Dissecting Flame: A Contemporary Malware for Cyber Espionage

Syed Muhammad Kumail Raza and **Zulfiqar Ibrahim**[1]
Information Security, WS-2018-19
Technical University of Kaiserslautern, Germany
sraza@rhrk.uni-kl.de, zulfikaribraheem@hotmail.com

***Abstract*— Cyber espionage has been the primary objective of majority of hacker groups. Malwares intended for this purpose focus on data collection through privacy breach and exploitation of security loopholes. Flame belongs to this class of so-called "targeted malwares" which targets communication devices & computer systems for collecting sensitive data and spreading itself across the network. It has been used to attack the computer systems of the energy-sector in the Middle East and South America to collect large amounts of sensitive data as well as to disrupt their operations. This paper presents the unique and sophisticated properties of targeted malwares dissecting Flame as a case study, exploring it's unusual exploitation mechanisms, underlying modules, variants and its execution architecture. The paper also present some unique challenges in Flame's detection and discusses why is it so hard to detect these malwares & to mitigate their impact. It then outlines some heuristics and guidelines to help counter this challenge. It also illustrate some statistics to show the estimated scale of impact of the Flame attack. Our goal is to understand the execution dynamics of this sophisticated malware which is implicitly violating the confidentiality, integrity and availability of sensitive-data, systems and security protocols, and derive heuristics to help detect and mitigate the attack.**

I. INTRODUCTION

Malware is a sequence of instructions designed to sabotage the intended behavior of the system without the users consent. It is a critical attack method as it can obfuscate itself in the system and present itself as a harmless program. Any software that deviates the system from instructed functionality to attackers desired functionality is also a Malware, Virus or Trojan. These malicious applications are used to gather sensitive information, they are becoming smarter in their method of attacks and require a comprehensive strategy for dealing with them. In June 2010, the Stuxnet malware [1] marked the start of a new era in the arms-race in cyber security. First time in history, a targeted cyber attack was discovered that aimed at physically destroying part of the critical infrastructure of a state. Reportedly, Stuxnet was not the first targeted attack against industrial systems [2], but the first one to receive worldwide attention due to its unique purpose as a cyber weapon. Thanks to the increasing attention, the shared knowledge and growing efforts of the malware research community, several new targeted threats that are related to Stuxnet in some way have been discovered. In October 2011, the CrySyS Lab in Budapest, Hungary discovered Duqu [3], a malware with striking similarities to Stuxnet, but apparently with a different objective. Indeed, Duqu does not aim at causing physical damage, but it is an information collecting malware used for cyber espionage. Flame is another information-collecting malware built on a platform different from that of Stuxnet and Duqu. Yet, researchers found identical code segments in an early Stuxnet variant and Flame, making them believe that Flame belongs to the same cyber espionage operation and it is indeed member of the Stuxnet family. Flame received worldwide attention of security researchers and practitioners due to its advanced spreading techniques based on masquerading as a proxy for Windows Update [4].

The campaign of malware-based attacks targeting the middle east in 2012 provides evidence of manifestation of these types of malwares for cyber-espionage [4]. Indeed, several organizations in the middle east, in particular in the energy industry, reported recently infections with sophisticated malware which exhibit suspicious similarities. Figure 1 is a graph by Kaspersky which shows the concentration of these attacks in the middle-east for political purposes. As in most sectors, attackers are often after valuable information. For example, we have seen attackers target intellectual property such as technology for photovoltaic research and wind turbines, or data on gas field exploration. Information such as this is of high value and can generate huge profits for attackers or their sponsors. The same information can also be misused for an act of sabotage. Many power utilities companies fear disruptive attacks the most, regardless of whether it is done by internal or external attackers. The energy sector has a high potential for critical disruption through sabotage attacks. Any interruption to the power grid would cause substantial chaos and cascading effects resulting in financial loss.

Flame being one of the most sophisticated one of these attacks, was quite unusual as a malware in the sense that it was an order of magnitude larger than typical malware samples (both for generic and targeted attacks). This paper gives an overview of the impact of the Flame malware as well as the disection of the malware itself. It also presents the main detection techniques and evaluates the state of current cyber-security infrastructures to tackle such sophisticated malwares. The paper is a result of an extended survey of a large number of technical reports as it rehashes available reports focusing on the key features of the malware and the new hacking techniques. We consider carrying out this exercise essential to understand the recent hacking trends and hopefully to forecast future attacks. The following section

[1]Department of Informatik, Technical University of Kaiserslautern www.uni-kl.de

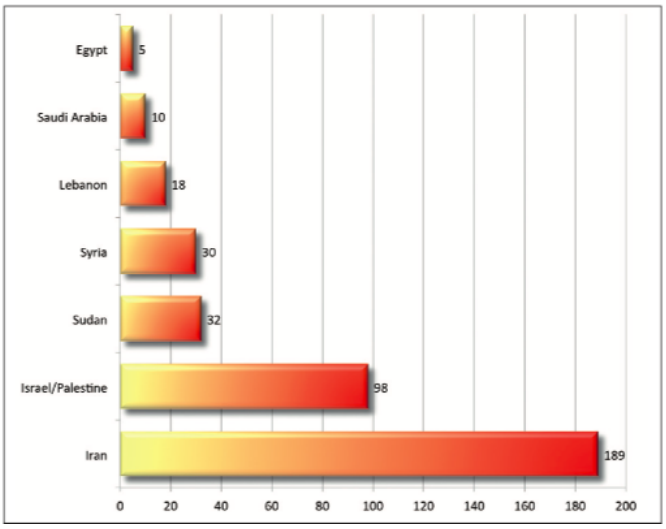

Fig. 1. The figure show the infections initially identified by Kaspersky were concentrated in the Middle East. Source: Kaspersky Lab [5], [4].

present a literature review of modern characteristics and trends of malwares. The remaining sections follow presenting disection of the Flame malware with it's modules, variants and working infrastructure, it's detection and mitigation. Finally some predicted future challenges and conclusion.

II. Review of Recent Malwares

Several common trends are emerging from successful recent malware attacks showing once again the inefficiency of the current security mechanisms. It is important to point out and discuss these trends in order to understand where the security field is heading. The following gives a brief account of the trends identified by the cyber-security experts in the literature.

The most striking characteristic of the recent malwares is their sophistication. This sophistication is a result of a significant amount of effort and resources spent in the development of the malware. There is a wide agreement that only teams of developers working for long period of times (several months if not years) could produce such advanced attack toolkits [4]. Recent malware are directed towards specific targets. The spreading mechanism is often controlled. The goal is not to infect the maximum number of victims. It is rather to infect a specific set of targets. At the same time, this contributes in keeping the malware undetected and working under the radar for a longer period of time. A targeted malware is a bigger threat to networks than mass malware, because it is not widespread and security products will not be able to provide a timely protection against it [6]. A common feature of all described malware is the use of legitimate certificates to sign parts of their malicious code. In the case of Stuxnet and Duqu, stolen certificates have been used to sign drivers allowing a stealth installation [7]. In the case of Flame, a limited certificate has been leveraged by the attackers to sign malicious binary code exploiting a flaw in an old signing algorithm. In the case of Shamoon [8], a third-party certified disk driver has been used as is to allow a user-level application to write on the Master Boot Record. Bencsath and Pek [9] provide a relevant discussion about using certificates to sign malicious code. Malwares commanded by a C&C server are common in cyber attacks. However, malware extending their capabilities while deployed in the victim is quite new and has been extensively used in the cyber attacks campaign on the middle east [2]. Downloading additional modules and executables after deployment allows to fine-tune the attack as more appropriate features are used for specific targets. Duqu malware used an extreme version of this feature as new modules are compiled and built specifically for every new infection [9]. This feature allowed Flame to remain undetected for a long period of time as the module in charge of escaping security products was continuously updated by downloading regularly improved versions of the module.

USB drives, in addition to zero-day exploits, are emerging as the main infection vector in targeted attacks [10]. Crafting a USB drive with a malicious LNK or autorun.inf file has been extensively used in particular in the initial infection in a LAN [11]. In Flame attacks, USB drives were used to steal information from victims in protected zones (without internet connection). In the case of Stuxnet, there are two different speculations for the initial infection vector of the Natanz uranium enrichment facility and both of them involve USB drives. The first theory suggests that the malware has been spread via flash drives distributed at a SCADA conference [12]. The second theory suggests that the malware got inside the facility through a Russian integrator company that built the plant where one of the engineers was lured to accept an infected USB drive which infected his laptop and then the PLC which he is manipulating [13].

All described malware (except Shamoon) have an uninstallation module. The module completely removes the malware from a system, deleting every single trace of its existence. The module can be launched remotely by the attack center (through the C&C servers) at any time [14]. Due to the amount of effort required in the development of these advanced malware, the attackers commanded the malware to commit suicide (both at infected machines and at the C&C servers) as soon as it is detected and related details posted online [14]. This makes any forensics investigation very difficult to carry out and leaves the possibility to deploy another variant of the malware in future attacks.

III. Dissecting Flame

Flame is the most sophisticated computer malware ever seen by the industry [15]. Flame, also known as Flamer and Skywiper is designed to steal and attack various databases [16]. It was discovered in late May 2012 however there are several evidences it is active in the wild as early as February 2010 [2]. Flame is a sophisticated attack platform with uncommon features such as the large size (900KB for the bare-bone version and 20MB when fully deployed), the

use of a LUA virtual machine, the use of bluetooth and above all the ability to steal data in so many different ways. A distinct functionality is used by flame malware is *Audio Spying* that can record audio, screenshots and can monitor keyboard activities and network traffic. For example flame has capabilities of keeping the records of video-call conversation by detecting and recognizing a microphone activity on the infected computer [7] violating its *confidentiality*. Flames constitution is quite complex with a large number of components and substantial size of some of its files. Our goal in this section is to get a quick understanding of the malwares purpose, and to identify its main modules, storage formats, encryption algorithms, injection mechanisms and activity in general. Other researchers contributed further results on Flame, including the identification of its modules and the description of the MD5 hash collision attack that enabled Flame to masquerade as a proxy for Windows Update [17]. The following section presents overview functionality and description of the module and variants of flame malware.

A. Modules and Variants

Many parts of Flame modules are written in Lua. They are then interpreted through the Lua virtual machine. The presence of the Lua interpreter and the fact that the modules are written in Lua makes it very easy to extend the functionalities of the malware by other modules downloaded from the attack center. The following points outline the main modules of Flame with their objectives.

- Flame has worm capabilities and hence can spread from one system to another. The spreading techniques are implemented in several modules and include: the use of network shares, USB using a malicious autorun.inf file (used also by Stuxnet), and more importantly using zero-day exploits, in particular the LNK (MS10-064) exploit [12].
- The main objective of Flame Malware is to steal data from infected systems [18]. The three modules FLASK, JIMMY, and MICROBE are in charge of gathering enormous amount of information about the victims.These modules use intelligence to collect relevant data: not all data is uploaded to the C&C server. Instead, Flame initially collects some preliminary information (metadata, sum- mary, header info, etc.) which is sent to the attack center for analysis. In the light of that information the attacker decides about which files are juicier and instruct the Flame client to steal data accordingly.
- Flame is the first Windows malware using bluetooth [19].When launched, this module enumerates devices around the infected machine and turns itself into a ”beacon”, that is, announces itself as a discoverable device. This bluetooth functionality allows the attacker to identify the victims social networks and to identify the victims physical location.
- Flame also contain modules which collects data to improve its escaping capabilities. Whenever Flame notices that Windows OS is issuing a message or launching a tool that is referencing one Flame file or component, adventcfg.ocx takes a screenshot which is then sent to the C&C server for analysis. Based on this analysis the attacker will add code to improve the existing version for the next update. This is one of the features that made Flame undetected for a long period of time (at least two years) [15].
- Shortly after its discovery by Kaspersky Lab, Flame went dark overnight [19]. In the last week of May 2012, the C&C servers of Flame sent an updated command to all infected systems to completely delete itself. SUICIDE is the module in charge of committing suicide.

B. Spreading Mechanisms

The main spreading mechansim of Flame is completely new and extremely sophisticated [20]. The main idea is to take advantage of the automatic Microsofts Windows update to infect other systems in the same network. This mechanism is implemented in the SNACK, MUNCH and GADGET modules. The first step is to carry out a WPAD (Web Proxy auto Discovery Protocol) Man-In-The-Middle (MITM) attack to redirect all victims traffic through the Flame infected machine. The next step is to intercept requests for Windows update. This is carried out by the MUNCH module. Once a Windows update request is intercepted, the GADGET module will prepare a fake update binary with the Flame installation file. It is widely known that Windows OS computers launch Windows update binaries without any restrictions provided that the update is genuine, that is, signed by a Microsoft certificate [18]. Hence for this attack to work, the Flame client should sign the fake update with a Microsoft certificate. The trick to be able to sign code with a Microsoft certificate is to leverage Microsoft Terminal Certificates. In a typical LAN, a client using Microsoft Terminal Services (or Remote Desktop Connction (RDP) to remotely access a Windows desktop needs a license. The licenses are typically issued by a Terminal Services Licensing Server (TSLS) which allows an enterprise to administrate and enforce licenses for connecting clients within its environment[21]. Before using the TSLS, it must be activated by contacting Microsoft. Microsoft issues a limited use certificate allowing only to verify the ownership of the TSLS. Flame designers managed to use the certificate to sign code using a flawed signing algorithm [22]. In June 3, 2012, Microsoft issued a security advisory [23] and the corresponding update to fix this issue by moving three certificates to the Untrusted Certificate Store making any code signed by them invalid. This is the first time that such certificate leveraging is used to spread malware.

C. Working Infrastructure and Execution

Two important features of Flame require infected machines to contact regularly a C&C server. First, the main goal of Flame is to steal data from victims. Infected machines need to upload regularly stolen data to the servers. Second, Flame is highly modular and constantly updated with new functionalities to gather data more efficiently and to improve

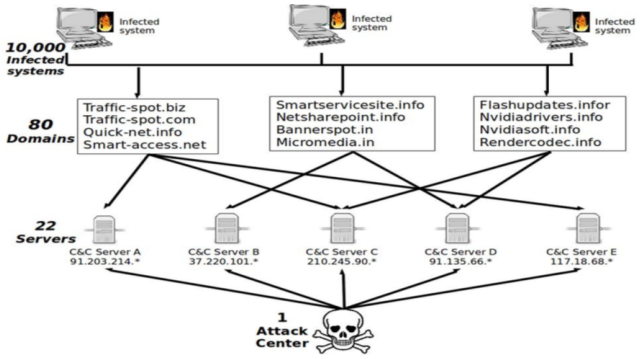

Figure 2. The Command and Control Platform behind Flame.

Fig. 2. The figure shows the overview of the working of Command and Control(CC) Centers of the Flame malware. [2].

its escaping capabilities. Infected machines need to contact regularly the servers for new updates and commands. Figure 2 shows how infected machine are connected to C&C servers and how these servers are controlled by the attack center. When a computer is infected with Flame, it uses a default configuration of 5 domains to contact the C&C servers. Once it successfully connects to a server, the list is updated to reach around 10 domains. In total, the infected machines use 80 domains to contact the C&C servers [20]. These domains are registered with fake identities (with fake addresses mostly in Germany and Austria) and with a variety of registrars. All used domains point to a total of 22 C&C server IPs hosted around the world [15]. These servers are hosted normally as any usual web server. The hosting companies providing the services are not aware of the activity of the servers. The attackers carefully configured the servers to look as any typical website server. These C&C servers were controlled by a single attack center [4].

As mentioned in the previous section, the SUICIDE module of Flame has been activated as soon as the first reports about the discovery of the malware are published (May 2012) [19]. However, according to analyzed samples, Flame clients (CLIENT TYPE FL) constitute only one out of four types of infected clients (CLIENT TYPE SP,CLIENT TYPE SPE, and CLIENT TYPE IP being the others) [9]. This indicates that the attackers behind Flame can deploy new variants anytime.

IV. Challenges in Detection

The ability to detect and classify malware provides a means to identify and stop the proliferation of malware across the network. Todays detection relies upon signatures of known malware to identify malware at the network and host [24]. Signature detection of advanced malware is complicated by polymorphism [25]. However, signature detection alone does not provide adequate protection for networks and end points from polymorphic malware. The McAfee Threat Report (2016) stated that on average the security saw almost 40 million new malware samples in each quarter for 2015. The main features for malware detection and prevention are security, safety, stealth and sustainability. These are explained briefly in this section.

1) **Security:** Security framework is used to analyse the platform to prevent the victim for attacks. Components of the framework have to be managed in a secure manner.
2) **Safety:** Various malicious files on the platform may cause unexpected incident. Especially for the dynamic analysis process, the safety framework captures malicious program and prevents any unwanted damage to network users. Particularly network access from the host that runs malware should be separated and should be controlled to prevent harmful network traffic.
3) **Stealth:** This framework is deployed on the Internet and is able to identify the existence or activities of the attacker. The behaviour of the system should not be very disrupting to minimize unnecessary risks. This program need to be handled with care especially on the network.
4) **Sustainability:** Counter malware activity is continuous process of keeping acquiring new specimens, storing acquired malware to database, analysis of specimen, generating signature for detection and reporting. As new malwares are created more frequently, this process cannot be stopped. The framework should be operable in a continuous manner.

Nikos et al. [26] highlight the potential issues that could have enabled the malware to evade detection by commonly used security technologies. All of these used Targeted operating system and architecture.

- All samples were targeting 32-bit versions of Windows. Interestingly enough, none of the malware would execute on 64bit systems. The additional security mechanisms that are available on the 64-bit versions of windows (especially Windows 7 and newer) complicate significantly the exploitation, especially when malware targets kernel components [27]. However, based on the fact that attackers had access to valid certificates (Stuxnet, Duqu) that they could use to sign the 64-bit components of their malware (thus allowed to execute in kernel space).
- For flame the initial infection method has not been identified, however infection through removable drives or spearfishing attacks, are realistic scenarios, as shown in Figure 3. Microsoft, since the release of Office 2010 Suite, has included protected view [26], a sandbox feature for opening office documents received from untrusted sources. By default, all new files are opened in protected mode, thus any potentially malicious payload would have to face the additional barrier of escaping the sandbox. As a result, we have to assume that victims who got infected were running outdated versions of Microsoft Office.
- All malware made use of exploits for command execution or privileged escalation, the vast majority of which were unknown to the security community (0-

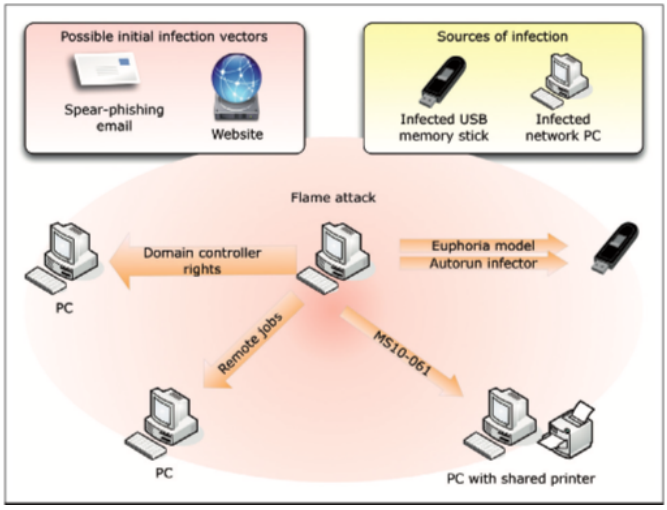

Fig. 3. Flame used multiple infection vectors which made it difficult to defend against [4].

day). However, the number of infected victims continued to increase, even when security patches addressing these vulnerabilities had been released, highlighting the lack of effective patch management processes.

- All malware communicated over ports 80/TCP, 443/TCP, or 22/TCP, as egress traffic destined to such ports is frequently allowed to pass through network access control mechanisms. Malware's success to communicate back to the C&C infrastructure highlights the fact that most of the victims had very relaxed internet access restrictions in place (if any) - a worrying finding, taking into account the sensitive nature of the targeted organizations.
- Flame was designed to detect and evade Antivirus Software, by having a list of different evasion techniques. Moreover it encrypted or obfuscated their network traffic, to and from the C&C servers, so as to pass under the radar of Network Intrusion Detection Systems (NIDS). The shortcomings of such systems, mainly due to the strong reliance on pattern matching (signature based detection) are well known to the research community for long [28], [29] nevertheless, they are most common (if not the only one) set of tools that defenders have.
- Flame, Stuxnet and Duqu binaries were digitally signed using compromised digital certificates. Thus, these samples would manage to infect hardened systems, where only digitally signed binaries were allowed to execute. Based on these findings, allowing execution of binaries based on the existence of valid digital signatures cannot be considered an effective defense on its own. This is particularly important for Antivirus and HIPS solutions, which tend to avoid real-time analysis of signed binaries for performance reasons.

1) Some Novel Detection Mechanisms: For malwares with the characteristics described in the previous section which are especially tailored to fool traditional detection systems require non-trivial detection techniques. An interesting method to reveal fraudulent process behavior (code injection) is the identification of increased usage of resources of certain processes with respect to a clean system. Context switch delta is one such metric that counts the number of context switches per refresh interval [26]. Other resource usage differences can be observed by comparing the number of threads or the referenced handles in the selected processes. In this way, the transmitted data will be secure and available for the user without compromising security. Behaviour analysis can be applied to classify some legitimate activity as suspicious or malicious. One technique for analysing the behaviour of a program is to study the sequence of operating system calls it makes [30]. Antivirus software can intercept these API calls while a program is running, and use heuristics to look for suspicious activity, terminating those with harmful behaviour. Various heuristics have been researched, such as looking for patterns used for self replication, but these all rely on monitoring a program once it is running. This is dangerous to rely on because the malware might cause harm to the system before it is recognized as malicious.

A basic problem with anomaly detection based approaches is that they may generate false alarms, and it is difficult to filter those false positives. More work on white listing techniques and collaborative information sharing may improve the situation. Academic research could contribute a lot in this area, because the problems require new, innovative solutions. We should also mention that, in some application areas, false positives may be better tolerated, because false negatives (i.e., missing a real attack) have devastating consequences. In particular, we believe that in critical infrastructures (such as nuclear power plants, chemical factories, certain transportation systems) where a successful logical attack on the connected IT infrastructure may lead to a fatal physical accident, false positives should be tolerated, and there should be expert personnel available to handle them.

A major reason for targeted attacks to be so successful is that the information available to the attackers and the defenders of systems is highly asymmetric: the attackers can obtain the commercially available security products, and fine tune their attack until these products do not detect it, while defenders have much less information about the methods and tools used by the attackers. One challenge is, thus, to break, or at least to decrease this asymmetry either by individualizing security products and keeping their custom configuration secret, or by gathering intelligence on the attackers and trying to predict their methods [26]. The second approach seems to be more difficult, and due to its slightly offensive nature, also morally questionable. Therefore, we argue that the (white hat) defensive security community should focus more on the first approach. One specific example in this vein would be the more widespread usage of honeypots and traps [31].

Other more formal techniques include pushdown model-

checking for malware detection[32], HTTP user-agent discrepancy identification[33] but these techniques require in depth offline analysis of the binaries of malware which of course cannot be done in real time.

V. Mitigation

Once a high-profile malware incident is detected, the most important tasks are (a) to contain the threat and recover from the incident locally, and (b) to disseminate the intelligence information to mitigate the global effects of the malware [34]. However, the problem is that once the victim of a malware attack has recovered from the incident, it has no incentive anymore to share information and forensics data. On the contrary, it prefers avoiding information sharing in order to preserve its reputation and the privacy of its sensitive data. Thus, again, the problem is related to misaligned incentives. Anecdotal evidence suggests that security vendors are often unable to obtain forensics information even if their own product detected the infection [34]. This lack of information sharing has negative consequences, as it practically hinders forensic analysis and efficient global response. For example, in case of Flame and Gauss, no dropper component has ever been made public, and it is possible that no dropper was identified at all, due to the lack of sharing forensic information by the victims. Consequently, the vulnerabilities exploited by the dropper of these pieces of malware are still unknown, and hence, they cannot be patched, letting the attackers continue using them in future attacks.

Taking the reactive approach is incredibly dangerous, because this model allows malicious code to infiltrate networks. Once its got through, todays complex malware and espionage codes can perform any number of functions, including self-replication and further infections. And so a reactive approach chasing down malware after its already infected a system means the damage could already be done. In the case of advanced malware, highly valuable intellectual property can be obtained within seconds of gaining system access. A fix is a fix, but your data has already been breached. The bottom line: chasing Flame or any advanced persistent threat or cyber- espionage tool after it has infected a network is futile: the damage may already be done [4].

The solution is to take a proactive approach rather than a reactive defence, which is precisely the concept behind application control and whitelisting security solutions. Using a proactive, default deny approach, malware like Flame would never gain access to a network, no matter how authentic it appears on the surface [4], [26]. Those valid Microsoft security certificates mean nothing in terms of application control. Herein lies the fundamental difference between a reactive and a proactive security system: enterprise networks utilising an application whitelisting program would not have allowed the installation of the Flame file (WUSETUPV.EXE), whether a valid security certificate is present or not. Instead, the attempted attack files would be held for review by the companys IT department, which must approve all programs and applications prior to any files or code gaining access to any machine on the companys network. In the case of Flame,

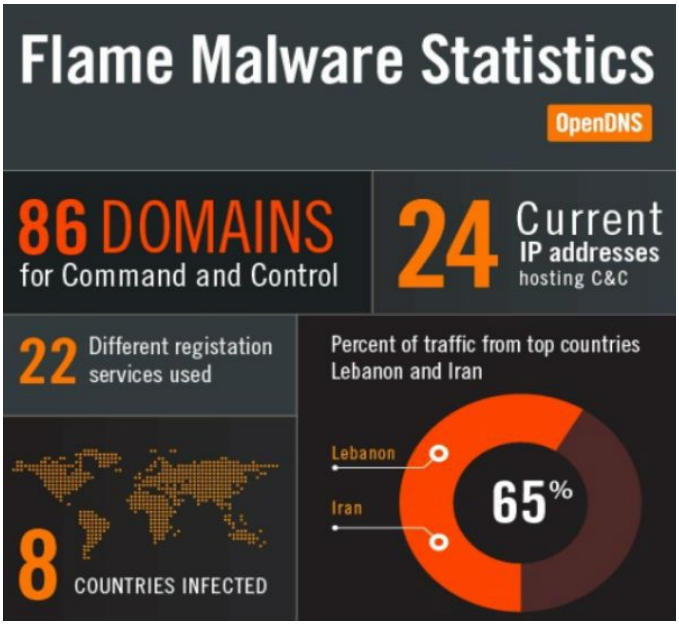

Fig. 4. Flame statistics chart as described by Kaspersky Labs [15], [35], [4].

the same code or multiple formats of malicious code could repeatedly attempt to gain access over the course of months or years with no success.

Outdated anti-virus(AV) software doesn't offer the total customisation capabilities possible with application whitelisting, and its not capable of stopping advanced, sophisticated malware and zero-day attacks. An effective whitelisting solution, however, will prevent all such attacks, especially in the case of advanced customisation and management. Its good to understand that in security, a layered approach is advisable. There is no silver bullet. But when it comes to endpoint security, AV is no longer sufficient. There is no need to supplement application control and whitelisting with a blacklisting program.

A. The Final Takeaway

If theres one thing that we should be taking away from Flame, its that we should never make assumptions when it comes to cyber-criminal capabilities. What we assumed in the past for instance, that only data files are at risk in the case of an attack is clearly obsolete. Flame has the capability to monitor nearly every activity on an endpoint, including conversations utilising Bluetooth technology. And as already discussed, Flames operators were extremely precise and targeted in their approach, knowing exactly how to get to just the right endpoints to obtain the necessary information and simply bypassing the rest. Why risk getting caught by being careless? Flame demonstrates just how organised, advanced and focused cybercrime can be. So if we cant predict what advances cyber-criminals will make in the coming months or years, how can we rely on a security system that does little to prevent attacks from unknown threats?

VI. Impact

According to estimates by Kaspersky in May 2012, Flame had initially infected approximately 1,000 machines, with victims including governmental organizations, educational institutions and private individuals [36], [5]. At that time 65% of the infections happened in Iran, Israel, Palestine, Sudan, Syria, Lebanon, Saudi Arabia, and Egypt with a "huge majority of targets" within Iran. Flame has also been reported in Europe and North America [20], [36], [4]. Figure 4 outlines the Flame malware statistics as estimated by various sources. The actual impact in monetary terms is extremely challenging to ascertain because of the unique objective of the malware.

VII. Conclusion

Detection of modern, unknown malware is an exceedingly important problem today to solve. This argument is especially true when targeted samples are under consideration because they violate all three founding properties of a secure system namely confidentiality, integrity and availability. In this paper we presented the analysis and dissection of one of the most sophisticated targeted malware: Flame. We looked at it's unique and unusual characteristics as a malware from it's large payload size to it's architecture and execution mechanism. We explored the security vulnerability of the targeted operating systems namely the flawed Digital Security Certificate signing of Microsoft Windows which made the attack so effective. We also outlined some of the major challenges in detecting malwares such as Flame and reviewed some novel and non trivial techniques to avoid them. We explored why mitigating such targeted cyber-attacks is not possible in a true sense and provided some heuristics which may prove to be helpful. Finally we presented some statistics outlining the estimated impact of the malware. We also learned that malware code obfuscation is an effective method used by majority of malware authors to avoid detection and harden reverse engineering analysis to hide the true nature of their code from researchers.

References

[1] James P Farwell and Rafal Rohozinski. Stuxnet and the future of cyber war. *Survival*, 53(1):23–40, 2011.

[2] Sami Zhioua. The middle east under malware attack dissecting cyber weapons. In *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, pages 11–16. IEEE, 2013.

[3] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Márk Félegyházi. Duqu: Analysis, detection, and lessons learned. In *ACM European Workshop on System Security (EuroSec)*, volume 2012, 2012.

[4] Kate Munro. Deconstructing flame: the limitations of traditional defences. *Computer Fraud & Security*, 2012(10):8–11, 2012.

[5] Eugene Kaspersky. Cyberscary: 4 digital threats to worry about. *Foreign Policy*, (197):78–78, 2012.

[6] Seth Hardy, Masashi Crete-Nishihata, Katharine Kleemola, Adam Senft, Byron Sonne, Greg Wiseman, Phillipa Gill, and Ronald J Deibert. Targeted threat index: Characterizing and quantifying politically-motivated targeted malware. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 527–541, 2014.

[7] Ravish Goyal, Suren Sharma, Savitri Bevinakoppa, and Paul Watters. Obfuscation of stuxnet and flame malware. *Latest Trends in Applied Informatics and Computing*, pages 150–54, 2012.

[8] Gaute Wangen. The role of malware in reported cyber espionage: a review of the impact and mechanism. *Information*, 6(2):183–211, 2015.

[9] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Mark Felegyhazi. The cousins of stuxnet: Duqu, flame, and gauss. *Future Internet*, 4(4):971–1003, 2012.

[10] Aditya Sood and Richard Enbody. *Targeted cyber attacks: multi-staged attacks driven by exploits and malware*. Syngress, 2014.

[11] Brett Stone-Gross, Christo Wilson, Kevin Almeroth, Elizabeth Belding, Heather Zheng, and Konstantina Papagiannaki. Malware in ieee 802.11 wireless networks. In *International Conference on Passive and Active Network Measurement*, pages 222–231. Springer, 2008.

[12] Aleksandr Matrosov, E Rodionov, D Harley, and J Malcho. Stuxnet under the microscope revision 1.1. eset. 2010, 2011.

[13] Steven Cherry and R Langner. How stuxnet is rewriting the cyberterrorism playbook. *Computerworld*, 2010.

[14] Gavin McWilliams. Malware detection methods for fixed and mobile networks. *Centre for Secure Information TechnologiesQueen's University Belfast*, pages 1–21, 2013.

[15] Oleg Demidov and Maxim Simonenko. Flame in cyberspace. *Security Index: A Russian Journal on International Security*, 19(1):69–72, 2013.

[16] Osamah L Barakat, Shaiful J Hashim, Raja Syamsul Azmir B Raja Abdullah, Abdul Rahman Ramli, Fazirulhisyam Hashim, Khairulmizam Samsudin, and Mahmud Ab Rahman. Malware analysis performance enhancement using cloud computing. *Journal of Computer Virology and Hacking Techniques*, 10(1):1–10, 2014.

[17] Marc Stevens. Fast collision attack on md5. *IACR Cryptology ePrint Archive*, 2006:104, 2006.

[18] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5(6):29, 2011.

[19] Symantec Tech Report June. Flamer: Urgent suicide. https://www.symantec.com/connect/blogs/flamer-urgent-suicide, June 29 2012. Accessed Feb 12,2019.

[20] A Gostev. Flame: Replication via windows update mitm proxy server. *Kaspersky*, 2012.

[21] Marc Stevens. Technical background on the flame collision attack. *CWI (Centrum Wiskunde & Informatica) News*, 7, 2012.

[22] Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P Smart. Flaws in applying proof methodologies to signature schemes. In *Annual International Cryptology Conference*, pages 93–110. Springer, 2002.

[23] Microsoft. Microsoft security advisory (2718704). https://support.microsoft.com/en-us/help/2718704/unauthorized-digital-certificates-could-allow-spoofing, June 29 2012. Accessed Feb 12,2019.

[24] Silvio Cesare and Yang Xiang. Malware variant detection using similarity search over sets of control flow graphs. In *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 181–189. IEEE, 2011.

[25] Fahad Bin Muhaya, Muhammad Khurram Khan, and Yang Xiang. Polymorphic malware detection using hierarchical hidden markov model. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 151–155. IEEE, 2011.

[26] Nikos Virvilis and Dimitris Gritzalis. The big four-what we did wrong in advanced persistent threat detection? In *2013 International Conference on Availability, Reliability and Security*, pages 248–254. IEEE, 2013.

[27] Scott Field. An introduction to kernel patch protection, 2006.

[28] Fred Cohen. Computer viruses: theory and experiments. *Computers & security*, 6(1):22–35, 1987.

[29] Michel Denault, Dimitris Karagiannis, Dimitris Gritzalis, and Paul Spirakis. Intrusion detection: approach and performance issues of the securenet system. *Computers & Security*, 13(6):495–508, 1994.

[30] Ammar AE Elhadi, Mohd A Maarof, and Ahmed H Osman. Malware detection based on hybrid signature behaviour application programming interface call graph. *American Journal of Applied Sciences*, 9(3):283, 2012.

[31] Niels Provos and Thorsten Holz. *Virtual honeypots: from botnet tracking to intrusion detection*. Pearson Education, 2007.

[32] Fu Song and Tayssir Touili. Pommade: pushdown model-checking for malware detection. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 607–610. ACM, 2013.

[33] Martin Grill and Martin Rehák. Malware detection using http user-agent discrepancy identification. In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 221–226. IEEE, 2014.

[34] Panos Kampanakis. Security automation and threat information-sharing options. *IEEE Security & Privacy*, 12(5):42–51, 2014.

[35] Ellen Nakashima, Greg Miller, and Julie Tate. Us, israel developed flame computer virus to slow iranian nuclear efforts, officials say. *The Washington Post*, 19, 2012.

[36] Kim Zetter. Meet flame–the massive spy malware infiltrating iranian computers. *Wired, 28th May,{Online resource} Available at: https://www. securelist. com/en/blog/208193522/The_Flame_Questions_and_Answers,[Accessed 20/11/2012]*, 2012.