side comparison of their most important TTPs. We also propose a series of detection and mitigation techniques to protect air-gapped networks from the main techniques used by all the malicious frameworks publicly known to date.

2. INTRODUCTION

Air-gapping is used to protect the most sensitive of networks: voting systems, industrial control systems (ICSes) running power grids, and SCADA systems operating nuclear centrifuges, just to name a few. In the first half of 2020 alone, four malicious frameworks designed to breach air-gapped networks emerged, bringing the total, by our count, to 17. This prompted us to step back and revisit each of those frameworks from the vantage point of having discovered and analyzed firsthand three of these in the past six years. Using the knowledge made public by more than 10 different organizations over the years, and some ad hoc analysis to clarify or confirm some technical details, we put the frameworks in perspective to see what history could teach us in order to improve air-gapped network security and our abilities to detect and mitigate future attacks.

This exhaustive study allowed us to isolate several major similarities in all of them, even those produced 15 years apart. Specifically, we focused our attention on the malware execution mechanisms used on both the connected and the air-gapped side of targeted networks and the malware functionalities within the air-gapped network (persistence, reconnaissance, propagation, espionage, and—at least in one case—sabotage activities), with a focus on the communication and exfiltration channels used to cross the air gap barrier and control the components running on the isolated networks. This also resulted in a systematic analysis structure that may be reused to document air-gapped malware that is discovered in the future.

Despite some differences and nuances found across all frameworks studied, our analysis shows how most differ on many of those aspects only from an implementation perspective, mostly due to the severe constraints imposed by air-gapped environments. Armed with this information, we will highlight some detection opportunities specific to the actual techniques observed in the wild.

Our aim is to convince the reader of the importance of having all the proper defense mechanisms to mitigate the techniques used by virtually all of these frameworks that have been observed in the wild, before starting to look into the many theoretical air gap bypass techniques that have received a lot of attention in recent years despite none of them ever being used in a real, publicly disclosed attack.

3. VICTIMOLOGY, ATTACKER PROFILES, TIMELINE

An air-gapped network is one that is physically isolated from any other networks in order to increase its security. Air-gapping is a technique used to protect networks interconnecting the most sensitive and high-value systems within an organization, systems that are naturally of high interest to numerous attackers, including any and all APT groups.

We can state without fear of contradiction that threat actors behind the known malware frameworks designed to attack air-gapped networks all belong to the advanced persistent threat (APT) category. Some frameworks have been attributed to well-defined, well-known threat actors:

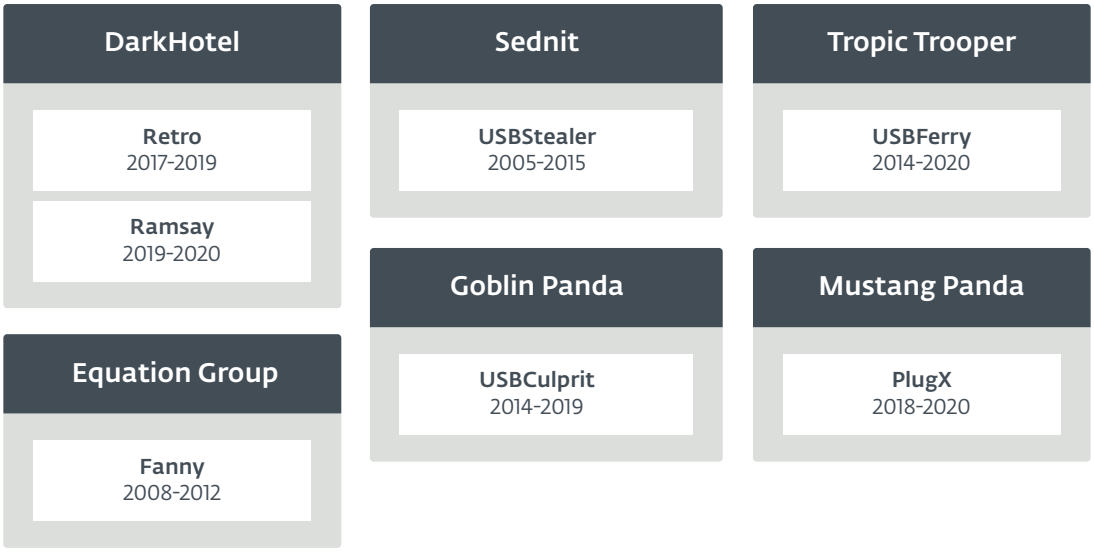

Figure 1 // Frameworks attributed with high confidence to known threat actors

For others, the attribution has been less clear-cut, speculative or controversial. Agent.BTZ, for example, has been attributed to Turla [1], but other experts are not so convinced [2].

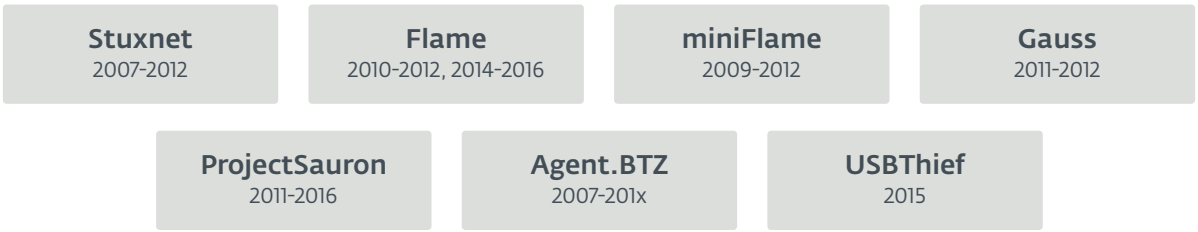

Figure 2 // Frameworks where attribution could not be determined or only in a speculative way

Finally, we have a trilogy of frameworks that constitute our special cases: these frameworks have been found in documentation from the Vault7 leaks and are described to have been in operation in a time range from 2013 to 2016; however, we haven't found samples in the wild to analyze first hand.

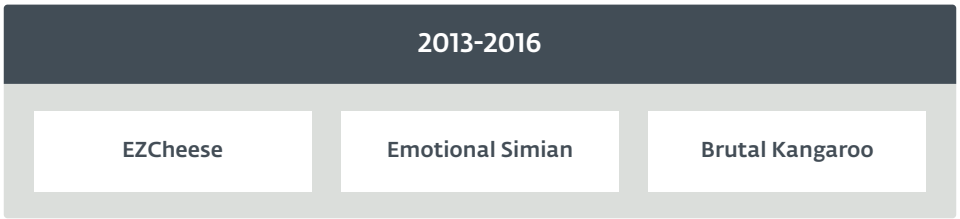

Figure 3 // Frameworks documented in the Vault7 leak

Despite the variety of threat actors behind these frameworks, all of them shared a common purpose: espionage. Even Stuxnet, best known for its sabotage capabilities, collected information about Siemens Simatic Step 7 engineering software projects found in compromised machines.

Due to the nature of its target and its capabilities to operate and propagate, Stuxnet has often been referred to as the first malware designed to attack systems in air-gapped networks. However, research published years after the Stuxnet discovery determined with reasonable confidence that a sample of Sednit's USB-Stealer dates from 2005 [3]. Figure 4 shows a historical view of the period of activity of each framework, along with the time of the first public report. This is also an indication of how difficult it is to detect this type of framework.

Note that the periods of activity are based on what has been reported publicly; in some cases, the researchers were not able to determine a precise period of activity based on observable facts but are rather approximated or inferred by using some reasonable hypotheses.

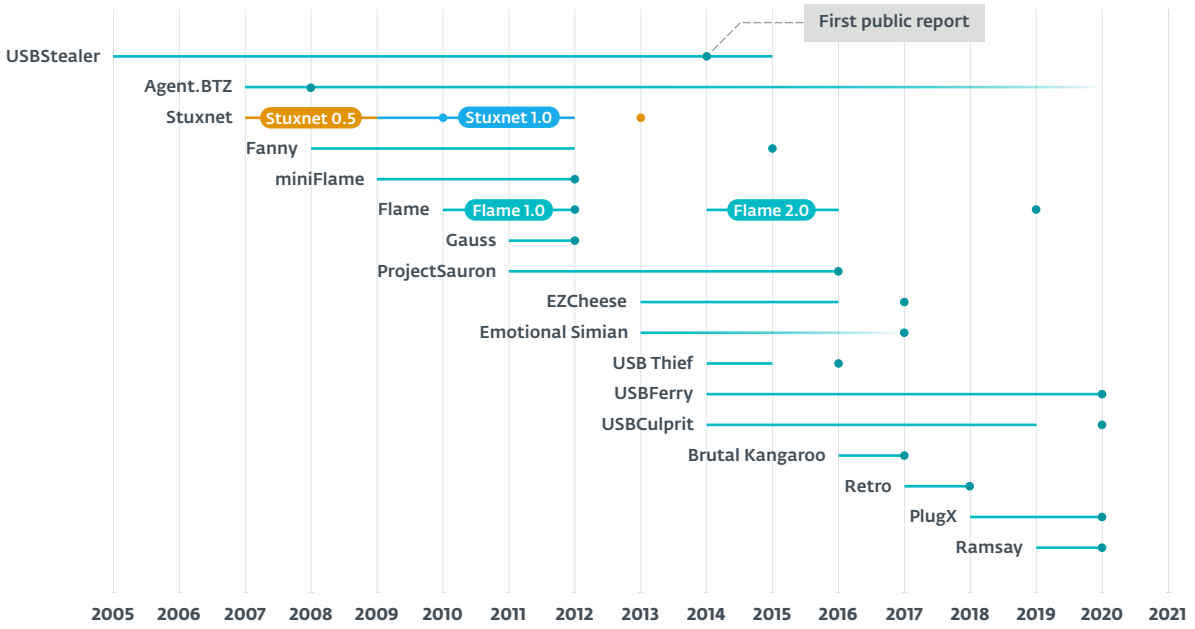

Figure 4 // Period of activity of all known frameworks and date of the first public report

This timeline highlights an important point:

- The majority of the frameworks were active for many years before being noticed, analyzed and reported publicly.

Contrary to the many malware families that play cat-and-mouse with security researchers by evolving even after they are exposed, most air-gapped frameworks haven't remained active long after the first analysis was published. This can be explained either by the fact that the analysis caused the attackers to consider their tool burned and to stop using it, or that the discovery and analysis happened after the framework stopped being used for unrelated reasons. There is also the unsettling possibility that antimalware solutions in those networks did not get updated and hadn't been able to detect these threats.

4. ANATOMY OF AIR-GAPPED SYSTEMS—A MALWARE PERSPECTIVE

Attack and compromise of systems in air-gapped networks require the attackers to develop capabilities that enable their tools to communicate via channels that are not commonly required in normal operations. It's obvious: they have to contend with the fact that these networks are isolated from the internet. In this section we discuss those specific areas where air-gapped malware tends to operate.

Before we begin, it is important to note that attacks against air-gapped networks are not all done in the same way and, in fact, there is no precise definition of what "air-gapped malware" actually is from the purely technical perspective. This sparked some lively discussions internally, until we finally agreed upon—for the purpose of this paper—the following definition for air-gapped network malware:

Malware, or a set of malware components acting together (a framework), that implements an offline, covert communication mechanism between an air-gapped system and the attacker that can be either bi-directional (command and response) or unidirectional (data exfiltration only).

Some frameworks have been left out intentionally from this study because we could not reliably determine that they fit in our definition. For example, it has been reported that DarkHotel has used Asruex since 2015 to attack isolated networks [4]; however, publicly available reports do not provide enough technical evidence demonstrating the presence of the minimal requirements needed to satisfy our working definition.

Interestingly, all the known frameworks were designed to attack Windows systems only.

Let's "zoom out" and have a look at the big picture. We decided to separate the frameworks into two broad categories: connected and offline. Most frameworks are built to provide fully remote end-to-end connectivity between the attacker and the compromised systems on the air-gapped side. We call these "connected frameworks". The general operating schema looks like this:

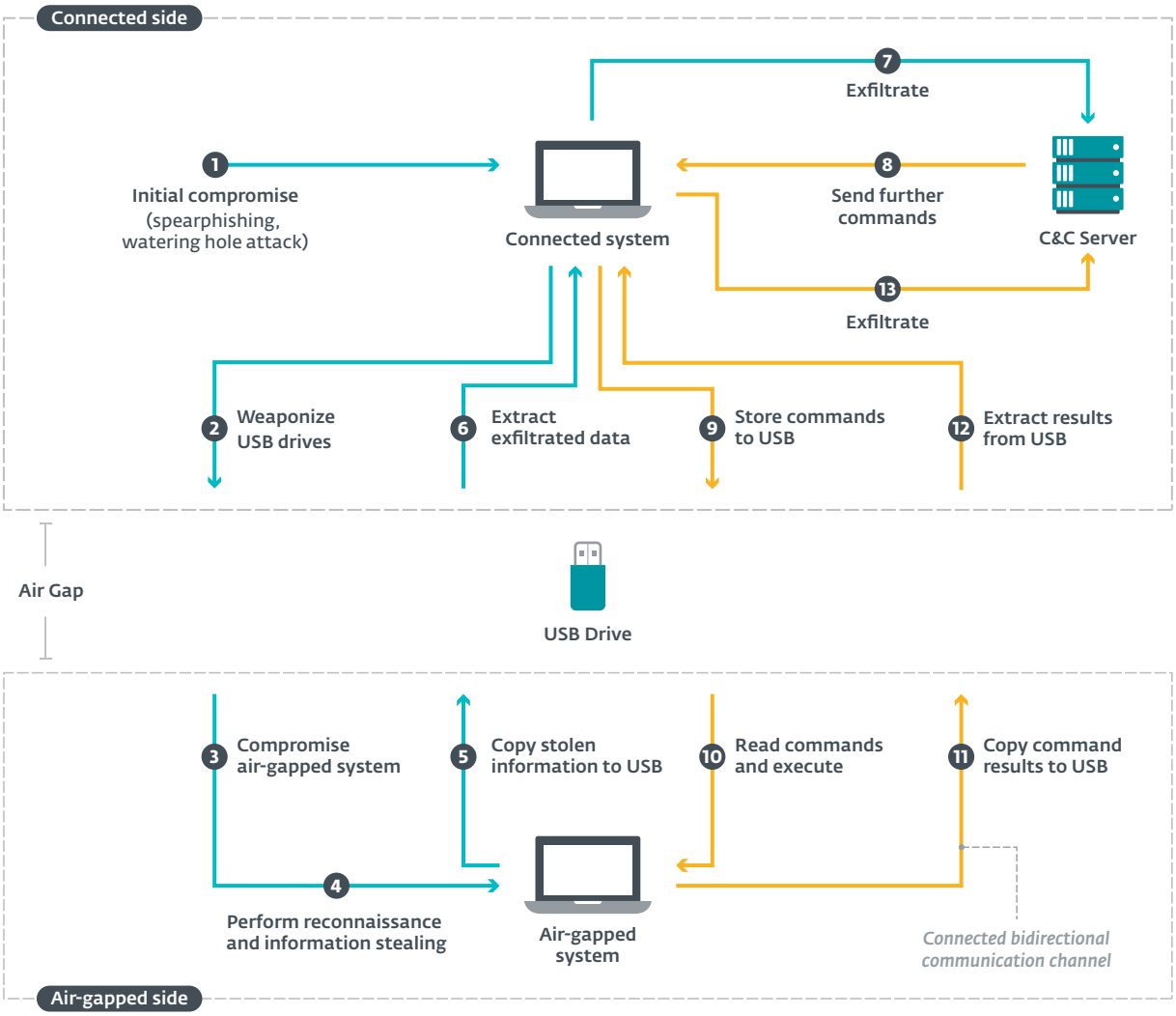

Figure 5 // Overview of the components and actions of a connected framework designed to attack air-gapped networks

In this scenario, the attack first targets an internet-connected system that is used alongside an air-gapped network ❶. Once compromised, that system is used to weaponize USB drives with a malicious payload and some mechanism to compromise the next target: the air-gapped system ❷. This is usually made possible because the USB drive is used to transfer information between both sides, in most cases by an unsuspecting victim.

The payload running on the air-gapped system usually comes with reconnaissance and information stealing capabilities ❹, whose output is stored back onto a USB drive ❺. When the USB drive reaches the compromised connected system, its contents are extracted ❻ and the data is exfiltrated to the attacker via the Internet ❼.

Some frameworks go one step further and support a two-way communication protocol: through a compromised system in the connected side, the attacker sends commands to the malware placed on the air-gapped network; this is done via a covert communication channel often placed on a USB drive (❽/❾/❿/⓫/⓬/⓭). These are the most powerful frameworks, granting the attackers the ability to run arbitrary code inside air-gapped networks.

In the other, rarer cases such as Ramsay and USBThief, the attack scenario does not involve any internet-connected systems at all. We call these "offline frameworks". In these cases, everything indicates the presence of an operator or collaborator on the ground to perform the actions usually done by the connected part of connected frameworks, such as preparing the initial malicious USB drive responsible for the execution on the air-gapped side ❷, executing the malware on the air-gapped system ❸, extracting the exfiltrated data from the drive ❻ and sending additional commands to the air-gapped side ❾. Figure 6 show how much the operating schema becomes simplified, from the malware perspective at least:

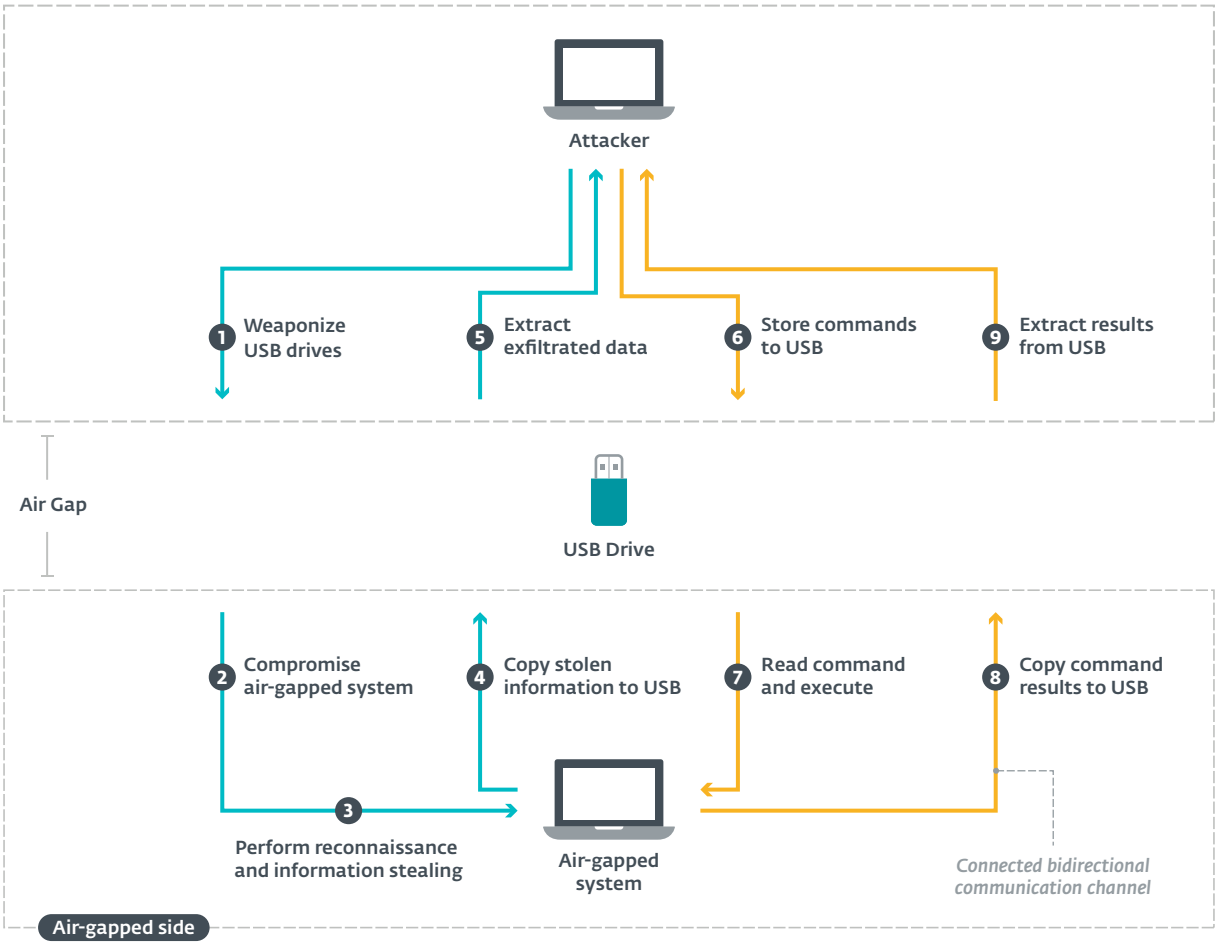

Figure 6 // Overview of the components and actions of an offline framework designed to attack air-gapped networks

4.1 Connected side execution vector

For connected frameworks, the first step to successfully compromise the air-gapped network is to get a foothold on a system that has internet connectivity, as Figure 7 shows. Unfortunately, when it comes to APTs, it's not always possible to know exactly how this happened but for the cases that we do know, the methods observed do not differ much from what we see in general malware: emails with malicious attachments, links, or USB worms. Table 1 summarizes the various methods.

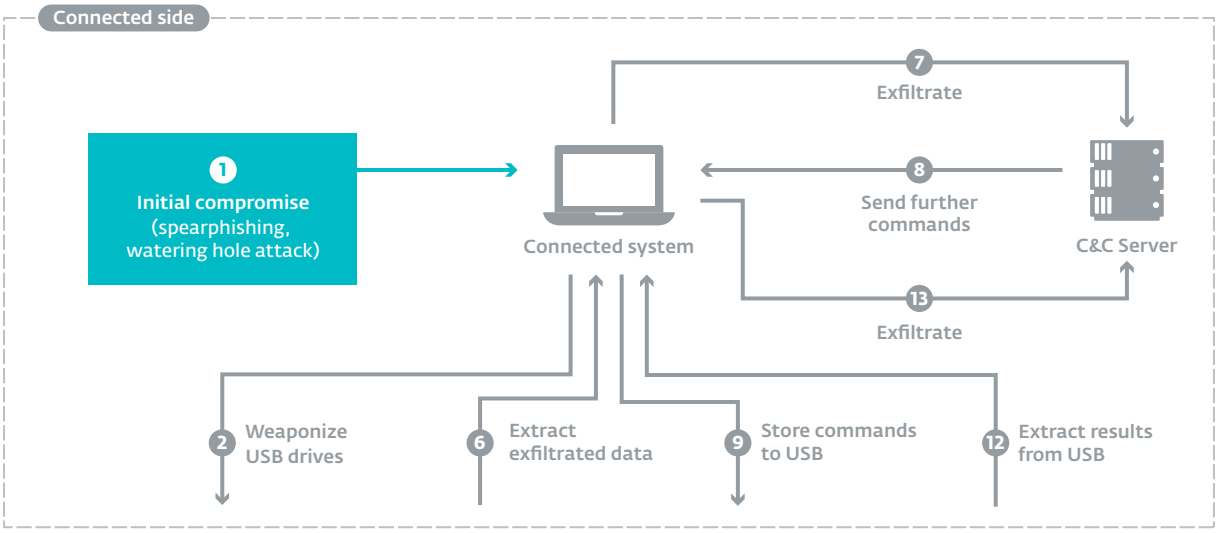

Figure 7 // Connected side system compromise. Part of Figure 5.

	Compromise method	Details
Connected frameworks		
USBStealer	Email (suspected)	Suspected spearphishing with malicious attachments.
Agent.BTZ	USB worm	Spreads by copying itself to USB drives with an autorun.inf file as execution vector.
Stuxnet	Unknown	It is not clear whether the compromission started via a connected system or was performed by a human asset with physical access to the air-gapped network.
Fanny	Unknown	Possibly downloaded and installed by another component at the request of the attackers.
miniFlame	Unknown	Possibly downloaded and installed by Gauss or Flame
Flame	Unknown	It is suspected that an exploit is used for the initial compromise of a potential victim.
Gauss	Unknown	Gauss does not propagates itself, it is unknown how victims get compromised by it.
USBFerry	Email	Malicious software installer in attachment.
USBCulprit	Email	Spearphishing with malicious RoyalRoad document exploiting one days (CVE-2012-0158, CVE-2017-11882, CVE-2018-0802), then USBCulprit is dropped selectively by the attacker.
Retro	Email	Spearphishing with documents that exploit CVE-2017-11882.
PlugX	Email	Spearphishing with ZIP archive containing a malicious LNK file.

	Compromise method	Details
Offline frameworks		
ProjectSauron	Not applicable (offline frameworks)	
EZCheese		
Emotional Simian		
USBCulprit		
USBThief		
Brutal Kangaroo		
Ramsay		

Table 1 // Techniques used to compromise the connected-side system

4.2 Air-gapped side initial execution vector

All frameworks have devised their own ways to reach the target air-gapped network and execute malware on a first system. They all have one thing in common, though: they all used weaponized USB drives.

The main difference between connected and offline frameworks is how the drive is weaponized in the first place. Connected frameworks usually deploy a component on the connected system that will monitor the insertion of new USB drives and automatically place the malicious component needed to compromise the air-gapped system.

Offline frameworks, on the other hand, rely on the attacker intentionally weaponizing their own USB drive.

What is interesting here is the variety of techniques used over time by these frameworks to get their payload executed on the target system. We can place these into three large categories.

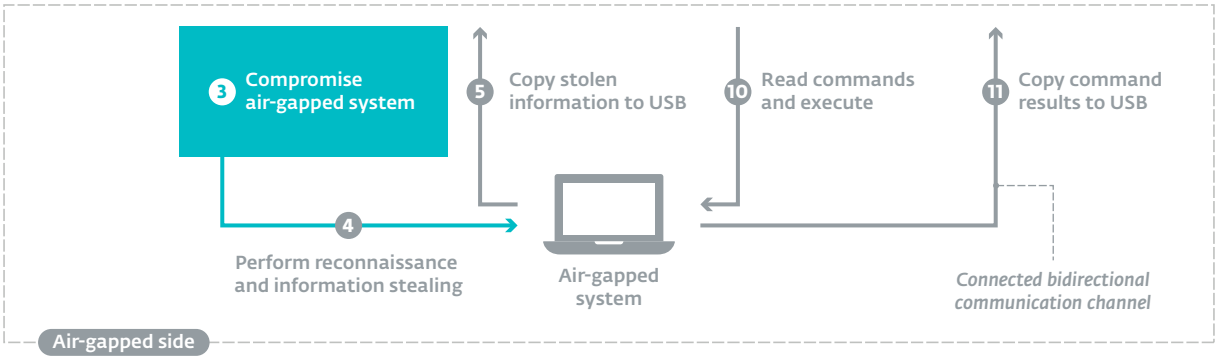

Figure 8 // Air-gapped side initial system compromise. Part of Figure 5.

Automated execution

Getting malicious code executed just by connecting a malicious USB drive into a computer is the most effective technique to compromise an air-gapped system.

Exploitation of LNK-related vulnerabilities

Malicious LNK files are usually used as the exploit to trigger a vulnerability in old components of Windows, such as the Windows Shell, that allow the malware to get remote code execution with no user action required other than viewing the LNK file in Windows Explorer.

The most famous vulnerability is without a doubt CVE-2010-2568, aka the "Stuxnet LNK exploit". But Stuxnet is far from the only framework to have used that vulnerability to gain initial access to air-gapped networks. In fact, it was later discovered that Fanny had used that exploit even before Stuxnet [5]. And even after Microsoft released a patch in 2010 for CVE-2010-2568, Flame, Gauss and miniFlame continued to use it afterwards.

The Vault7 leaks also revealed that the Brutal Kangaroo tool suite used two LNK-related exploits: CVE-2015-0096 and a second unidentified one.

LNK-related vulnerabilities are an extremely powerful way to spread malware because they have the potential to allow code to execute without user interaction upon USB drive insertion in vulnerable systems. Table 2 shows that while only two such vulnerabilities have been confirmed as exploited in the wild, they are not so uncommon: almost a dozen have been discovered and patched by Microsoft in the last 10 years, 4 just in 2020.

Advisory date	CVE	Vulnerability name	Exploited in the wild
2010-08-02	*CVE-2010-2568*	Shortcut Icon Loading Vulnerability (remote code execution)	Fanny, Stuxnet, Flame, Gauss, miniFlame
2015-03-10	*CVE-2015-0096*	DLL Planting Remote Code Execution Vulnerability	Brutal Kangaroo (suspected)
2017-06-13	*CVE-2017-8464*	LNK Remote Code Execution Vulnerability	`Blacksquid`, `Lucifer`, `etc.`
2018-08-14	*CVE-2018-8345*	LNK Remote Code Execution Vulnerability	
2018-08-14	*CVE-2018-8346*	LNK Remote Code Execution Vulnerability	
2019-08-13	*CVE-2019-1188*	LNK Remote Code Execution Vulnerability	
2019-09-10	*CVE-2019-1280*	LNK Remote Code Execution Vulnerability	
2020-03-10	*CVE-2020-0684*	LNK Remote Code Execution Vulnerability	
2020-02-11	*CVE-2020-0729*	LNK Remote Code Execution Vulnerability	
2020-06-09	*CVE-2020-1299*	LNK Remote Code Execution Vulnerability	
2020-07-14	*CVE-2020-1421*	LNK Remote Code Execution Vulnerability	

Table 2 // History of RCE vulnerabilities related to LNK files

Overview of the "Stuxnet LNK exploit"

There is a common misconception that the vulnerability is in the LNK (Windows Shell Link aka "LNK") file format itself but it's really not the case. Stuxnet's LNK files are not malformed or crafted in a way that make the Windows Shell parse the structure incorrectly and as a result execute shellcode (as is common with other vulnerabilities, for example, the more recent case of CVE-2020-0684). Rather, it took advantage of an old feature of the Control Panel and a couple of design flaws that allowed it to load a DLL into the Windows Explorer process.

When Windows Explorer browses a folder containing LNK files, it opens them to read the metadata that specifies where it should get the icon to display for the linked file. In the case of Stuxnet, it specifies that the location of the target is the Control Panel and the icon should be loaded from a CPL file. Windows CPL files are DLL files with a different file extension and are expected to have an export called `CPlApplet`, which is the entry point for the Control Panel Applet.

Figure 9 shows the contents of Stuxnet's LNK file, with highlighted metadata that specifies the location of the CPL file from which Explorer should obtain the icon to display.

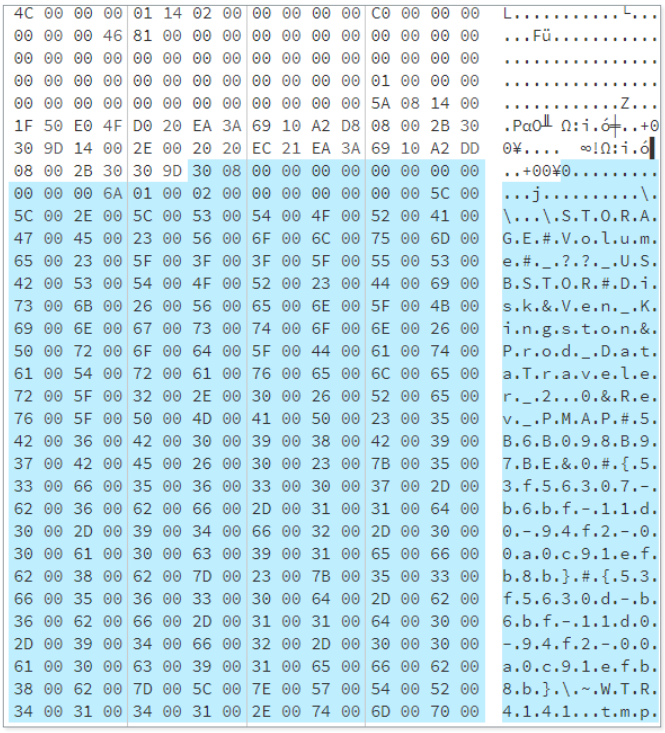

Figure 9 // Stuxnet's malicious LNK file specifies a file with a .tmp extension as its icon source.

The code in the Windows shell that handled CPL files was flawed in the way that it dynamically loaded the file using `LoadLibraryW`—which causes the execution of code—and then calls the `CPlApplet` export to initialize and request the needed item, and yet another problem is that the location of the CPL file was not verified via the Windows registry at `Software\Microsoft\Windows\CurrentVersion\Control Panel\Cpls` nor were the entries in the `MMCPL` section of the `Control.ini` file for registered CPLs. This is how Stuxnet's hidden DLL, `~WTR4141.TMP` in the root of the USB drive, was executed automatically when the potential victim browsed the drive.

Non-automated execution (unknowingly triggered)

In these scenarios, the execution of the malicious code depends on tricking an unsuspecting legitimate user into executing the malicious code.

Use of the AutoRun/AutoPlay Windows feature

This is an old (and very successful) execution vector that allowed several malware families to spread via USB and network drives. While AutoRun has existed since Windows 95, the terms AutoRun and AutoPlay had been used interchangeably until Windows XP when the two were differentiated:

1. AutoRun was introduced in Windows 95 with the purpose of making it easier to trigger the automatic execution of installers on CDs by following the instructions in a file named `autorun.inf`. This mechanism was also used when the user double-clicked on the drive's shortcut in My Computer. It was later enabled and extended in Windows XP to support other types of removable media such as USB drives and network drives by checking enabled types in the Windows registry.
2. Although a primitive form of AutoPlay existed back in Windows 95 and 98, it was revamped in a new way to present the user with a pop-up window with a menu of options for easy access and automatic launching of files when a device was inserted.

These are the features abused by frameworks such as USBStealer and Agent.BTZ, as well as an earlier Stuxnet version that implemented a Flame component that weaponized USB drives with a malicious `autorun.inf` file (as seen in Figure 10) that contained both the Stuxnet executable and the AutoRun instructions. It disabled AutoPlay to force the user to go to My Computer or use the entry in the navigation tree of Windows Explorer; with `shell32.dll` it adds an additional command "Open" to the context menu that will execute Stuxnet if the potential victim clicks on it or double-clicks on the drive shortcut.

Figure 10 // Part of Stuxnet's autorun.inf file, *https://docs.broadcom.com/doc/security-response-w32-stuxnet-dossier-11-en*

Figure 11 shows how USBStealer uses more traditional instructions to achieve a similar effect but without disabling AutoPlay:

```
[autorun]
open=
shell\open=Explore
shell\open\command="System Volume Information\USBGuard.exe" install
shell\open\Default=1
```

Figure 11 // USBStealer autorun.inf file contents.

In 2009 Microsoft released an update to disable the AutoRun functionality for media other than CDs and DVDs. Windows 7/8/8.1/10 AutoRun does not support the majority of instructions that allowed malicious code to be executed through `autorun.inf` files, leading also to no more custom options in the AutoPlay window. Given these restrictions, some frameworks such as Stuxnet, Flame (and Gauss and miniFlame) switched to the use of exploits.

Planting of malicious files

Aside from AutoRun/Autoplay, an attacker can plant a malicious executable or LNK file on the USB drive in a way to lure the user to execute them. Ramsay, for example, used a trojanized installer for 7zip, while USBThief used DLL hijacking against the legitimate portable applications Firefox, Notepad++ and TrueCrypt. It is worth mentioning that none of these frameworks have functionality or specific techniques to trojanize or hijack specific applications (as in the case of USBThief for example); in these particular cases all samples were found with the technique already applied.

USBThief's technique may also hide the malicious activity from security monitoring tools since the parent process executing the malicious DLL will be coming from a known executable.

Mustang Panda's PlugX operates a bit differently, but require action to be taken by the user. The malware would store a copy of itself on USB drives under the `RECYCLE.BIN` folder, hide all existing folders in the drive's root and then create one LNK file for each hidden folder. Each LNK file would of course point to the malicious executable. This tactic would preserve the appearance of the clean drive and the malicious code would be executed as soon as an unsuspecting user would try to enter one of the directories.

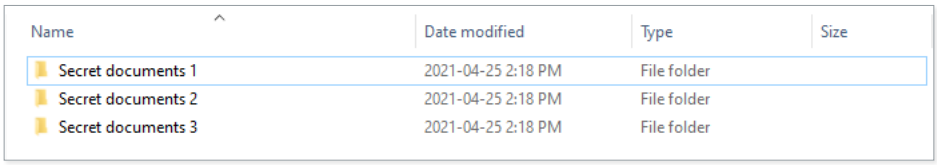

Figure 12 // Example content of a USB drive prior to PlugX's weaponization

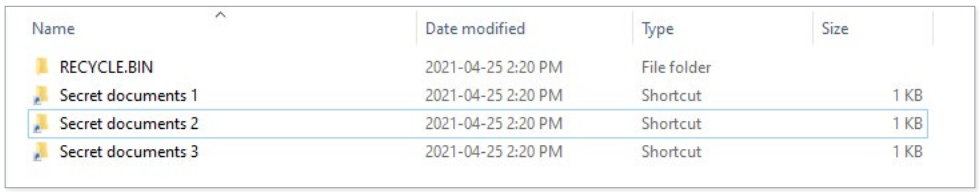

Figure 13 // Contents of the USB drive after PlugX's weaponization

Lastly, attackers can also put malicious Office documents on USB drives. Ramsay for example used custom-created malicious RTF documents exploiting two known vulnerabilities to install itself.

Retro used a slightly different approach, scanning USB drives looking for existing Word documents and then converting them to RTF format with an exploit that triggers a script to copy the reconnaissance component to the local drive and execute it. This technique is less likely to raise suspicion since the presence of documents on the USB drive would be expected.

Non-automated execution (deliberately performed)

This case is similar to the previous technique; however, analysis indicates that the attackers did not intend for an unsuspecting user of an air-gapped system to trigger the execution of the malware. Instead, they relied on a human actor to deliberately launch the malware on the target systems.

The case of USBCulprit is a prime example of that, the malware is not directly accessible or visible to the potential victim, and there is no trigger mechanism, preventing accidental execution: from a compromised machine, USBCulprit copies itself to a USB drive when a particular hidden file is found in a directory called `$Recycle.Bin` created as a staging area for exfiltration of data.

In other cases such as USBThief, the malware is deployed in a manner that support covert execution by a human actor: the attackers place in USB drives, portable versions of well-known software and use techniques such as DLL hijacking or DLL side-loading to indirectly execute the malware in the target system.

	Automated execution		**Non-automated execution (unknowingly triggered by the target)**				**Non-automated execution (deliberately performed)**
	Malicious LNK	Malicious `autorun.inf` triggering automatic execution (AutoRun)	Trojanized/ hijacked PE	Malicious Office documents	Malicious LNK	Malicious `autorun.inf` manipulating AutoPlay	
Connected frameworks							
USBStealer						X	
Agent.BTZ						X	
Stuxnet	X					X	
Fanny	X						
miniFlame	X						
Flame	X					X	

	Automated execution		Non-automated execution (unknowingly triggered by the target)				Non-automated execution (deliberately performed)
	Malicious LNK	Malicious `autorun.inf` triggering automatic execution (AutoRun)	Trojanized/ hijacked PE	Malicious Office documents	Malicious LNK	Malicious `autorun.inf` manipulating AutoPlay	
Gauss	X						
USBFerry						X	
USBCulprit							X
Retro				X			
PlugX					X		
Offline frameworks							
ProjectSauron							X *Hypothesis*
EZCheese	X						X
Emotional Simian	X						
USBThief			X				X
Brutal Kangaroo	X	X					X
Ramsay			X	X			X

Table 3 // Techniques used to compromise the first air-gapped system

4.3 Air-gapped side functionalities

Persistence

When it comes to persistence, these frameworks can be divided in two groups:

1. **Persistent:** It's designed to install and persist in new systems and continue collecting data for later exfiltration. Some of these frameworks present the capabilities to receive commands via covert communication channels and install or execute new components in the compromised system beyond existing execution vectors in already compromised USB drives.
2. **Non-persistent:** Designed for in-memory reconnaissance; collects information that is staged on the USB drive for exfiltration.

There are no known special techniques designed for persistence in air-gapped machines. Once the malware is launched through its execution vector, it only depends on the capabilities of the malware that enable it to access powerful persistence methods. But from our research we can see that most of the frameworks just employ persistence based in the Windows registry to get executed at system startup or loaded when a process is created via hijacking.

Notable exceptions are Stuxnet and Fanny, that used zero-day exploits to elevate their privileges and persist by installing drivers; so too is the case of Flame, which registers itself as a Security Support Provider (SSP) authentication package and gets loaded into critical system processes the next time the system boots.

Reconnaissance and espionage activities

Once an attacker gains access to a new air-gapped system, reconnaissance and espionage activities can be started.

As we discussed earlier, attackers usually do not have direct control over which specific air-gapped system will become compromised, as this largely depends on which system an unsuspecting user will connect the malicious USB drive to. To address this and discover on what kind of system they are running, most frameworks perform some type of reconnaissance activities.

Host-specific information gathering related to the system itself.

Frameworks gather information such as computer name, username, domain name, list of running processes, listing of files in directories, drives and network shares, as well as network configuration information that can help the attacker plan for lateral movement. Some frameworks, for example USBFerry, USBCulprit, Retro and PlugX, simply pipe the output of built-in Windows commands (for instance, `ipconfig` or `netstat`) to a file that is exfiltrated to the attacker (see *Detection opportunity #1*: air-gapped side host reconnaissance activity); other frameworks such as Flame, Stuxnet, and Gauss, perform their reconnaissance directly using the Windows API, which could be considered stealthier than the former's approach. In this sense, Gauss went even one step further with its strategy:

- Initially, the Gauss reconnaissance component, which is only deployed from USB drives, collected system information that, once exfiltrated to the attackers, could be used to prepare a special payload encrypted with a key derived from a mix of system information that would be hard to match with any other system in the world. Think of it as a key derived from a fingerprint.
- Once they have it prepared, they added it to the reconnaissance module and commanded Gauss on the connected side to download it in the connected side machine. And the propagation process would start again until it reached the system where it could be decrypted.

This strategy allowed the attackers to be extremely selective of which system in an air-gapped network they would actually compromise and to have a payload that could only be decrypted by the unique targeted system. Researchers have tried millions of combinations to try to decrypt the payload and even have released to the public all the technical details needed to attempt breaking the encryption. So far, all attempts have failed [6].

Active network reconnaissance/probing

While less common, some frameworks also have the capability to explore the network they are inside of with active measures—ones that generate network traffic. For example, USBFerry and USBCulprit used built-in Windows commands such as `tracert` and `ping` to determine whether the system they are running on has internet connectivity or not.

Ramsay used a much more aggressive approach by automatically scanning all IP addresses in the local subnet and testing the discovered hosts to determine whether they are vulnerable to *CVE-2017-0144*, also known as EternalBlue. Note that we have not seen any Ramsay component capable of exploiting the vulnerability.

More details about these reconnaissance activities and how to detect them are described in the *Defending air-gapped networks section*.

Espionage activity

In several cases, the specific details of the framework's espionage capabilities were fairly simple and straightforward for researchers to determine. USBStealer, USBCulprit, Retro, USBThief, PlugX and Ramsay, for example, were clearly built to steal specific files. The flexibility of this mechanism varied from one framework to another; some had a static list of file extensions hardcoded in the malware binary, while others had these dynamically provided via configuration files or other method.

Only Retro has the ability to take screenshots at regular intervals.

Flame arguably had the most extensive information-stealing capabilities, being able to steal whole documents or metadata, images and GPS coordinates, technical CAD drawings and more. Symantec has the best description for Flame's data collection capabilities: "Even describing it as an industrial vacuum cleaner does not do it justice." [7]

In other cases such as Gauss, details about espionage capabilities on the air-gapped side are uncertain or just simply unknown.

Propagation to other hosts on the air-gapped side

We previously covered the various mechanisms used by the attackers to compromise an initial air-gapped system. Some frameworks also include spreading capabilities built right into the malware that's running on the air-gapped side to perform lateral movement and thus gain deeper access into the air-gapped network.

This aspect also varies widely from one framework to another. While older frameworks had ways to spread automatically using USB or network-based exploits, most recent ones rely on a human action to compromise additional systems.

Surprisingly, a significant proportion of the frameworks have no spreading mechanism at all. Those that do use propagation techniques either based on USB drives or network connectivity inside the air-gapped network are sometimes, but not always, the same techniques as those used to perform the initial air-gapped system compromise.

USB drive-based propagation

As far as using USB drives to spread, the frameworks employed one of the five broad sub-techniques, where the malicious code would be copied to the drive along with some way for it to be launched.

Stuxnet, Fanny and Retro used exactly their respective techniques to spread inside the air-gapped network as the one they used to penetrate the network in the first place.

Ramsay used something relatively similar to Retro, but instead of weaponizing existing Word documents, Ramsay's spreader component would trojanize all executable files found on all newly inserted USB drives, waiting for a user to launch one of them.

Agent.BTZ and USBStealer both used a trick more often used for the initial air-gapped system compromise, which is to place the malware on the USB drive alongside an `autorun.inf`.

As for USBCulprit, it is one very strange case. USBCulprit has the same code running on both the connected side and the air-gapped side, and so has the same ability to weaponize USB drives on both sides. As discussed previously, that mechanism consists of copying itself into a hidden folder on newly inserted USB drives, without adding any mechanism to launch itself. We would thus assume that an insider would have to execute the file on purpose on other systems in order to spread the malware further, making it a partially automated propagation mechanism.

Network-based propagation

Very few frameworks have network-based propagation on the air-gapped side. Stuxnet was well known for its use of CVE-2010-2729, the Windows Print Spooler RCE exploit. Flame used it as well and a unique method, never used by any other known malware, which was to perform a man-in-the-middle to intercept Windows Update traffic coming from air-gapped systems to spread malicious update components signed with a forged but valid code-signing certificate.

Ramsay's approach is far more primitive, as it trojanizes PE files on USB drives and network drives. We could still imagine this technique being extremely effective when air-gapped systems have a mapped drive pointing to a central file server containing the installation files for all the software allowed to be deployed within the air-gapped environment.

	No built-in propagation mechanism	USB drive-based propagation					Network-based propagation
		Payload + LNK exploit	Payload + autorun.inf	Payload only (*no execution mechanism*)	Trojanized Word documents	Trojanized PE files	
Connected frameworks							
USBStealer							
Agent.BTZ			X				
Stuxnet		X	X				X CVE-2010-2729 (Print Spooler RCE)
Fanny		X					
Flame							X Windows Update hijacking technique CVE-2010-2729 (Print Spooler RCE)
miniFlame		X					
Gauss	X						
USBFerry	X						
USBCulprit				X			
Retro spreader plugin					X		
Mustang Panda's custom PlugX	X						
Offline frameworks							
ProjectSauron	X						
EZCheeze	X						
Emotional Simian	X						
USBThief	X						
Brutal Kangaroo	X						
Ramsay						X	X (infection of PE files found on network shares)

Table 4 // Techniques used to spread within the air-gapped networks

4.4 Communication and exfiltration channel

At the beginning of this white paper, we clarified our definition of "air-gapped malware" and split air-gapped frameworks in two categories: connected and offline. The difference from the communication and exfiltration point of view is significant: online frameworks require an online, traditional C&C communication channel connecting the attacker to the connected-side compromised host, and an offline one connecting the connected-side compromised host and the air-gapped systems, as shown in Figure 14. On the other hand, Figure 15 shows how offline frameworks only require the latter.

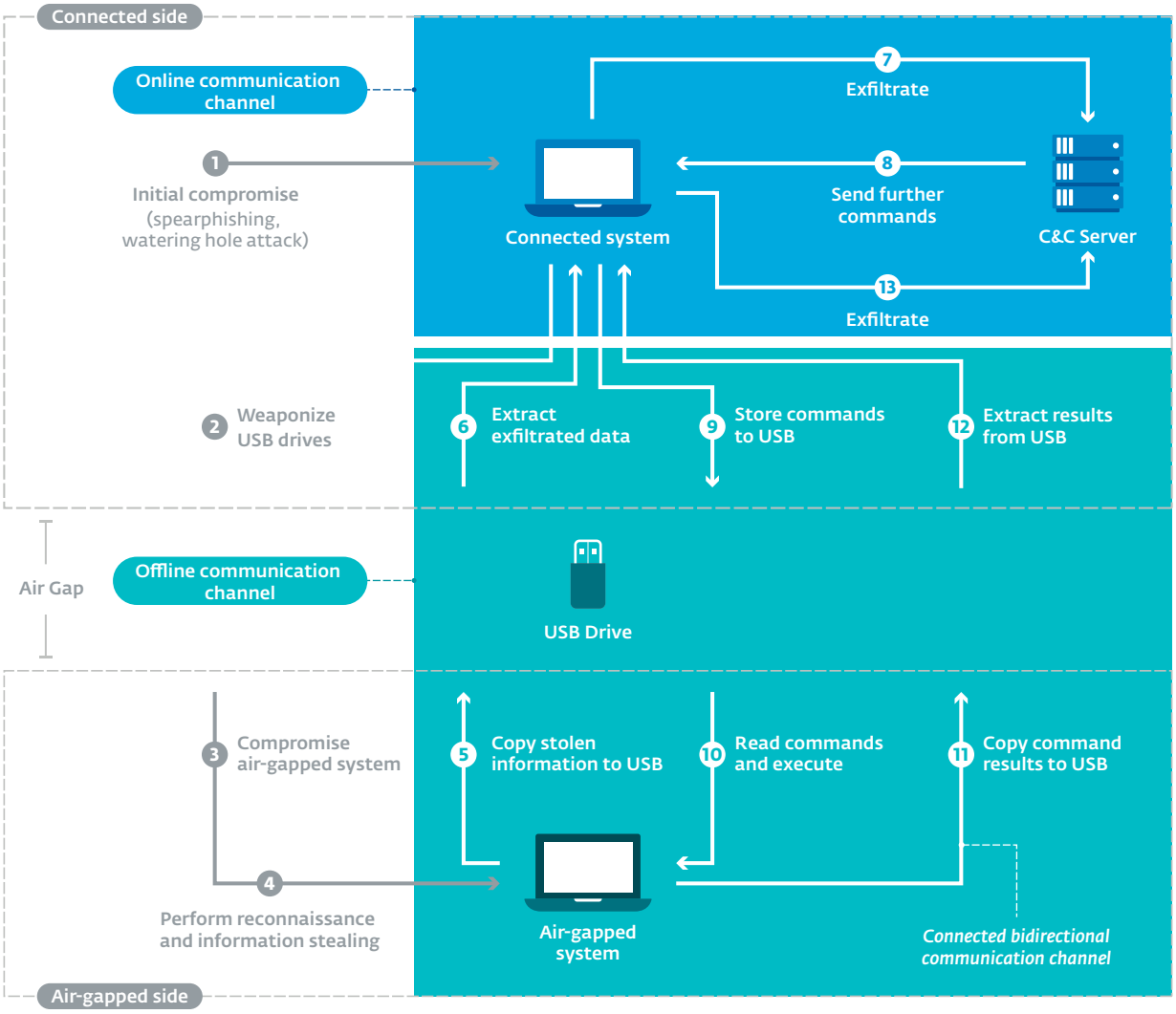

Figure 14 // Online and offline communication channels in connected frameworks

In this section, we explore how both online and offline communication channels are implemented in known frameworks.

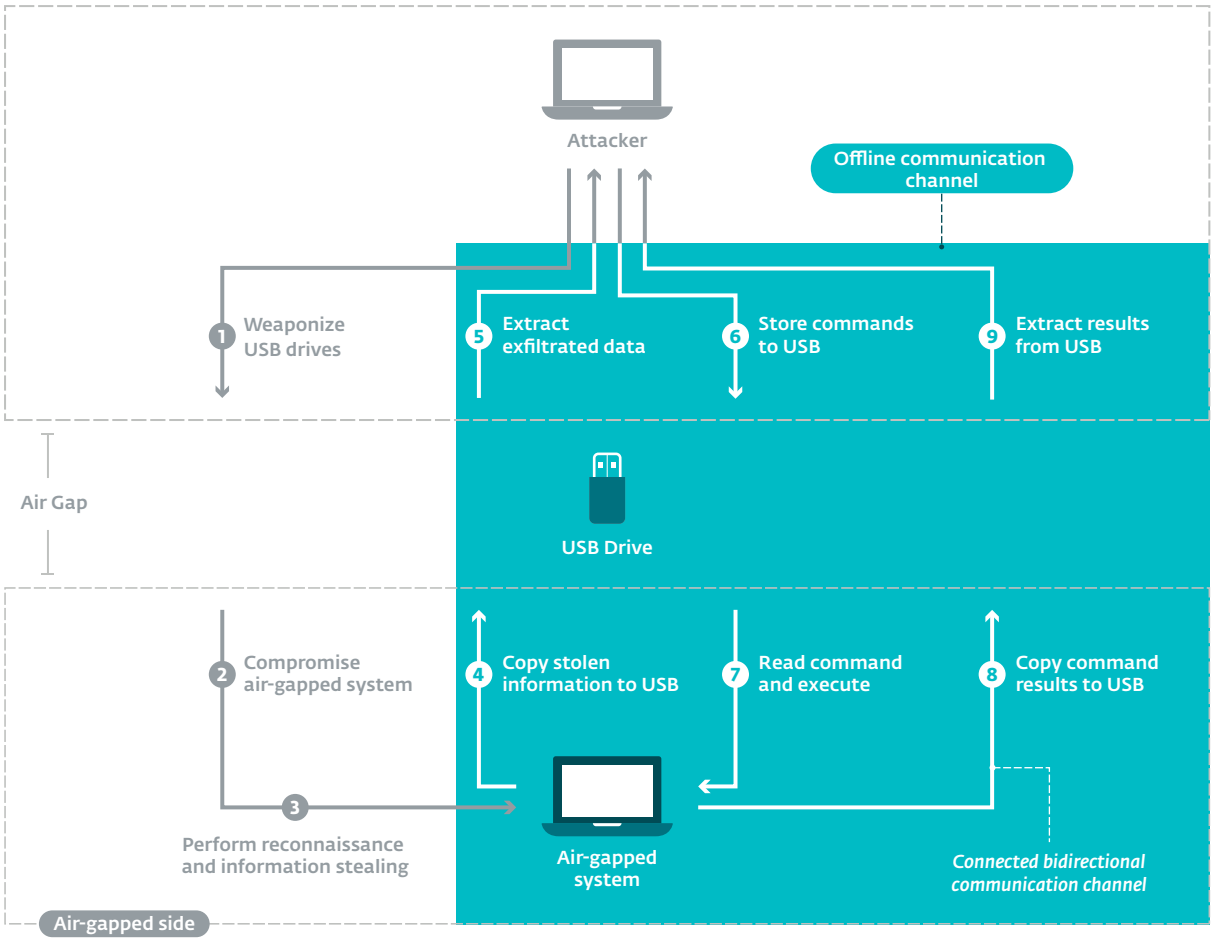

Figure 15 // Offline communication channel in offline frameworks

Online channels

Akin to the techniques described in the Connected side section, the traditional online communication channels in the connected frameworks we studied have no characteristics specific to attacking air-gapped networks. Techniques vary from one framework to another, just like when studying regular malware; thus we won't focus too much on that part.

In general, we found that the level of sophistication and complexity of the online communication channel were relatively on par with the overall sophistication level of the other parts of the frameworks themselves.

Offline channels

The presence of an offline communication channel is the core part of our definition of what air-gapped malware is. This is how the malware bypasses the air gap defense layer to transfer information in and out, or sometimes just out, of the target network.

An offline channel can be seen as a specific communication protocol running over a certain physical transmission medium across the air gap.

One of the first things that comes to mind when talking about attacks against air-gapped networks is how the air gap can be bypassed. In fact, new research on covert physical transmission mediums is published on a regular basis. One of the most prolific researchers in that domain is certainly Mordechai Guri, lead cybersecurity researcher at Ben-Gurion University of the Negev. He and his team have demonstrated the feasibility of numerous techniques [8] [9] that allow information transfer across airgaps with various levels of attack deployment complexity and available bandwidth.

While there have been alleged sightings of in-the-wild attacks using such techniques[1], no peer-reviewed case has been publicly analyzed and disclosed. Practically all the malicious frameworks targeting air-gapped networks publicly known to date used USB drives as the physical transmission medium to transfer information across air gaps.

Looking at the communication protocols developed by the various attackers, we can see a much broader diversity of techniques.

Table 5 illustrates how about half of the frameworks only implement unidirectional protocols. In these attack scenarios, the information can only flow from the compromised air-gapped system to the attacker, and not the other way around. This means the malware component running on the air-gapped side does not have any update mechanism or backdoor capabilities: the attacker simply has no way at all to send updates or commands to control the compromised system.

Instead, the malware components running on the air-gapped side are designed to perform specific, hardcoded tasks, usually reconnaissance and information stealing, and then exfiltrate the information back to the attacker via the USB drive.

Frameworks implementing bidirectional protocols are more flexible, as they allow much better control over the compromised air-gapped hosts. Interestingly, not all frameworks make full use of this capability. In fact, most implement only a small, not very flexible set of commands, such as steal files matching specific patterns or run a specific executable file present on the USB drive.

Probably the most noteworthy exception is Ramsay, which has one of the most rich and flexible offline communication protocol. For one thing, commands can be issued to air-gapped compromised systems by planting a specially crafted control file either on a USB drive or on a network share that's accessible to the target system. These files also contain a GUID, which allows the attacker to either restrict the commands to a single, specific target, or to be "target agnostic" and recognizable by any system compromised by Ramsay. This feature makes the most sense when the command file is distributed via a network share. The protocol consists of only three commands, which is enough to be able to run anything on the host: execute a file, load a dll, and run a batch file.

1 *https://en.wikipedia.org/wiki/BadBIOS*

	Exfiltration only (unidirectional)	Command and response (bidirectional)
Connected framework		
USBStealer		X
Agent.BTZ	X	
Stuxnet		X
Fanny		X
miniFlame	X	
Flame		X
Gauss	X	
USBFerry	X	
USBCulprit		X
Retro	X	
PlugX	X	
Offline framework		
ProjectSauron	X	
EZCheese	X	
Emotional Simian		X
USB Thief	X	
Brutal Kangaroo		X
Ramsay		X

Table 5 // Types of offline communication protocols

Another thing to look at when studying offline channels is the way the information is encoded on the USB drives. Once again, we see a wide variety of techniques, such as:

- Files marked with hidden or system attribute
- Information placed in hidden storage space, such as hidden virtual filesystems or at specific raw offsets
- Information compressed in password-protected RAR archives
- Information encrypted and appended to existing Word documents
- Files stored in NTFS Alternate Data Stream (ADS)

In all cases, the idea is the same: prevent data leak detection and access to stolen information by a fourth party.

5. DEFENDING AIR-GAPPED NETWORKS

It goes without saying that defending air-gapped networks against cyberattacks is a very complex topic that involves several disciplines. It is far from our intention to claim that we have a magical solution to this problem. That being said, there is value in understanding how known frameworks operate in air-gapped environments and deriving ways to detect and block common malicious activities.

This section presents ideas to detect and block malicious activities that are common to a significant portion of the studied frameworks. None of them are revolutionary, but we hope that our data-driven approach will help defenders prioritize their defense mechanisms. In other words, that defenders first implement defense mechanisms against what known malware has been doing so far, before trying to block techniques that have not yet been used yet.

For example, defenders could decide to first properly defend against malicious LNK/autorun files—which 70% of the frameworks have used either to compromise patient zero or to spread further within the air-gapped network—before implementing TEMPEST[2] countermeasures to shield their network against computer emanations such as electromagnetic, sound or mechanical vibrations that, while absolutely possible, have not been seen used in any framework so far.

5.1 Protection opportunity #1: prevent email access on connected hosts

In the connected side system compromise section, we have seen how several frameworks used connected side systems that are used in conjunction with air-gapped networks as the initial point of entry into targeted air-gapped networks. Naturally, those systems should be treated as such and hardened with strict security measures.

Not surprisingly, email is the most frequent technique used by attackers. Preventing direct access to emails on connected systems would mitigate this execution vector. This could be implemented with browser/email isolation architecture, where all email activity is performed in a separate, isolated virtual environment.

5.2 Protection opportunity #2: disable USB ports on air-gapped systems

Physically removing or disabling USB ports on all the systems running in an air-gapped network is the ultimate protection that would have prevented all the 17 frameworks to succeed. However, doing so comes with an important usability drawback.

While removing USB ports from all systems may not be acceptable for all organizations, it might still be possible to limit functional USB ports only to the systems that absolutely require it.

5.3 Protection opportunity #3: sanitize USB drives before insertion in air-gapped systems

We've seen to what extent frameworks abuse USB drives to execute malicious code on air-gapped systems. A USB drive sanitization process performed before any USB drive gets inserted into air-gapped system could disrupt many of the techniques implemented by the studied frameworks.

Imagine a quarantine system into which users of the air-gapped network would connect any USB drive before insertion into the air-gapped system and that would automatically sanitize the drive. This would not prevent deliberate human action, but would still cover a significant portion of the attack scenarios.

Remove all LNK and autorun.inf files from USB drives

Table 6 shows which frameworks use USB drives loaded with either a malicious LNK or autorun file to compromise isolated systems or to propagate further. In total, 13 out of 17 frameworks could have been thwarted by the simple removal of those two file types from USB drives before they got inserted in any air-gapped system.

2 Telecommunications Electronics Materials Protected from Emanating Spurious Transmissions (TEMPEST) https://en.wikipedia.org/wiki/Tempest_(codename)

Framework	Malicious LNK	Malicious `autorun.inf`
Connected framework		
USBStealer		X
Agent.BTZ		X
Stuxnet	X	X
Fanny	X	
miniFlame	X	
Flame	X	X
Gauss	X	
USBFerry		X
USBCulprit		
Retro		
PlugX	X	
Offline framework		
ProjectSauron		
EZCheese	X	
Emotional Simian	X	
USBThief		
Brutal Kangaroo	X	X
Ramsay		

Table 6 // Frameworks using malicious LNK and autorun.inf files

In fact, the history of critical RCE vulnerabilities related to LNK files that have been patched by Microsoft in the last 10 years, as shown in *table 2*, speaks for itself: if LNK files on USB drives are not absolutely needed in an air-gapped network, they should be removed before they reach sensitive systems.

Perform a malware scan

Even with LNK and `autorun.inf` files gone, some frameworks would still pose a threat in a different way. Retro, for example, weaponized USB drives with malicious Office documents containing n-day exploits, Ramsay relies on trojanized executable files. Systematically scanning, on the connected side, all USB drives with an up-to-date antimalware product would provide a baseline layer of protection before the files get to the air-gapped side.

5.4 Protection opportunity #4: restrict file execution on removable drives

Several techniques used to compromise air-gapped systems end up with the straight execution of an executable file stored somewhere on the disk, which could be prevented by configuring the relevant Removable Storage Access policies as shown on Figure 16 on all air-gapped systems.

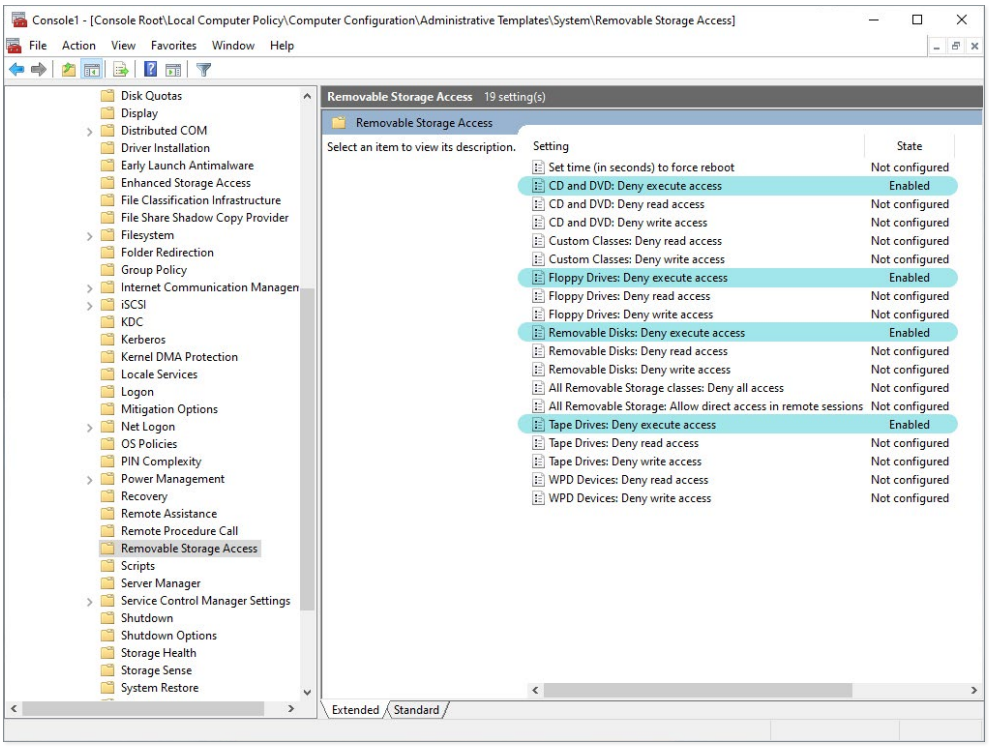

Figure 16 // Window policies to control Removable Storage Access

Implementing application whitelisting would also prevent Ramsay's propagation mechanism that infected executable files found on network drives.

5.5 Protection opportunity #5: maintain air-gapped systems updated

Several frameworks have used exploits against Windows or Microsoft Office either to perform the initial compromise of an air-gapped system or to spread further in the isolated network. Table 7 shows a summary of all the exploits seen used on the air-gapped side. It also shows how often n-day exploits were used.

Had the target systems been fully patched, the attacker's task to succeed with their attacks would have been more difficult, forcing them either to develop or acquire suitable zero-day exploits or to use alternate, less efficient techniques.

It is also worth noting that endpoint security products are generally able to detect and block several exploit classes (network- and file-based, for example), so having such technology not only deployed but also kept up to date could have had a positive impact.

It is frequently reported that air-gapped systems are outdated because, by definition, they are air-gapped and thus can't reach update servers. If that really is the case, could it be possible that maintaining a full air gap causes more security issues and actually could be harmful? We do not have an answer to that complex topic, but seeing how many n-day exploits are used against air-gapped networks, it is a possibility that should be considered by air-gapped network administrators.

For example, an air gap could be partially broken to allow dual-homed update servers such as Windows Server Update Services (WSUS) or an endpoint security update mirror server, to maintain all other air-gapped systems fully updated.

Alternatively, it is possible to deploy air-gapped update servers, although manual operations are required to regularly import the update data into the air-gapped side. For example, Microsoft supports a dual WSUS setup to service a disconnected network[3], in which the administrators are required to manually transfer the new updates from the connected WSUS instance to the air-gapped one via a removable drive.

That being said, we have to keep in mind that Flame actually managed to spread via Windows Update [10]. However, no other similar cases have been reported so far.

	Zero day	One day
Connected framework		
USBStealer		
Agent.BTZ		
Stuxnet	CVE-2010-2568 (LNK RCE) CVE-2010-2729 (Print Spooler RCE) CVE-2010-2743 (Keyboard layout EoP) CVE-2010-3338 (Task Scheduler EoP) CVE-2010-2772 (hardcoded password in Siemens Simatic WinCC)	CVE-2006-3439 (RPC RCE) CVE-2008-4250 (RPC RCE)
Fanny	CVE-2010-2568 (LNK RCE) MS09-025 (EoP)	
miniFlame		CVE-2010-2568 (LNK RCE)
Flame		CVE-2010-2568 (LNK RCE) CVE-2010-2729 (Print Spooler RCE)
Gauss		CVE-2010-2568 (LNK RCE)
USBFerry		
USBCulprit		
Retro		CVE-2017-8570 (Microsoft Office RCE)
PlugX		
Offline framework		
ProjectSauron		
EZCheese	Use of an unspecified LNK exploit. It is not known whether the vulnerability was exploited before or after a patch was released.	
Emotional Simian		
USBThief		
Brutal Kangaroo	CVE-2015-0096 (LNK RCE) + a second unidentified exploit	
Ramsay		CVE-2017-0199 (Microsoft Office/ WordPad RCE) CVE-2017-11882 (Microsoft Office RCE)

Table 7 // Use of exploits inside the air-gapped sides (usage on the connected side excluded)

3 https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd939873(v=ws.10)

5.6 Detection opportunity #1: air-gapped side host reconnaissance activity

Some frameworks use Windows commands to perform their reconnaissance activities. This activity can be caught easily by various methods. For example, systems can be configured to generate audit events when processes are created[4] and their logs monitored[5] to detect the execution of specific processes.

For example, Table 8 could be used as a base to build a reconnaissance-specific Sigma rule[6]. The detection of such activity inside an air-gapped network would be highly suspicious.

Reconnaissance activity performed by Flame, Stuxnet or Gauss, however, could not be detected this way.

Command	USBFerry	USBCulprit	Retro plug-in	PlugX
tasklist	(no arg)	/v	/v	
systeminfo			(no arg)	(no arg)
hostname	(no arg)			
ipconfig	/all	/all	/all	/all
arp	-a	-a	-a	-a
netstat	-ano	-aon	-ano	-ano
nbtstat	-n			
route	Print		print	
sc			query wlansvc	
netsh wlan show			qll, profiles, interface, net-works mode=Bssid	

Table 8 // Commands and parameters issued to perform reconnaissance on air-gapped systems

5.7 Detection opportunity #2: air-gapped side network-based reconnaissance activity

Only a minority of frameworks have active network reconnaissance activity that would generate actual network traffic.

Such activity generated by USBFerry and USBCulprit could be detected on the compromised host directly using the same technique described in the previous section.

Command	USBFerry	USBCulprit
ping	-n 1 <various hosts>	google.com
tracert	-h 8 <some local addresses>	
net	view, view /domain, use, user, user administrator, share, localgroup administrators	view

Table 9 // Active network reconnaissance activity via Windows commands

Detecting the presence of Ramsay mass scanning of the entire local subnet for hosts vulnerable to Eternal-Blue could be done in several ways. For example, a properly updated host-based IDS on all hosts would raise the relevant alerts and point directly to the host running Ramsay.

4 https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/audit-process-creation

5 https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4688#security-monitoring-recommendations

6 https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/win_susp_commands_recon_activity.yml

6. CONCLUSION

We have seen how the frameworks can be split into two categories: connected frameworks, which are operated fully remotely, and offline frameworks, which rely on a human asset on the ground. Despite the use of various techniques to breach the initial air-gapped system, to propagate inside the network or to exfiltrate stolen information, all the frameworks share one common goal: spy on their target.

Discovering and analyzing this type of framework poses unique challenges. They sometimes are composed of multiple components that all have to be analyzed together in order to have the complete picture of how the attacks are really being carried out.

Also, security vendors such as ESET rely on telemetry to discover new threats on systems where their products are running. By definition, systems running within air-gapped networks do not send such telemetry, which creates a significant blind spot that contributes to increasing the time to discovery and detection of new malware targeting air-gapped networks.

Understanding how malware attacks air-gapped networks can help identify and prioritize detection and protection mechanisms. For example, we saw how all frameworks relied on USB drives one way or the other to spy on air-gapped systems, and none of them used any other type of covert communication channels against which TEMPEST restrictions would need to be implemented[7].

We also saw how the various offline frameworks work and how they all rely one way or another on human assets on the ground to operate, which can impact an organization's threat model.

We will continue to monitor further developments in this very niche malware category to help defenders protect their networks.

For any inquiries, or to make sample submissions related to the subject, contact us at threatintel@eset com.

7 https://www.sans.org/reading-room/whitepapers/privacy/introduction-tempest-981

7. REFERENCES

1. Unknown, "Intellipedia—Air Gapped Network Threats," 24 01 2019. [Online]. Available: https://theintercept.com/document/2019/01/24/intellipedia-air-gapped-network-threats/. [Accessed 02 05 2021].
2. A. Gostev, "Agent.btz: a Source of Inspiration?," 12 03 2014. [Online]. Available: https://securelist.com/agent-btz-a-source-of-inspiration/58551/. [Accessed 02 05 2021].
3. J. Calvet, "Sednit Espionage Group Attacking Air Gapped Networks," [Online]. Available: https://www.welivesecurity.com/2014/11/11/sednit-espionage-group-attacking-air-gapped-networks/. [Accessed 26 04 2021].
4. Tencent, "Advanced threats: Ramsay malware's attack technology analysis against isolated networks," 28 05 2020. [Online]. Available: https://s.tencent.com/research/report/1000.html. [Accessed 02 05 2021].
5. Global Research and Analysis Team, "A Fanny Equation: "I am your father, Stuxnet"," Kaspersky, 17 02 2015. [Online]. Available: https://securelist.com/a-fanny-equation-i-am-your-father-stuxnet/68787/. [Accessed 01 05 2021].
6. Global Research and Analysis Team, "The Mystery of the Encrypted Gauss Payload," Kaspersky, 14 08 2012. [Online]. Available: https://securelist.com/the-mystery-of-the-encrypted-gauss-payload-5/33561/. [Accessed 02 05 2021].
7. A. L. Johnson, "W32.Flamer: Enormous Data Collection," Symantec, 06 04 2012. [Online]. Available: https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=c354d3e9-5e2d-4a08-91e9-fd85e193ef62&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments. [Accessed 26 04 2021].
8. M. Guri, "The Air-Gap Jumpers," Ben-Gurion University of the Negev, 08 08 2018. [Online]. Available: https://i.blackhat.com/us-18/Wed-August-8/us-18-Guri-AirGap.pdf. [Accessed 03 05 2021].
9. M. Guri, "Exfiltrating data from air-gapped computers via ViBrAtIoNs," *Future Generation Computer Systems*, vol. 122, no. September 2021, pp. 69-81, 2021.
10. A. Gostev, "'Gadget' in the middle: Flame malware spreading vector identified," Kaspersky, 04 06 2012. [Online]. Available: https://securelist.com/gadget-in-the-middle-flame-malware-spreading-vector-identified/33081/. [Accessed 26 04 2021].
11. Global Research and Analysis Team, "THE PROJECTSAURON APT," Kaspersky, 09 08 2016. [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07190154/The-ProjectSauron-APT_research_KL.pdf. [Accessed 26 04 2021].

8. APPENDIX 1: CONNECTED FRAMEWORKS OVERVIEW

Those are the frameworks that have been found in the wild, that match our definition of "air-gapped malware" and whose analysis has been made public. We included these IdentityKits as direct references to provide a per-framework overview of their functionalities. Consult their respective references for all the technical details.

It is important to note that except for the frameworks originally discovered and documented by ESET, the vast majority of the information found in this paper originated from third parties, who deserve all the credit for their research.

8.1 IdentityKit #1: USBStealer

Threat actor

Sednit

Period of operation

2005–2015 (discovery: 2014-11-11)

Purpose and targets

Perform espionage against governmental institutions in Eastern Europe.

References

2014-11-1, Sednit Espionage Group Attacking Air-Gapped Networks, ESET (*https://www.welivesecurity.com/2014/11/11/sednit-espionage-group-attacking-air-gapped-networks/*)

Characteristics

Connected side execution vector

- Suspected classic spearphishing.

Air-gapped side execution vector

- From connected side, USBStealer persisted as a service named "USB Disk Security" or via Run key.
- Monitored for the arrival of new USB drives, and weaponized them by copying USBStealer and an `autorun.inf` file.
- Created `desktop.in` file—to signal that the machine had internet access—in the System Volume Information directory of the USB drive.

Communication channel

- Connected side: Unknown.
- Air-gapped side:
  - USBStealer created a file named after the computer name and `.in` as file extension; the file was used to receive commands from the connected side compromised machines. Additionally, if the file `desktop.in` was found in System Volume Information directory, it recorded the Hardware ID of the USB drive and track of it.
  - Commands included: copy file to drive, copy files matching pattern, execute program, exfiltrate data from all drives, search for files and save results to USB drive, and more.

Air-gapped side basic functionalities

- Persistence: Attempted to register itself as a service named "USBGuard", but if when it failed then Windows Registry run key was used instead.

- Reconnaissance and Espionage:
  - The second time an interesting USB drive visited the compromised host, USBStealer would copy all the files from its local storage to the drive, with a particular interest in files with extensions `.skr` and `.pkr` from PGP Desktop cryptography software, and `.key`—possibly from another related type of software. USBStealer also had a list of files to search, they were different in some samples but include unknown files such as: `Win32Negah.dll`, `Ssders.dat`, `Settings.dat`, `audit.dat`, `key.in`, `key.out`, `z_box.exe` and others.
  - When the file with the commands was found, it executed them sequentially. If one of them failed, the files were copied into a local storage and attempted to copy them to the next drive that was inserted.

Propagation

- No propagation on the air-gapped side. USB drives were only weaponized on the connected side.

8.2 IdentityKit #2: Agent.BTZ

Threat actor

Unclassified

Period of operation

Began in 2007, but it spread out of control for several years.

Purpose and targets

US Military, with a long list of collateral victims because of its uncontrolled worm capabilities. Russia was the top affected country for several years.

References

- 2008-11-28, Pentagon computer networks attacked, LA Times (*https://www.latimes.com/archives/la-xpm-2008-nov-28-na-cyberattack28-story.html*)
- 2008-11-30, Agent.btz—A Threat That Hit Pentagon, ThreatExpert (*http://blog.threatexpert.com/2008/11/agentbtz-threat-that-hit-pentagon.html*)

Characteristics

Connected and air-gapped side execution vector

- Used an `autorun.inf` file with a command to execute `rundll32.exe` to load a malicious DLL from the USB drive.

Communication channel

- Connected side: Connected via HTTP to two C&C servers from where it downloaded a `.jpg` file that contained malicious code that was injected into Internet Explorer.
- Air-gapped side
  - When a new USB drive was inserted, it attempted to copy the file `%windir%\System32\1055cf76.tmp` to the drive as `thumb.db`. It is believed that the file was downloaded or created by another component downloaded from the C&C servers. When the USB drive had a file named `thumb.db`, Agent.BTZ copied it to the `%windir%\System32` directory, executed it and then deleted the original from the drive.
  - When the USB drive had a file named `thumb.db`, it was copied to the `%windir%\System32` directory with the name `mysysmgr.ocx`.
  - Then Agent.BTZ appended the existing `thumb.db` file with a package of CAB files that contained the logs and information collected from the system. If the file did not exist, it was created.

Air-gapped side basic functionalities

- Persistence: Copied itself as a DLL file to `%windir%\System32` with a name generated from local files and registered it as an In-Process Server that would get it loaded into `explorer.exe`.
- Reconnaissance: Collected system and network information in XML format in an encrypted file. Information included: paths to special system directories, user and computer name, directory listings, and network information such as IP, gateway, WINS, DHCP and DNS servers.
- Propagation: Monitored for the insertion of USB drives and when they were not already affected, it created an `autorun.inf` file and copied itself as a DLL along with the files described in the air-gapped side communication channel.

Discussion

Agent.BTZ, along with USBStealer and Fanny, is a textbook case of a framework designed for air-gapped network reconnaissance: it was capable of persisting, collecting information and logging its activities, as well as exfiltrating all this data and extracting new code to execute in compromised machines.

The difference is that Agent.BTZ had been made with an unlimited worming capability that allowed it to spread far and wide, indiscriminately. In one infamous incident it managed to access air-gapped side networks at the Department of Defense of the United States, classifying it as a significant threat to classified networks, according to leaked documents.

8.3 IdentityKit #3: Stuxnet

Threat actor

Unclassified

Period of operation

Stuxnet 0.5: 2007-2009
Stuxnet 1.0: 2009-2012

Purpose and targets

Telemetry indicates that Stuxnet patient zero was located in Iran, where most of the compromised systems where registered. It is widely believed that Stuxnet was made for a sabotage operation of Iran's nuclear program. However, Stuxnet at some point started spreading outside of its intended region, reaching several countries in other continents.

References

- Stuxnet 1.0:
  - 2010-09-30, W32.Stuxnet Dossier, Symantec (*https://docs broadcom com/doc/security-response-w32-stuxnet-dossier-11-en*)
  - 2011-01-04 Stuxnet under the microscope, ESET (*https://www welivesecurity com/wp-content/uploads/2012/11/Stuxnet_Under_the_Microscope pdf*)
- Stuxnet 0.5:
  - 2013-02-26, Stuxnet 0.5: The Missing Link, Symantec (*https://docs broadcom com/doc/stuxnet-missing-link-13-en*)

Characteristics

Connected side execution vector

- Unknown although presumably:
  - Human asset with physical access to the air-gapped network (AGN) would plug a USB drive containing Stuxnet.

  - Starting from connected side compromised machines by an unknown malware framework or exploit. There is evidence of this given the compilation times of some of the samples, which include reconnaissance data saved by Stuxnet itself in its own file: this data indicates a short period of time from compilation to initial compromise, too short to be prepared and handed to a human asset; therefore it's believed it was compiled and transferred via internet to compromised machines.
- Malicious LNK files exploiting vulnerability CVE-2010-2568.
- Malicious `autorun.inf` file that contains Stuxnet and `shell32.dll` command trick.
- Windows Print Spooler Service vulnerability MS10-061.
- Windows Server Service vulnerability MS08-067.
- Infected Siemens Simatic Step7 project files using exploit for vulnerability CVE-2012-3015. Stuxnet also had the capability to update older versions of itself in infected project files.
- Vulnerable WinCC machines with database software that contained a hardcoded password made it possible to inject Stuxnet in the database and execute it.
- Propagation to network shares using scheduled jobs and Windows Management Instrumentation.

Air-gapped side execution vector

- Same as for the connected side.

Communication channel

- Connected side: Connected via HTTP to two C&C servers where it would get a new configuration to update in its main DLL, and a new binary to load in its own process or a remote process. The same channel was used by Stuxnet to exfiltrate collected information.
- Air-gapped side: Stuxnet had no covert communication channel based on USB drive; however, information collected from the system was appended to the configuration block of its main DLL, and when it reached a machine on the connected side it would upload it to its C&C servers.
- Stuxnet was able to communicate with other instances running on the same network in one of two ways:
  - Windows Mailslots (in version 0.5).
  - Windows Remote Procedure Call (in version 1.0).
- Via these two mechanisms, Stuxnet was able to update itself, or older versions running on other machines, as well as send modules for execution.

Air-gapped side functionalities

- Persistence: Exploited two EOP vulnerabilities: MS10-073 and Task Scheduler vulnerability. Then installed two drivers:
  - Rootkit: Installed as a service, would hide Stuxnet files in USB drives.
  - Load point: Installed as a service and loaded by Windows' boot process, it injected Stuxnet into a list of processes. Interestingly, one entry in the list was a file (module) to be loaded into `explorer.exe`, but the file was not created by Stuxnet.
- Reconnaissance: Collected system information (such as computer name, OS version, domain name, time, IP address) and information about Simatic Step 7 software and project paths; the information was appended to its configuration block forming a log of systems that Stuxnet had compromised. The information was sent via HTTP to its C&C servers in connected side machines.
- Propagation in air-gapped network: As previously listed in execution vectors.

Discussion

Stuxnet's ultimate purpose was not exclusively espionage, as in all the frameworks here described, it was designed to reach computers using WinCC and Step 7 software to attack PLC devices. Stuxnet did this by replacing one of the original Step 7 DLLs that communicated with the PLC; the fake DLL intercepts send/receive requests and it is able to inject code and hide its presence, like a rootkit. The code injected into the PLC was designed for sabotage.

8.4 IdentityKit #4: Fanny

Threat actor

Equation Group

Period of operation

2008–2012

Purpose

Performs reconnaissance and espionage against unknown entities in Pakistan, Indonesia, Vietnam, and China.

References

- 2015-02-16, Equation: The Death Star of Malware Galaxy, Kaspersky (*https://securelist.com/equation-the-death-star-of-malware-galaxy/68750*)
- 2015-02-17, A Fanny Equation: "I am your father, Stuxnet", Kaspersky (*https://securelist.com/a-fanny-equation-i-am-your-father-stuxnet/68787/*)

Characteristics

Connected side execution vector

- Unknown. Possibly deployed by another backdoor.

Air-gapped side execution vector

- Malicious LNK files exploited CVE-2010-2568 in USB drives.

Communication channel

- Connected side: In compromised systems with Internet access: contacted two C&C servers via HTTP and would receive commands to relay, exfiltrate collected data, or download a fully-featured trojan—usually DoubleFantasy.
- Air-gapped side: Covert communication channel via hidden storage space on USB drives allowed Fanny to receive commands.

Air-gapped side functionalities

- Persistence: Exploited MS09-025 to elevate its privileges to install a driver for persistence.
- Reconnaissance: Collected system information (OS information, list of processes, user and computer name, etc.).
- Propagation in air-gapped network:
  - Monitored registry changes to detect the insertion of new devices that matched USB drives and scanned their filesystems when they were FAT16 and FAT32 to find hidden storage space. When found, it would check for new commands that included updating its configuration and copy local files to the drive for exfiltration, copy a DLL to `%TEMP%` folder and load it, or load a DLL from the USB drive.

- When the USB drive was not affected, it created a hidden storage space and planted the LNK files that exploited CVE-2010-2568 to execute a malicious DLL to install Fanny on the system. The hidden storage space was created by changing the hint value that indicates where to look for available sectors, and created an invalid entry in the FAT directory: a combination of attributes unrecognized by Windows made it ignore it. The descriptor of the invalid entry contains a magic value as directory name, random data and the offset to the hidden storage space; looking for the magic value is how Fanny finds its storage space.

Espionage

- Would copy specific files and stage them together with information of systems for later exfiltration when a USB drive was connected.

Discussion

Fanny was one of oldest frameworks to compromise air-gapped networks, however: sporting two powerful zero day exploits at the time, driver, rootkit to hide its files, and a hidden storage space altering the filesystem of the drive for a covert communication channel and exfiltration point, made it one of the most sophisticated frameworks on this whitepaper. Indeed, some APTs seem to have learned from Fanny's design.

8.5 IdentityKit #5: miniFlame

Threat actor

Unclassified

Developed by the same factory as Flame and Gauss

Period of operation

2009–2012

Purpose

Perform reconnaissance and espionage against unknown entities in Iran, Sudan, Palestine, Saudi Arabia, Qatar, Lebanon, France, and the United States.

References

- 2012-10-15, miniFlame aka SPE: "Elvis and his friends", Kaspersky (*https://securelist.com/miniflame-aka-spe-elvis-and-his-friends/68560*)

Characteristics

Connected side execution vector

- Both components were downloaded by miniFlame when a requested was issued by command of the attackers. There components were:
  - `icsvntu32.ocx`: performed the weaponization of USB drives. On the connected side, it was installed in `%windir%\System32` and persisted using a COM hijack via the Windows registry.
  - `petsec.sys`: was launched from USB drives and performed reconnaissance.

Air-gapped side execution vector

- In connected side, `icsvntu32.ocx` collected information about USB drives and logged it. When a new USB drive was inserted and was not affected, it created the file `.thumbs.db` that included a TTL counter that limits the times the reconnaissance component would be launched: when the limit was reached, the drive is cleaned from all malicious content. `petsec.sys` was decrypted and written to the root of the drive as `CatRoot.tmp` or `System32.dat`, then it created all the necessary files and directories for the malicious LNK files that exploited CVE-2010-2568 and the junction point directories, that would launch the reconnaissance component.

Communication channel

- Connected side: miniFlame contacted its C&C servers via HTTP; the number of servers varied in each sample. Some of the servers were shared with Flame and its C&C framework was able to handle the communication protocols for Flame and Gauss as well. The collected information during reconnaissance were uploaded to the servers.
- Air-gapped side: No communication channel. Only exfiltration of collected information during reconnaissance.

Air-gapped side functionalities

- Persistence: `petsec.sys` did not persist on the system.
- Propagation: Had no propagation on the air-gapped side. USB drives are only weaponized on the connected side.
- Reconnaissance:
  - Collected information (such as computer name, OS version, platform type, network adapters, ARP table, processes and its modules, listing of files in root directories and network drives (with a max limit of files set at 200), network shares) and appended it to the `.thumbs.db` file on the USB drive.
  - Marked system as compromised by leaving a value in the Windows registry.
  - The component also checked the TTL value in `.thumbs.db` and when it reached zero, the drive was cleaned from all malicious content but the `.thumbs.db` file was left in it.

8.6 IdentityKit #6: Flame

Threat actor

Unclassified

Developed by the same factory as miniFlame and Gauss

Period of operation

- Flame 1.0: 2010–2012
- Flame 2.0: 2014–2016

Purpose

Perform extensive espionage operations against unknown entities in Iran, Palestine, Sudan and Syria, among other countries.

References

- Flame 1.0: 2012-28-04, The Flame: Questions and Answers, Kaspersky (*https://securelist.com/the-flame-questions-and-answers/34344/*)
- Flame 2.0 2019-14-04, Who is GOSSIPGIRL?, Chronicle Security (*https://medium.com/chronicle-blog/who-is-gossipgirl-3b4170f846c0*)
- 2012-06-04, 'Gadget' in the middle: Flame malware spreading vector identified, Kaspersky (*https://securelist.com/gadget-in-the-middle-flame-malware-spreading-vector-identified/33081/*)

Characteristics

Connected side execution vector

- Unknown. It is suspected that attackers used an exploit for MS10-033 vulnerability, but remains unconfirmed.

Air-gapped side execution vector

- USB thumb drives:
  - File `autorun.inf` that contained the malware and `shell32.dll` trick to maximized the chances of execution by a potential victim. Same as Stuxnet.

  - Malicious LNK files exploited CVE-2010-2568 with junction point directories, same as Stuxnet but improved exploit.
- Flame's local network spreading capabilities might have enabled it to spread on AGN via:
  - Fake updates signed with forged but valid certificates and delivered via Windows Update interception.
  - MS10-061 Windows Print Spooler exploit.
  - Remote Scheduled Tasks.
  - If Flame had administrative rights on the domain controller, it created backdoor user accounts that would be used to copy itself to remote machines.

Communication channel

- Connected side: Contacted C&C web servers via HTTP with SSL. Flame had a large list of domains, with one sample using up to 11 domains to contact its C&C servers.
- Air-gapped side: Covert communication channel used an encrypted database in a hidden file in the USB drive.

Air-gapped side functionalities

- Persistence: Flame installed itself in the Windows registry as an authentication package; in the next system boot it was automatically loaded by `lsass.exe`.
- Reconnaissance: Flame collected a wide range of system and network information as well as whole documents or metadata, images and GPS coordinates, technical CAD drawings and more.
- Propagation: In systems with no connectivity Flame monitored for the arrival of USB drives; if the device had not been weaponized, it created the file `HUB001.dat`, which contained a database with all the collected information, and created all the necessary files and directories to exploit CVE-2010-2568, or, in older versions, created an `autorun.inf` file. Flame implemented a special trick to use `HUB001.dat` as a covert communication channel: in USB drives with FAT filesystem, it renamed `HUB001.dat` to ".", an invalid name that is ignored by Windows when parsing the contents of the filesystem, thus making it "invisible".
- If the drive was weaponized, Flame forwarded its collected information to the existing database and looked for components whose version was higher to update itself. The database also contained a TTL value, when it reached the limit, the contents of the database were stored on the local system and the drive was cleaned.

8.7 IdentityKit #7: Gauss

Threat actor

Unclassified

Developed by the same factory as Flame and miniFlame

Period of operation

2011-2012

Purpose

Perform reconnaissance and espionage against unknown entities in Lebanon, Israel, Palestine, the United States, and other countries.

References

- 2012-09-08, Gauss: Abnormal Distribution, Kaspersky (*https://securelist.com/gauss-abnormal-distribution/36620*)

Characteristics

Gauss was very similar to miniFlame in its strategy to perform reconnaissance of air-gapped networks. There were two components: one to weaponize USB drives (called Kurt or Godel) and a special payload that performed collection of information when a new host was reached. In the IdentityKit for miniFlame we described those modules rather than miniFlame as a whole because they were downloaded and the component in charge of weaponizing USB drives persisted in the system and worked independently of miniFlame; however, in Gauss the respective component did not persist and was loaded by the orchestrator. This is an important distinction between the implementation of Gauss and miniFlame.

Connected side execution vector

- Unknown.

Air-gapped side execution vector

- On the connected side, the Kurt/Goldel component monitored for the arrival of new USB drives by querying the Windows registry. When a new drive was inserted, it created the malicious LNK files that exploited CVE-2010-2568, and a file named `.thumbs.db`. The payloads `System32.dat` (for 32-bit system) and `System32.bin` (for 64-bit system) were extracted from `dskapi.ocx` resources 100 and 101 and written to the USB drive.
- To check whether the USB drive was already affected it looked for the `.thumbs.db` file, and if it matched the expected format, it was copied to local storage and later exfiltrated to its C&C servers, the USB drive was cleaned from all malicious content and weaponized again.

Communication channel

- Connected side: Gauss contacted its C&C servers via HTTPS, and collected reconnaissance results were uploaded.
- Air-gapped side: No communication channel. Only exfiltration of collected information during reconnaissance.

Air-gapped side functionalities

- Persistence: Although the payload was copied to the local `%TEMP%` directory and reloaded from there, it did not persist on the host.
- Reconnaissance:
  - It collected system and network information (OS version, workstation information, network adapters, routing table, process list, environment variables and disk information, network shares, network proxy, MS SQL servers, URL cache.
  - Collected information was encrypted and appended to the `.thumbs.db` file. It also checked the TTL value contained there and when it reached the limit, the USB drive was cleaned from all malicious content, leaving only the `.thumbs.db` file.
  - Gauss payloads contained special sections that were decrypted using a key generated by a complex algorithm that used data collected from a previous reconnaissance pass. When the system was reached again and the conditions matched to generate the key, the contents of the sections were decrypted with RC4 algorithm and executed.
- Propagation: No propagation in air-gapped side. USB drives were only weaponized in connected side.

Discussion

Gauss payloads are an interesting case because now we have an example of one of the purposes behind the approach of slowly mapping and surveying the systems in the air-gapped network: with the information collected, the attackers are able to encrypt a payload that only can be decrypted on specific computers. Given the complexity of generating such a password, it hasn't been possible to decrypt the contents of the sections in the payloads. Read more about it here: *The Mystery of the Encrypted Gauss Payload.*

8.8 IdentityKit #8: USBFerry

Threat actor

Tropic Trooper

Period of operation

December 2014–May 2020 (at least)

Purpose

Steal defense—and maritime—related documents from government, military and navy institutions and national banks in Taiwan and the Philippines.

References

- 2020-05-12, Tropic Trooper's Back: USBferry Attack Targets Air-gapped Environments, Trend Micro (*https://documents.trendmicro.com/assets/Tech-Brief-Tropic-Trooper-s-Back-USBferry-Attack-Targets-Air-gapped-Environments.pdf*)

Characteristics

Connected side execution vector

- Malicious software installer sent via email.

Air-gapped side execution vector

- The malware component that was running on a connected system would place USBferry and an `autorun.inf` file on newly inserted USB drives. This execution vector would not work on recent Windows systems with AutoRun/AutoPlay disabled.

Communication channel

- Connected side: Contacted C&C servers via HTTP.
- Air-gapped side: No commands, just automatic exfiltration of documents straight to USB drives.

Air-gapped side functionalities

- Persistence: Registered itself as a Winlogon helper DLL in Windows registry in `Shell` subkey[8].
- Reconnaissance and espionage:
  - Automated host reconnaissance based mostly on built-in Windows commands (ipconfig, net view, arp, netstat, net share, tracert). Obtained host network configuration and connectivity capability, limited attempt to get network topology (using ping and tracert) and volume information.
  - Copied files to USB drives and monitored when the files were last written to, so as to update them to their latest versions.
- Propagation: The same as described in air-gapped side execution vector.

8 *https://attack.mitre.org/techniques/T1547/004/*

8.9 IdentityKit #9: USBCulprit

Threat actor
Cycldek/Goblin Panda/Conimes

Period of operation
2014-2019

Purpose
Steal documents and network information of compromised computers from government organizations across several Southeast Asian countries, mostly Vietnam, Thailand and Laos.

References
- 2020-06-03, Cycldek: Bridging the (air) gap, Kaspersky (*https://securelist.com/cycldek-bridging-the-air-gap/97157/*)

Characteristics
Connected side execution vector
- Seen downloaded by RedCore malware.

Air-gapped execution vector
- USBStealer copied itself to a USB drive when a file named `2.txt` was found on the drive from the connected side or air-gapped side, but no execution vector was created or copied. Suspected human factor required to execute the malware on the air-gapped side.

Communication channel
- Connected side: Unknown. Possibly performed via RedCore.
- Air-gapped side: Primitive mechanism: the presence of a particular file (for example, `1.txt`) on the USB drive indicated whether USBCulprit should do something.

Air-gapped side basic functionalities
- Persistence: Registered itself in Windows registry Run key.
- Reconnaissance and espionage:
  - Some variants had host network reconnaissance functionality using built-in Windows commands (ipconfig, ping, arp, net view, nbtstat, tasklist).
  - Searches for files with extension `*.pdf;*.doc;*.wps;*docx;*ppt;*.xls;*.xlsx;*.pptx;*.rtf` from hardcoded list of directories. The files were grouped and archived in a password-protected RAR file and copied to the USB drive only if no `1.txt` ws found in the drive, when it's found, USBCulprit expected to find an archive of exfiltrated documents in `$Recyc1e.Bin` folder and made a local copy.
  - Ability to execute arbitrary files on newly inserted USB drive (update the malware, add functionalities).
- Propagation: Selective partial propagation: USBCulprit only weaponized a USB drive when file `2.txt` was found on it.

8.10 IdentityKit #10: Retro

Threat actor
DarkHotel

Period of operation
2017

Purpose
Perform espionage and mapping of air-gapped networks of unknown entities.

References
- 2018-04-23, The latest APT organization "parasitic beast" activity disclosure, Tencent (*https://s.tencent.com/research/report/465.html*)
- 2020-05-28, Advanced threats: Ramsay malware's attack technology analysis against isolated networks, Tencent (*https://s.tencent.com/research/report/1000.html*)

Characteristics
Connected side execution vector
- Spearphishing with decoy document that exploited CVE-2017-11882.

Air-gapped side execution vector
- In connected side: plug-in `infsvc.exe` monitored for the arrival of USB drives to find and convert suitable Word documents into RTF format and added an RCE exploit for CVE-2017-8570 to execute a reconnaissance component that was copied to the `%TEMP%` folder in the local machine.

Communication channel
- Connected side: Unknown.
- Air-gapped side: No communication with the attackers. Only exfiltration of files.

Air-gapped side basic functionalities
- Persistence: Unknown.
- Reconnaissance and espionage:
  - When executed in USB drives, the reconnaissance component was copied to the local `%TEMP%` as `taskhost.exe`. It collected system and network configuration information using Windows built-in commands with the output redirected to a `.rtt` file on the local machine; commands included systeminfo, tasklist, netstat, ipconfig, netsh, and more.
  - Created a listing of all directories and files in the system and saved it to its respective `.rtt` file.
  - Took a screenshot every five minutes and saved them as `.jpg` files, then encrypted it and stored in the another local storage folder.
  - When all the information was collected, it abused WinRAR to create a compressed archive of everything in its local storage folders.
  - Monitored for the arrival of USB drives and copied the archive to the drive's offset 1000000000 with a magic value. It would later be extracted back in the computer that was running Retro with its `infsvc.exe` plug-in.
- Propagation: None.

8.11 IdentityKit #11: PlugX

Threat actor
Mustang Panda

Period of operation
2018-2020

Purpose
Perform espionage against unknown entities in Hong Kong, Vietnam, Australia and China (according to Avira), with more specific targets identified by Anomali based on the decoy documents.

References
- 2019-10-07, China-Based APT Mustang Panda Targets Minority Groups, Public and Private Sector Organizations, Anomali (*https://www.anomali.com/blog/china-based-apt-mustang-panda-targets-minority-groups-public-and-private-sector-organizations*)
- 2020-01-31, New wave of PlugX targets Hong Kong, Avira (*https://www.avira.com/en/blog/new-wave-of-plugx-targets-hong-kong*).

NOTE: Both reports analyzed the same operation, but Avira is the first to have analyzed the component built to target air-gapped networks. Hence, we consider the Avira report as the date of discovery.

Characteristics
Connected side execution vector
- Spearphishing emails with ZIP files that contained a multistage loading mechanism that ultimately installed the malware and a decoy document.

Air-gapped side execution vector
- PlugX monitored for the arrival of USB drives and weaponized them by creating a hidden folder called `RECYCLE.BIN` where it copied its files. In the root of the drive it created LNK files with interesting names to entice the user into launching them.

Communication channel
- Connected side: Contacted its C&C servers via HTTP.
- Air-gapped side: No communication with the attackers. Only for exfiltration of collected information.

Air-gapped side basic functionalities
- Persistence: Registers itself in Windows registry Run key.
- Reconnaissance and Espionage:
  - When PlugX detected that it is on a system with no internet connectivity, it activated its air-gapped mode. Tried to find the USB drive from where it was executed and dropped a BAT file that contained commands to log information about the system and network (some commands include ipconfig, netstat, arp, tasklist, systeminfo); output was directed to files on the USB drive.
  - It searched the entire system for documents (`.doc, .docx, .ppt, .pptx, .xls, .xlsx, .pdf`) and copied them to the USB drive and encrypted them with RC4. In some versions it could also do this by abusing WinRAR software to have the program create an archive and instruct it to search and compress all of the files in a given directory.
- Propagation: Same as air-gapped side execution vector.

Discussion
PlugX is a malware that is used independently by different groups. Only Mustang Panda has customized it with capabilities to attack air-gapped networks.

9. APPENDIX 2: OFFLINE/HUMAN ASSET FRAMEWORKS OVERVIEW

With the development of new technologies, our visibility into nation-state efforts grew as their operations expanded and grew exponentially, allowing the research community to discover and document some of the most sophisticated frameworks to date where air-gapped malware is only one component of a large toolkit designed for seemingly boundless collection of intelligence.

The Vault7 document leaks opened a window into the development branches that create these types of capabilities; from these documents we learned about their so-called "implants". Some have been found and documented publicly but in many cases we couldn't match them to any malware samples in our collections.

We decided to include in this paper IdentityKits of a particular set of documented tools from the Vault7 leaks, to refresh our knowledge and to describe the potential threat these tools could pose to air-gapped networks, if ever put in the hands of capable attackers. One exception among these special cases is ProjectSauron: a sophisticated toolkit employed by an unknown threat actor, reported and documented by Kaspersky, that includes exfiltration capabilities inside AGNs.

9.1 IdentityKit #12: ProjectSauron

Threat actor

Unclassified

Period of operation

2011–2016

Purpose and targets

Perform espionage against

- Kaspersky Telemetry: Government, scientific research centers, military, telecommunication providers, and finance in Russia, Iran, Rwanda and possibly in Italian-speaking countries as well.
- Symantec Telemetry: Individuals located in Russia, an airline in China, an organization in Sweden, and an embassy in Belgium.

References

- 2016-08-08, ProjectSauron: top level cyber-espionage platform covertly extracts encrypted government comms, Kaspersky (*https://securelist.com/faq-the-projectsauron-apt/75533/*)
- 2016-08-08, Strider: Cyberespionage group turns eye of Sauron on targets, Symantec (*https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=ce2df4da-afe9-4a24-b28c-0fb3ba671d95&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments*)

Characteristics

ProjectSauron is a special case, as we only know of one component for the air-gapped side

- MyTrampoline: A plug-in that, when loaded, waited for the arrival of USB thumb drives to search after the its first partition for a shadow filesystem where MyTrampoline stages the files that were prepared for exfiltration in a local storage folder; files were also extracted from the shadow filesystem and copied to the local storage. MyTrampoline had no capabilities to weaponize USB drives.
- The shadow filesystem was placed after the first partition with folders for "in" and "out" where files were staged for exfiltration and extraction by MyTrampoline. As far as we know, there were no special files containing commands or new blobs to be executed.

We only know one component that we suspect would be used on the connected side

- Online Pusher v2: collected the files in the local storage that were extracted by MyTrampoline and exfiltrated them to a server that worked as a "vault", or repository for stolen information.

However, we are missing the component that would weaponize USB thumb drives, meaning that we don't know what was the execution vector that enabled ProjectSauron to go from the connected to the air-gapped side and persist a special configuration that collected files and launched MyTrampoline. Unlike the case of Flame or Gauss, no component has been found that checked when a certain limit was reached to clean USB drives, as to not continue compromising more machines and risk an escape à la Stuxnet.

9.2 IdentityKit #13: EZCheese

Source

Vault7 Leaks—Wikileaks in 2017

Period

(Estimated) 2013–2016

Purpose

Perform espionage against unknown entities.

References

- *https://wikileaks.org/vault7/document/EzCheese-v6_3-User_Guide_Rev__2014-01-07/EzCheese-v6_3-User_Guide_Rev__2014-01-07.pdf*

Characteristics

EZCheese was a tool to perform espionage by preparing a USB drive with LNK files that contained an exploit as execution vector and a reconnaissance DLL component. It was intended for operations with an asset on the ground to manually plug in the USB drive and navigate to the folder where the LNK files were located with execution happening automatically.

Connected side execution vector

- Unknown.

Air-gapped side execution vector: Giraffe exploit

- Windows XP SP3, Windows Vista, Windows 7 and Windows 8 LNK exploit. At the time when the document was written, the vulnerability had been patched.

Communication channel

- Air-gapped side: No commands, just automatic exfiltration of documents straight to USB drives.

Air-gapped side basic functionality

- Persistence: No persistence mechanism.
- Reconnaissance and espionage:
  - EZCheese reconnaissance component had the ability to collect information about the operating system, BIOS, user accounts, installed applications, computer information, services and processes, directory listings, perform WMI queries, query Windows registry, and collect files in the system or specific folders.
- Propagation: No propagation mechanism.

9.3 IdentityKit #14: Emotional Simian

Source
Vault7 Leaks—Wikileaks in 2017

Period of operation
Starting in 2013

Purpose
Espionage.

References
- *https://wikileaks.org/vault7/document/Emotional_Simian-v2_3-User_Guide/Emotional_Simian-v2_3-User_Guide.pdf*

Characteristics
Connected side execution vector
- Deployable by means of another implant installed on the system.

Air-gapped side execution vector
- On the connected side an implant "server" is installed (with an unknown persistence mechanism) that monitors for the arrival of whitelisted USB drives to weaponize them with LNK files used to load the payload DLL. It's unknown if those LNK files exploit any vulnerability as in the case of EZCheese.
- Whitelisted drives are a set of USB drives known to the operator. It's likely that there is a component on the connected side, on a machine previously compromised by another implant, that collects information about USB drives and their drive serial number, the information is then used to conFigure commands for the "server" component in the compromised machine.

Communication channel
- Air-gapped side: Covert communication channel hidden in USB thumb drives, described as a "covert partition" but no further information about it.

Air-gapped side basic functionalities
- Persistence: Unknown mechanism.
- Reconnaissance: Payload collects BIOS and computer information, installed applications, OS information, processes, services, user accounts, devices, directory listings, and various types of files.
- Propagation: None.

9.4 IdentityKit #15: USBThief

Threat actor
Unclassified

Period of operation (estimation)
2014-2015

Purpose
Collect a wide range of system and network information and steal documents from unknown entities in Côte d'Ivoire.

References
- 2016-23-03, New self-protecting USB trojan able to avoid detection, ESET (*https://www.welivesecurity.com/2016/03/23/new-self-protecting-usb-trojan-able-to-avoid-detection*)

Characteristics

Connected side execution vector

- Unknown.

Air-gapped side execution vector

- Hijacked portable applications (Firefox, Notepad++ and TrueCrypt) placed on USB drives.

Communication channel

- Connected side: Unknown.
- Air-gapped side: No communication channel with the attackers. Only exfiltration of collected information.

Air-gapped side basic functionalities

- Persistence: None.
- Reconnaissance:
  - When the first loader of USBThief was executed through a hijacked application, a multistage loading began until the final payload was loaded and injected into the process `svchost.exe`. USBThief's loading mechanism had been designed to make it difficult to recover and decrypt its next stages; one interesting technique that it used was encrypting one of the stages with an AES key derived from the USB drive Volume Serial Number.
  - The payload collected extensive system information including the entire Windows registry tree from HKCU root key, a listing of all files in all directories, several types of documents and images. Many more capabilities were enabled through an open source library called WinAudit.
  - All the collected information was stored encrypted on the USB drive.
- Propagation: None.

Discussion

USBThief is one of the most interesting cases among all the frameworks described here. Following the Vault7 leaks, many people (including at ESET) *have been theorizing* that USBThief, first known approximately a year before the leak, could be the program called Rainmaker that is described in the Vault7 documents. However, we now think that USBThief shares more similarities with another program called Margarita.

From the documentation of Rainmaker, we know it was designed to hijack the popular media player software VLC by placing a manifest file with a Side-by-Side assembly to force Windows to load `psapi.dll` from the current directory where VLC is running (on the USB drive). The Rainmaker Stub DLL will be loaded instead and it will redirect the API used by VLC to the real `psapi.dll` and decrypt the information stealer DLL component and load it into memory with its own custom loader.

The documentation states that it's decrypted with AES algorithm and a key generated from the USB drive's assigned Volume Serial Number. This description matches how USBThief's 2nd stage loads the DLL of its 3rd stage which in turns loads the payload (as seen in Figure 17). But this is also where the strong similarities end: Rainmaker is designed to collect documents from certain paths and store then in an Alternate Data Stream folder.

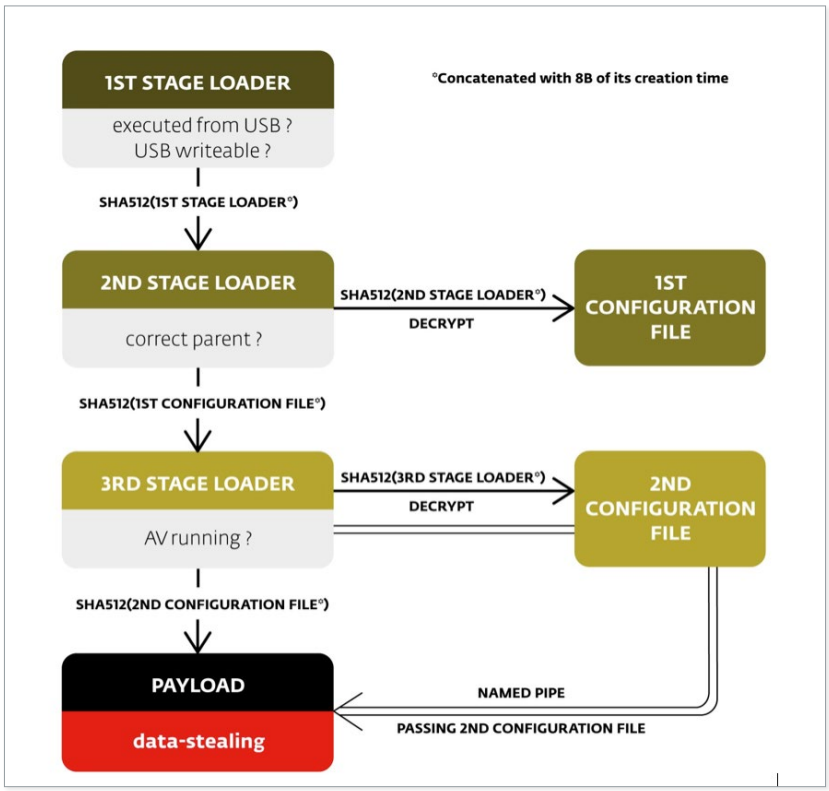

Figure 17 // Illustration of USBThief's multistaged loading process
Source: https://www.welivesecurity.com/2016/03/23/new-self-protecting-usb-trojan-able-to-avoid-detection/

Margarita, on the other hand, supports at least three selectable execution vectors, one of which is called OpenBar, described as a collection of techniques to trojanize portable applications. Margarita's information stealer component contains an extensive list of capabilities that match that of USBThief. Collected data is staged in `.dat` files and the folder's naming format displayed in screenshots of Margarita's configuration software also matches what was seen in USBThief's configuration files.

Rainmaker and Margarita describe a malicious program and concept of operation in which a human asset is in charge of the covert execution of malicious code with espionage capabilities. With USBThief we can conclude that these capabilities have been deployed in the real world by a nation-state actor and with a measure of success given Margarita was developed at least in 2011 and newer tools for new missions were developed at least until 2016.

9.5 IdentityKit #16: Brutal Kangaroo

Source
Vault7 Leaks—Wikileaks in 2017

Period of operation
Starting in 2016.

Purpose
Espionage

References
- https://wikileaks.org/vault7/#Brutal%20Kangaroo
- https://wikileaks.org/vault7/document/#brutalkangaroo
- https://wikileaks.org/vault7/document/Shadow-v1_0-User_Guide_2012-08-31/Shadow-v1_0-User_Guide_2012-08-31.pdf

Characteristics

Brutal Kangaroo is a tool that enables collection of intelligence in a target system or network. Its operator can weaponize the drive and conFigure its execution vectors as well as the payloads to perform a survey of the system or execute implant payloads such as Shadow: a persistent implant to be installed to covertly collect information for an extended period of time and exfiltrate it through prepared USB drives that have a hidden storage space. Shadow is also capable of receiving commands to exfiltrate certain specific files or collect files by matching patterns.

Brutal Kangaroo was a newer development that deprecated other similar tools and implants such as EZCheese and Emotional Simian.

Connected side execution vector

- Unknown

Air-gapped side execution vector

- EZCheese (Giraffe Links): Windows XP SP3, Windows Vista, Windows 7 and Windows 8 LNK exploit. At the time when the document was written, the vulnerability had been patched.
- Okabi Links: Windows 7 `autorun.inf` with LNK files exploit (CVE-2015-0096).
- RiverJack Links: Windows 7, Windows 8 and Windows 8.1 exploit using junctions, library-ms and LNK files.
- Direct execution.

Communication channel

- Air-gapped side:
  - Shadow: Covert communication channel hidden in USB thumb drives. The operator can create a new set of commands and payloads for a specific Shadow implant in the AGN network, or for all implants. It's received from connected side and handled by the Shadow components that relays the commands via USB drive channel.

Air-gapped side basic functionalities

- Persistence:
  - Brutal Kangaroo payloads don't persist in the compromised system.
  - Shadow implant persists in the system as a service.

Reconnaissance

- Brutal Kangaroo: The suite includes a set of payloads designed for collection of information, with the following capabilities:
  - System survey.
  - Directory listing of removable drives, fixed drives, remote drives, CD drives.
  - Collect files by pattern with rules on exclusion of folder, maximum or minimum size, or by creation/access/modification time.
  - Survey of USB drives in the system.
- Payload stage collects information via one of three selectable methods:
  - NTFS Alternate Data Stream.
  - Writes data to a (new) file in the drive.
  - Appends the data to an existing image file, a magic value is used to locate the start of the data.
- Shadow: The implant has the capability to collect files by extension or specific files. Shadow exfiltrates the files by creating a hidden storage space after the first partition on the USB drive.

Propagation

- Unknown.

9.6 IdentityKit #17: Ramsay

Threat actor
DarkHotel

Period of operation
Unknown
Assessment is that the framework was under development at time of analysis (2019-2020)

Purpose
Perform espionage against unknown targets.

References
- 2020-05-13, Ramsay: A cyber-espionage toolkit tailored for air-gapped networks, ESET (*https://www.welivesecurity.com/2020/05/13/ramsay-cyberespionage-toolkit-airgapped-networks/*)

Connected side execution vector
- Unknown

Air-gapped side execution vector
- Malicious RTF documents exploiting CVE-2017-0199 (Office/WordPad RCE), CVE-2017-11882 (Office RCE).
- Trojanized 7zip installer.
- Trojanized PE files.

Communication channel
- Connected side: Ramsay does not have network capabilities to contact remote C&C servers.
- Air-gapped side: Searches network shares and USB drives to find potential files that might contain commands, these files are Microsoft Word documents, PDF and ZIP files.

Air-gapped side basic functionalities
- Persistence: Multiple mechanisms:
  - Load via AppInit_DLLs registry key.
  - Scheduled Task via COM API.
  - Phantom DLL Hijacking of Windows services: Windows Search hijacking `msfte.dll`, Microsoft Distributed Transaction Coordinator hijacking an Oracle dependency `oci.dll`.
- Reconnaissance and Espionage:
  - Ramsay's primary collection target is Microsoft Word documents that it searches for on the system, removable media and network drives. Located documents are staged in a preliminary collection directory on the local system drive where they are compressed with a dropped instance of WinRAR and encrypted with RC4 algorithm. All the files are added to a special container blob that includes a magic value and the Hardware Profile GUID of the compromised system to identify from which system they were collected. Ramsay scans again to find Word documents to append them to the container. The extraction of these containers and exfiltration to servers is done by a component that has yet to be discovered.
  - In addition to collection of documents, some components have implemented a network scanner to find hosts susceptible to the EternalBlue SMBv1 vulnerability (CVE-2017-0144)
- Propagation: Scans system drive and network shares to trojanize PE files.

ABOUT ESET

For more than 30 years, ESET® has been developing industry-leading IT security software and services to protect businesses, critical infrastructure and consumers worldwide from increasingly sophisticated digital threats. From endpoint and mobile security to endpoint detection and response, as well as encryption and multifactor authentication, ESET's high-performing, easy-to-use solutions unobtrusively protect and monitor 24/7, updating defenses in real time to keep users safe and businesses running without interruption. Evolving threats require an evolving IT security company that enables the safe use of technology. This is backed by ESET's R&D centers worldwide, working in support of our shared future. For more information, visit www.eset.com or follow us on *LinkedIn*, *Facebook* and *Twitter*.

ENJOY SAFER TECHNOLOGY™