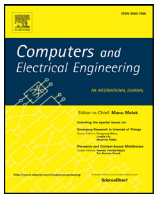
Computers and Electrical Engineering 66 (2018) 353–368

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng



Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics☆

Krzysztof Cabaj[a], Marcin Gregorczyk[b], Wojciech Mazurczyk[b,*]

[a] Warsaw University of Technology, Institute of Computer Science, Warsaw, Poland
[b] Warsaw University of Technology, Institute of Telecommunications, Warsaw, Poland

ARTICLE INFO

Article history:
Received 23 November 2016
Revised 16 October 2017
Accepted 17 October 2017
Available online 27 October 2017

Keywords:
Ransomware
Malware
Software-defined networking
Network security

ABSTRACT

Ransomware is currently one of the key threats facing individuals and corporate Internet users. Especially dangerous is crypto ransomware that encrypts important user data, and it is only possible to recover it once a ransom has been paid. Therefore, devising efficient and effective countermeasures is a pressing necessity. In this paper we present a novel Software-Defined Networking (SDN) based detection approach that utilizes the characteristics of the ransomware communication. Based on an observation of network communication between two crypto ransomware families, namely CryptoWall and Locky, we conclude that an analysis of the HTTP message sequences and their respective content sizes is enough to detect such threats. We show the feasibility of our approach by designing and evaluating a proof-of-concept SDN-based detection system. The experimental results confirm that the proposed approach is feasible and efficient.

1. Introduction

The year 2016 was named “the year of the ransomware” by the mass media, and this type of threat is currently considered by the security community and law enforcement agencies (e.g., Europol’s recent “2016 Internet Organized Crime Threat Assessment” report [1]) as a key threat to Internet users. Ransomware is a type of malicious software that is designed for direct revenue generation and which after infection holds the victim’s machine or user’s critical data “hostage” until a payment is made. Ransomware developers are constantly improving their “products” making it harder to design and develop effective and long-lasting countermeasures. Considering the fact that more and more devices are foreseen to be connected to the Internet due to the Internet of Things (IoT) paradigm, it makes it a perfect environment for ransomware to spread in the foreseeable future [2]. The ransomware plague is so widely spread that there are even crime-as-a-service tools available in the dark web (like TOX ransomware-construction kit [3]) that allow even inexperienced cybercriminals to create their own customized malware, to manage infections, and profits.

There are two main types of modern ransomware, i.e., locker and crypto. The infection for both kinds of malicious software happens in a similar way, i.e., a user machine is infected by means of various attack vectors, e.g., by drive-by-download, malvertisement, phising, spam, or different forms of social engineering, etc. However, what comes after the infection is different for both types. *Locker ransomware* denies the user access to an infected machine but typically the underlying system and files are left untouched. On the other hand, *crypto ransomware* is a kind of data locker that prevents the user from

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. Zhihan Lu.
* Corresponding author.
E-mail addresses: kcabaj@ii.pw.edu.pl (K. Cabaj), m.gregorczyk@tele.pw.edu.pl (M. Gregorczyk), w.mazurczyk@tele.pw.edu.pl (W. Mazurczyk).

https://doi.org/10.1016/j.compeleceng.2017.10.012
0045-7906/© 2017 Elsevier Ltd. All rights reserved.

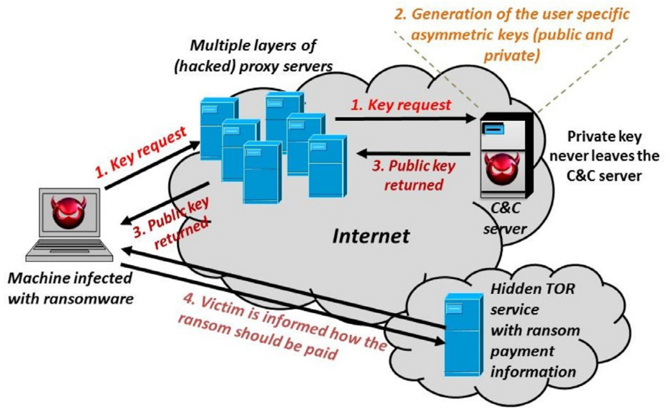

2. Generation of the user specific asymmetric keys (public and private)
Multiple layers of (hacked) proxy servers
1. Key request
1. Key request
3. Public key returned
3. Public key returned
Private key never leaves the C&C server
C&C server
Internet
Machine infected with ransomware
4. Victim is informed how the ransom should be paid
Hidden TOR service with ransom payment information

Fig. 1. Typical asymmetric key cryptography-based ransomware scheme.

accessing her/his vital files or data (e.g., documents, pictures, videos, etc.) by using some form of encryption. Therefore, attacked files are useless until a ransom is paid and the decryption key is obtained. Then after the user's machine is locked or data is encrypted the victim is presented with an extortion message. In many cases, paying the ransom to the cybercriminal is the only way to get back access to the machine/data. The value of the requested ransom differs and is typically in the range US$300–$700, and the favored payment currency is bitcoins [2]. It must be emphasized that not only individual users are currently targeted, but also companies and institutions like hospitals and law enforcement agencies, etc. Clearly, effective and efficient solutions to counter ransomware infections are desired.

Although the first cases of crypto ransomware have been known for more than 10 years (e.g., Trojan.Gpcoder), it must be emphasized that the recent plague (Symantec reported an astounding 4000% rise in crypto ransomware incidents in 2014 [4]) of this type of malware is related to the improved design of the cybercriminals' "products". The main difference now is that crypto ransomware has moved from custom or symmetric key to asymmetric key cryptography (Fig. 1). In this case, when the machine is infected, it contacts the C&C (Command & Control) server through multiple proxy servers (which are typically legitimate but hacked machines) to request a public encryption key. At the C&C a pair of matching public-private keys is generated for each infection and the public key is returned to the compromised host (the private key never leaves the C&C server). Then the public key is used to securely transfer the session key to encrypt the chosen files, which are deemed the most important for the user. It is worth noting that if correctly implemented, asymmetric crypto ransomware is (practically) impossible to break. The most prominent ransomware, and one of the first to introduce asymmetric key cryptography, is CryptoWall 3.0, which was discovered at the beginning of 2015, and later followed by others, such as CryptoWall 4.0 and Locky, etc.

Software-defined networking (SDN) is one of the emerging networking paradigms [5]. Its main benefit is that it allows the decoupling of the control and data planes, i.e., the underlying network infrastructure is abstracted from the applications. Therefore, the network can be managed in a logically centralized way. Apart from many potential applications [5], recently SDN has become a promising opportunity to provide security for current networks in a more flexible and effective manner [6]. Moreover, currently, SDN serves as the base of cloud computing environments in public, private, and hybrid clouds, and it is envisioned to become a core of future network services. The majority of network device manufactures are supporting SDN with their physical and virtual equipment using the OpenFlow protocol. SDN is standardizing the management of heterogeneous networks. Applications written for the SDN controller will work without additional adjustments to different devices supporting SDN in both physical and virtual environments. These are the main reasons why SDN was chosen as the key technology for our solution.

Taking the above into account, in this paper we present a dedicated SDN-based system for ransomware detection and mitigation. It must be noted that when it is possible to successfully discover ransomware communication, then, obviously, sometimes, it may be too late to prevent encryption for that particular victim. However, it is still possible to utilize this incident to provide feedback for the detection system to "save" other users. This phenomenon is well-known in nature where a single organism often has to make a self-sacrifice for the sake of the group [7]. As more and more analogies between cybersecurity and nature are continuously drawn [8], this may be another opportunity to reuse natural experiences in this regard to improve communication networks' defenses.

The proposed detection system, in this paper, has a focus on the crypto ransomware that utilizes asymmetric cryptography. While designing and developing the system we took into consideration results from the traffic analysis of two modern ransomware families, i.e., CryptoWall and Locky. Based on the performed analyses we concluded that these ransomware families share some similarities, which can be utilized to create an efficient and effective detection system. Thus, the main contributions of the paper are:

- Use of the SDN-based architecture to mitigate crypto ransomware, which allows the creation of a flexible and effective detection and prevention system.
- Presentation of results of the network measurements-based behavioral analysis of two ransomware families, namely CryptoWall and Locky.
- Design of the SDN-based detection and mitigation system that relies on our findings from the performed behavioral analyses mentioned above.
- Development and evaluation of the proof-of-concept implementation of the proposed detection system.

The rest of this paper is organized as follows: first, we present existing work related to malware detection using the characteristics of the HTTP traffic, SDN-based approaches to cyber threat detection and prevention including ransomware, and, in particular, existing ransomware detection methods. In Section 3 we present the results from the behavioral analyses of two ransomware families. In the next section, a SDN-based ransomware detection system that relies on ransomware HTTP traffic characteristics is described and evaluated. Finally, Section 5 concludes our work.

2. Related work

In this section we review existing work, first related to ransomware detection and then SDN-based threat detection with special emphasis for ransomware countermeasures. Moreover, we analyse methods for discovering threats based on HTTP protocol traffic analysis.

Malware detection has been extensively studied in recent years. In general, such techniques can be broadly divided depending on where the malware activities are observed, either at the network-level [9,10], system-level [11,12], or both [13]. In this paper we propose a network-level SDN-based solution.

From the functional perspective, still the most common approach to malware detection is payload-based, i.e., these approaches are only effective if the malware communication is invariant [14]. Thus, if the malicious software is using plaintext communication protocol, then such invariants may exist (e.g., protocol keywords can serve as a part of the payload signatures). The same situation can be experienced for several encrypted malware communication protocols, where certain invariant plaintext fragments result in certain invariant encrypted data (if encryption keys are not wisely used), which can be easily applied as a signature as well [15]. However, often various malware families (including ransomware) do not exhibit the characteristics indicated above, which means they can circumvent payload signature-based detection systems [16]. Recently, a novel approach was proposed that uses tamper resistant features of the transport layer protocol to distinguish malware heartbeat messages (that sustain the connection with the C&C) from the legitimate traffic. However, the authors noted that they observed a substantial decrease in the detection of malicious software in which traffic is disguised in HTTP messages [17].

When it comes specifically to countering ransomware several works exist. In [18] the author proposed a Heldroid system that employs static taint analysis together with lightweight symbolic execution to find code paths that indicate device-locking activity or attempts to encrypt files on external media. The authors of [19] describe the results from a long-term study of ransomware between 2006 and 2014. Based on the gathered data, they concluded that the number of ransomware families with sophisticated destructive capabilities remains quite small. They also proposed a detection system that is based on the monitoring of abnormal file system activity. Scaife et al. [20] introduced an early-warning detection system for ransomware that monitors all file activities and alerts the user in case something suspicious is identified using a union of three features, i.e., file type changes, similarity measurement, and entropy. Another recent work [21] introduced EldeRan, which is a machine learning approach for dynamically analysing and classifying ransomware. The proposed solution monitors a set of actions performed by applications in their first phases of installation and tries to detect the characteristics signs of ransomware. The obtained experimental results proved that this approach is effective and efficient, which also shows that dynamic analysis can support ransomware detection by utilizing the set of characteristic features at run-time that are common across families, and that helps the early detection of new variants.

When it comes to SDN-based solutions tailored to security purposes, the first work that proposed a general SDN-based anomaly detection system was put forward by Mehdi et al. in 2011 [6]. The authors showed how four traffic anomaly detection algorithms could be implemented in an SDN context using OpenFlow compliant switches and NOX controller. The obtained experimental results proved that these algorithms are significantly more accurate in identifying malicious activities in home networks when compared to the ISP. Further, other researchers utilized SDN to detect network attacks [22] or to monitor dynamic cloud networks [23].

Several recently published papers deal with SDN-based malware detection. Jin and Wang [24] analyzed malicious software behaviors on mobile devices. Based on the acquired knowledge they proposed several mobile malware detection algorithms, and implemented them using SDN. Their system was able to perform real-time traffic analysis and to detect malicious activities based only on the connection establishment packets. In [25] the authors designed and developed an SDN-based architecture specialized in malware analysis aimed at dynamically modifying the network environment based on malicious software actions. They demonstrated that this approach is able to trigger more malware events than traditional solutions. To the best of our knowledge, no works, so far, have proposed ransomware detection using a SDN-based system.

HTTP traffic characteristics have been already used as a recognition property for detecting malware. For instance, a characteristic combination of HTTP URI request parameters in the malware communication may be utilized to discover malicious

Table 1
Network measurement statistics for CryptoWall 4.0 and Locky ransomware.

Ransomware type	No. of samples	No. of sample executions	Size of traffic traces
CryptoWall 3.0/4.0	359 (331 for CW3.0 and 28 for CW4.0)	3700	≈5 GB
Locky	428	2200	≈330 MB

communication. For example, Perdisci et al. showed that clustering HTTP traffic can be used to extract behavioral features that can be used to recognize HTTP-based malware [26]. In [27] Zegers investigated whether the order of HTTP request headers can be used to recognize malware communication. However, the author states that different websites have their own header order. This is similar to applications that communicate over HTTP (Windows updates and anti-viruses). Therefore, the conclusion is that it is unfeasible to use the HTTP headers' order to reliably identify malicious software. Kheir [28] also analysed User Agent (UA) anomalies within malware HTTP traffic to extract the signatures for its detection. His observation was that almost one malware out of eight uses a suspicious UA header in at least one HTTP request (this includes typos, information leakage, outdated versions, and attack vectors such as XSS and SQL injection). Based on the above, the authors developed a new classification technique to discover malware user agent anomalies. The obtained experimental results showed that this approach considerably increases the malware detection rate. A similar approach was proposed in [29], where to detect malware the authors took advantage of statistical information about the usage of the UA of each user together with the usage of a particular UA across the whole analysed network and typically visited domains. However, it must be noted that, so far, no approach exists that takes into account the sequences of HTTP messages and their respective sizes as the main feature for the detection system, as we propose in this paper.

The first dedicated SDN-based ransomware security solutions designed to improve the protection against ransomware was recently proposed in [30]. The authors introduced two approaches that took advantage of the fact that without successful communication to the C&C server the infected host is not able to retrieve the public key and, as a result, it cannot start the encryption process. Both methods rely on dynamic blacklisting of the proxy servers used to relay communication between an infected machine and the C&C server. However, the main drawback of the proposed approaches is that they are efficient only when the ransomware proxy servers have been previously identified by means of behavioral analysis of known malware samples. Therefore, it is not possible to discover new campaigns when they first appear.

Due to this fact, in the paper, we take a different angle, i.e., we utilize the characteristics of the network communication between the infected host and a proxy server to detect the ransomware data exchange. As we observe, based on two "popular" ransomware families, the communication protocol utilized in the analyzed malware samples are similar; therefore, the utilization of this knowledge can lead to the development of an efficient and effective ransomware countermeasure. Such an approach can also be treated as a complementary solution to the detection methods proposed in [30], as it can form a source of feedback for the dynamic blacklisting of the proxy servers.

3. Crypto ransomware traffic characteristics based on CryptoWall and Locky families

Based on our experiences with crypto ransomware, we decided to choose two families, namely CryptoWall and Locky, to present their communication characteristics and to illustrate different levels of malware sophistication. These two ransomware examples utilize asymmetric cryptography; however, they differ when it comes to the network part of the communication, although they both use HTTP protocol. That is why, in the following subsections, we describe both ransomware families' communication characteristics in detail. Samples for further analysis were gathered from a freely available source, mainly the malwr.com service. Gathered samples were later evaluated using the Maltester dynamic analysis environment [31] developed at the Warsaw University of Technology. Table 1 summarizes our measurement efforts for the two malware families indicated above.

3.1. CryptoWall communication characteristics

CryptoWall 4.0 has been active since October 2015, and it superseded the previous 3.0 version. From the network traffic perspective, to communicate with the C&C server CryptoWall uses domain names instead of direct IP addresses. An analysis of the traffic from infected machines revealed that CryptoWall communication is based on HTTP POST messages. The communication is directed to the scripts uploaded onto the hacked web servers (proxy servers) and it is encrypted using the RC4 algorithm. However, the encryption key is very easy to retrieve as it is incorporated within the HTTP request.

The CryptoWall communication is depicted in Fig. 2. During the first data exchange, the ransomware reports its unique identifier and the victim's IP address to the C&C, which acknowledges the received information. In the second exchange, the response contains an image containing instructions for the victim and TOR address of the ransom webpage, the victim's personal code and an RSA 2048-bit public key that is used for encrypting the data. Then the encryption process starts. Finally, during the last data exchange an acknowledgement for the public key reception is provided. It is worth noting that CryptoWall 4.0 does not report the end of the encryption process to the C&C and does not report the number of encrypted files, which was the case for its predecessor – CryptoWall 3.0.

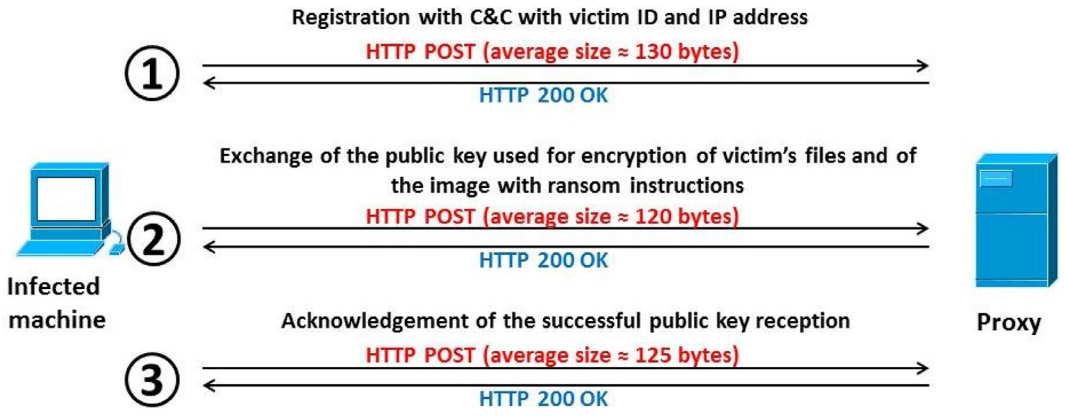

Registration with C&C with victim ID and IP address
HTTP POST (average size ≈ 130 bytes)
HTTP 200 OK
Exchange of the public key used for encryption of victim's files and of the image with ransom instructions
HTTP POST (average size ≈ 120 bytes)
HTTP 200 OK
Acknowledgement of the successful public key reception
HTTP POST (average size ≈ 125 bytes)
HTTP 200 OK
Infected machine
Proxy

Fig. 2. CryptoWall 4.0 communication.

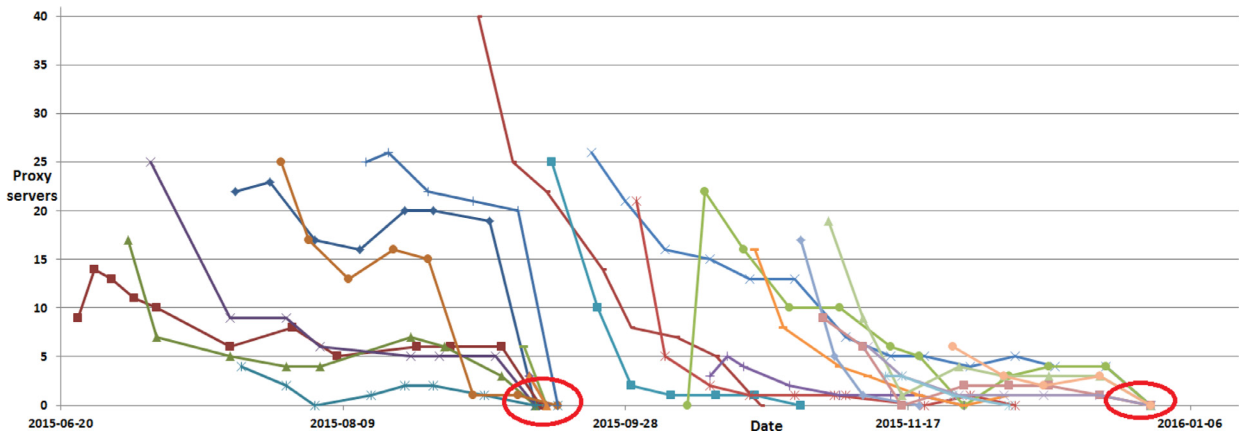


Fig. 3. CryptoWall 3.0 proxy servers' activity.

During our research we investigated a total of 359 CryptoWall samples, from which 28 were CryptoWall 4.0 and the rest were CryptoWall 3.0. Each CryptoWall sample contained a hardcoded list of proxy servers that is used during the transfer of the public key from the attacker C&C server. We also discovered that typically these servers are victims too. To execute the proxy script, the cybercriminals behind CryptoWall utilize compromised legitimate servers. When a new campaign of CryptoWall appears there are many samples with the same proxy list. Our analyses for CryptoWall 4.0 revealed 19 proxy lists. The average proxy list contained 47 server addresses (the shortest list had 27 and the longest 70 addresses). Using information about the servers in the proxy list we investigated how long these servers had been responsive. Figs. 3 and 4 illustrate the number of responsive servers in the detected proxy lists for CryptoWall 3.0 and 4.0, respectively. What should be emphasized is that the longest responding proxy server was active for as long as 11 weeks.

Two of the time instants observed in Fig. 3 are particularly interesting and are highlighted with red ovals. The first one is related to the massive shutdown of proxy servers and the immediate appearance of a few new samples with new proxy lists. The second represents the complete shutdown of the CryptoWall 3.0 infrastructure.

It is worth noting that at the time of the CryptoWall 3.0 shutdown there was a noticeable lack of CryptoWall 4.0 server activity (Fig. 4). It might be assumed that during this time the cybercriminals were performing (most probably) some kind of infrastructure update or maintenance. Another, more general conclusion is that typically the number of active servers from the proxy server list decreased over time.

3.2. Locky communication characteristics

Locky ransomware samples have been observed since mid-February 2016. Its communication patterns are similar to those of the CryptoWall family. The first resemblance is related to the utilization of HTTP POST messages. However, in this case the encryption scheme used to secure the data exchanged was better chosen. Thus, when this article was written, it was not possible to decrypt the Locky communication. Therefore, the information presented in this subsection is mostly deduced based on our previous experience and knowledge of ransomware. The communication in the case of Locky consists of four HTTP POST exchanges, which (most probably) serve the same aim as in the case of CryptoWall 3.0/4.0 (Fig. 2). For example,

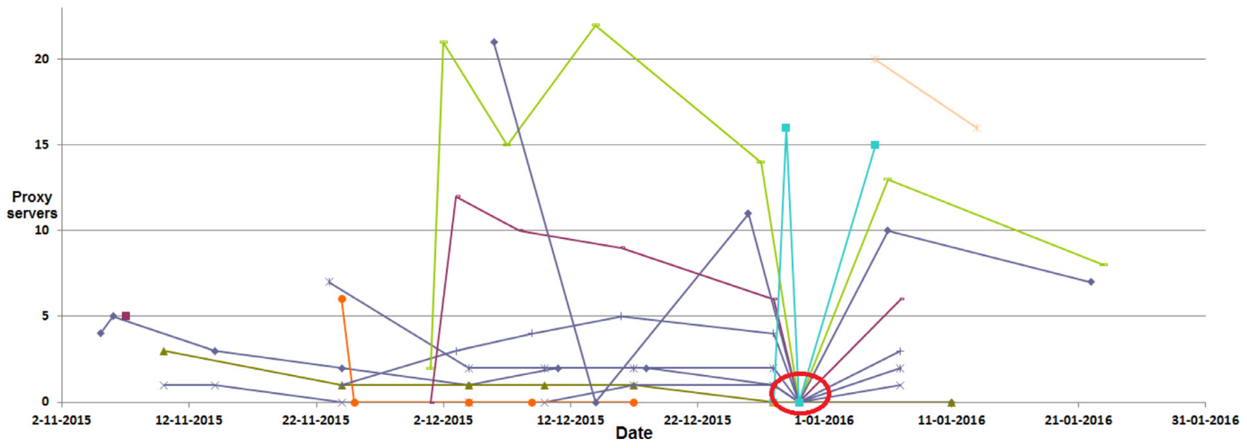


Fig. 4. CryptoWall 4.0 proxy servers' activity.

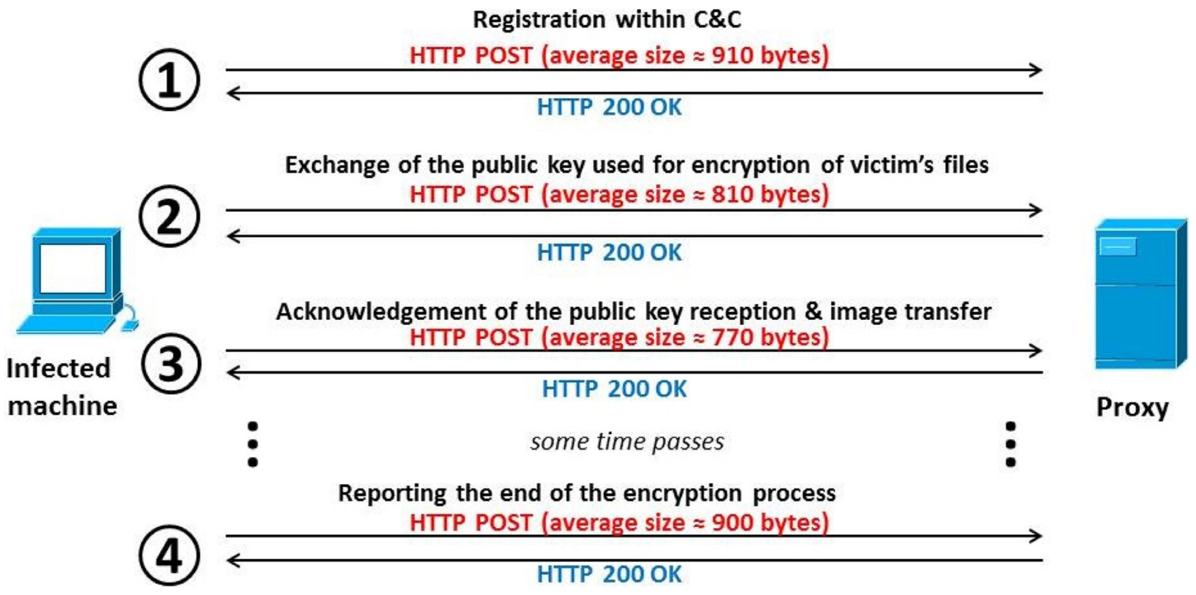

Registration within C&C
HTTP POST (average size ≈ 910 bytes)
HTTP 200 OK
Exchange of the public key used for encryption of victim's files
HTTP POST (average size ≈ 810 bytes)
HTTP 200 OK
Acknowledgement of the public key reception & image transfer
HTTP POST (average size ≈ 770 bytes)
HTTP 200 OK
some time passes
Reporting the end of the encryption process
HTTP POST (average size ≈ 900 bytes)
HTTP 200 OK
Infected machine
Proxy
1
2
3
4

Fig. 5. Locky (most probable) communication.

it is possible that the image transfer as well as the public encryption key transmission is performed during the third HTTP data exchange as the size of the resulting instructions presented to the user when extorting the ransom is of a similar size as the HTTP 200 OK response.

The other difference between Locky and CryptoWall is that the disruption of the communication when transmitting the public key in the case when CryptoWall was able to block the encryption process. However, Locky, in such cases, did not stop the encryption and used a hardcoded key (probably one dedicated key per sample). Additionally, in Locky the communication capabilities to contact the C&C server have been significantly extended. Each Locky sample has its own list of C&C IP addresses hardcoded. In the case that these addresses are already blocked it uses a DGA (Domain Generation Algorithm) to generate new C&C domains and tries to contact them. It is also worth noting that the first Locky samples generated exactly the same HTTP POST message sizes. However, Locky has evolved over time and currently the message sizes vary, as presented in Fig. 5.

We have been gathering Locky samples since the end of March 2016. Identified samples were executed in the Maltester dynamic analysis environment. Obtained results allowed characterizations of the Locky network behavior. During these more than eight months we observed various changes in Locky activities. The most interesting findings are further elaborated in this section. Until now, we observed three distinct versions of the Locky communication protocol. The first version used the same URL, i.e., main.php and in all executions the sizes of the messages sent to the C&C server were the same: 101, 55, and 94 bytes of (probably) encrypted binary data. At the beginning of April the protocol was changed, and then each execution of the ransomware sample resulted in randomly generated message sizes. This behavior is similar to the change

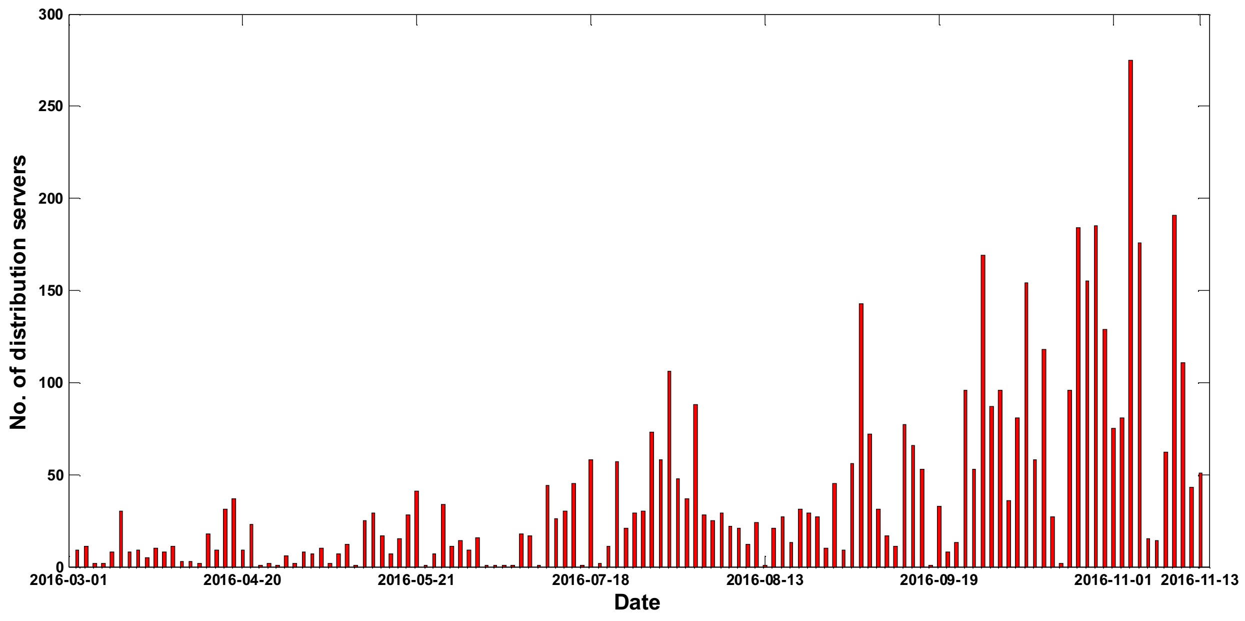


Fig. 6. Locky distribution servers' statistics (based on data from http://ransomwaretracker.abuse.ch).

Table 2
Statistics of the analyzed Locky samples.

C&C Server URL	Date of first sample analysis	No. of samples	No. of hardcoded C&C	No. of different DGA algorithms	DLL entry point name
main.php	2016.03.21	41	18	7	–
submit.php	2016.03.28	24	15	3	–
userinfo.php	2016.05.03	226	42	9	–
access.cgi	2016.05.30	2	2	1	–
/upload/_dispatch.php	2016.05.31	18	14	9	–
/php/upload.php	2016.08.01	11	16	3	–
/data/info.php	2016.08.29	19	14	8	1
apache_handler.php	2016.09.27	20	22	8	1
linuxsucks.php	2016.10.24	9	11	5	2
message.php	2016.11.03	57	24	20	11
information.cgi	2016.11.21	1	3	1	1

observed in the CryptoWall family during the transition from version 3.0 to 4.0. The last observed modification to the communication protocol happened around the end of July, when the encoding of data using only ASCII printable characters was first observed. This Locky version has greater message sizes and was used during our experiments.

From March 2016 we also observed changes in how the malware was distributed. Initially, during the first stage, typically, a Microsoft Word or Excel document with embedded hostile macro was sent as SPAM, and later the main stage of the ransomware was downloaded from the distribution server and executed. Later, around the end of May, we started observing encrypted executables. The first stage macro in the document downloaded an encrypted file, decrypted it on the infected machine and executed it. Obviously, the file is hosted on the distribution server without the decryption key and cannot be executed. The number of distribution servers between March and November 2016 is illustrated in Fig. 6. It can be observed that till June 2016 the number of distribution servers discovered daily was typically not higher than 50; however, in the last months it doubled with a few spikes reaching more than 150. This means that Locky was still active and the cybercriminals behind it were still trying to reach as many victims as possible. At the time of finishing this article, after a week of silence, the first sample of a new campaign was discovered and analyzed.

The most eye-catching aspect of Locky are the URLs used for C&C communication. During our research we investigated 11 distinct URLs. We decided to use their names as indicators of the new Locky campaigns. Our analyses revealed that each campaign lasted no longer than one month. From version 3.0 of the communication protocol we observed various features of the gathered samples in more detail. We discovered, among others, hardcoded C&C servers, used DGAs (Domain Generation Algorithms), and entry function DLL names. All the analyzed details are summarized in Table 2. The provided information allows an illustration of the extent of the resources used by the cybercriminals' infrastructure.

The second column of Table 2 contains the date when we analyzed the first sample from a given campaign. We did our best to analyze samples as they appeared; however, sometimes when the attackers introduced more modifications to their code it took more time to analyze how the samples can be gathered. Therefore, the provided dates can be used only as an approximate time of the start of a new campaign. The fourth and fifth columns are associated with the various aspects of communication with the C&C server. As mentioned at the beginning of this section, Locky used more reliable C&C communication when compared to CryptoWall. Firstly, Locky tried to contact the hardcoded IP addresses of the C&C servers. Our analysis shows that each sample had from two to five such addresses hardcoded within its binary. During our research we observed that if the hardcoded C&C servers were shut down then the next samples were equipped with previously unseen IP addresses for malicious servers. Due to this fact we observed an average of almost 18 C&Cs per analyzed campaign (maximum of 42). In cases when all hardcoded C&C servers were not active, Locky switched to the second method in which it utilized DGA. What is interesting, is that samples from the same campaign used various DGA algorithms, which in effect led to various domains being generated during the same time frame. The number of used DGA algorithms is presented in the fifth column in Table 2. The sixth column contains the number of used Entry Functions. The first six campaigns used normal executables during the main stage of the ransomware infection. However, at the end of August, we noticed a change in the malware distribution technique. We discovered that the downloaded file was a DLL library. To execute malware in such a form additional information, i.e., the name of the entry function, is needed. The first two campaigns utilizing such a solution used the same, simple entry function with the name “qwerty”. However, the next campaigns switched to more complicated and often modified entry function names. Obviously, all such changes were introduced to make ransomware analysis more difficult and to increase the time needed for security professionals to understand its behavior.

4. SDN-based ransomware detection based on HTTP traffic characteristics

As mentioned in the previous section, all analyzed ransomware families utilize a custom protocol that is used for downloading the encryption key and image, etc. from the attacker’s C&C server. Due to the fact that the communication process is similar for both families, this characteristic feature can be used for ransomware infection detection. That is why we use this knowledge to introduce a novel network traffic classification method that is based solely on the size of the data inserted by the victim into the outgoing sequence of the HTTP POST messages.

4.1. Proposed detection method

The proposed scheme can be divided into three main phases (Fig 7). The first is a learning phase in which real ransomware samples and the network traffic generated by them were used to extract the characteristic features from the outgoing HTTP messages (particularly their size). During the second, fine-tuning phase we focus on adjusting the parameters of the detection method. Finally in the last, detection phase we utilize data gathered during the two previous phases and detect infections using the proposed SDN based solution. Later, in Section 4.4 we present the results of the evaluation of the introduced detection method using both malicious and benign network traffic. It must also be emphasized that for each ransomware family the described three-phase procedure is conducted separately. All the phases are described in detail in the following sections.

4.1.1. Learning phase

In this phase, the data used later for the detection purposes were prepared and preprocessed. During this process, ransomware samples for each ransomware family (f), which were accumulated (details on how this dataset was formed can be found in Section 4.3), were executed in the controlled environment and all traffic generated by the infected machine was captured for further preprocessing. For this purpose we utilized the Maltester environment as described in [31]. From the obtained traffic, initially, three HTTP POST messages sent by the malware were located and the respective content lengths were extracted. In the result, the kth infection ($k \in (1, n)$, where n is the total number of all ransomware samples for a given family) is characterized by a vector S_{fk} consisting of three numbers $[s_{1k}, s_{2k}, s_{3k}]$, with each representing the content size sent in the consecutive HTTP POST messages (POST triples vector). The set of all S_{fk} was treated as a base to establish the main distinguishing feature for detection purposes.

4.1.2. Fine-tuning phase

When the learning set of vectors S_{fk} was (separately) prepared for each ransomware family (f), then the centroid vector of the corresponding feature vectors (C_f), as well as the minimal and maximal Euclidean distances for all vectors from S_{fk} to C_f, were calculated. Then using C_f and datasets that consist of the HTTP traffic from new infections (not utilized during the learning phase) and the benign HTTP traffic (without infections), the limit distance (d_{Lf}) from the centroid value was fine-tuned (the exact fine-tuning procedure is explained later in the experimental section). During the fine-tuning phase, information related to the minimal and maximal distance to the centroid was utilized. As a result, at the end of this process all the necessary parameters used later during the detection phase were calculated. In particular, these were the three values to form the centroid vector C_f and the limit distance d_{Lf} that was used to assess whether the currently evaluated feature vector extracted from the monitored HTTP traffic was malicious or not (Fig. 8, left). It must also be emphasized that for each ransomware family there will be a separate centroid vector C_f as well as the limit distance d_{Lf} established. All details concerning the obtained experimental results are presented later in Section 4.4.

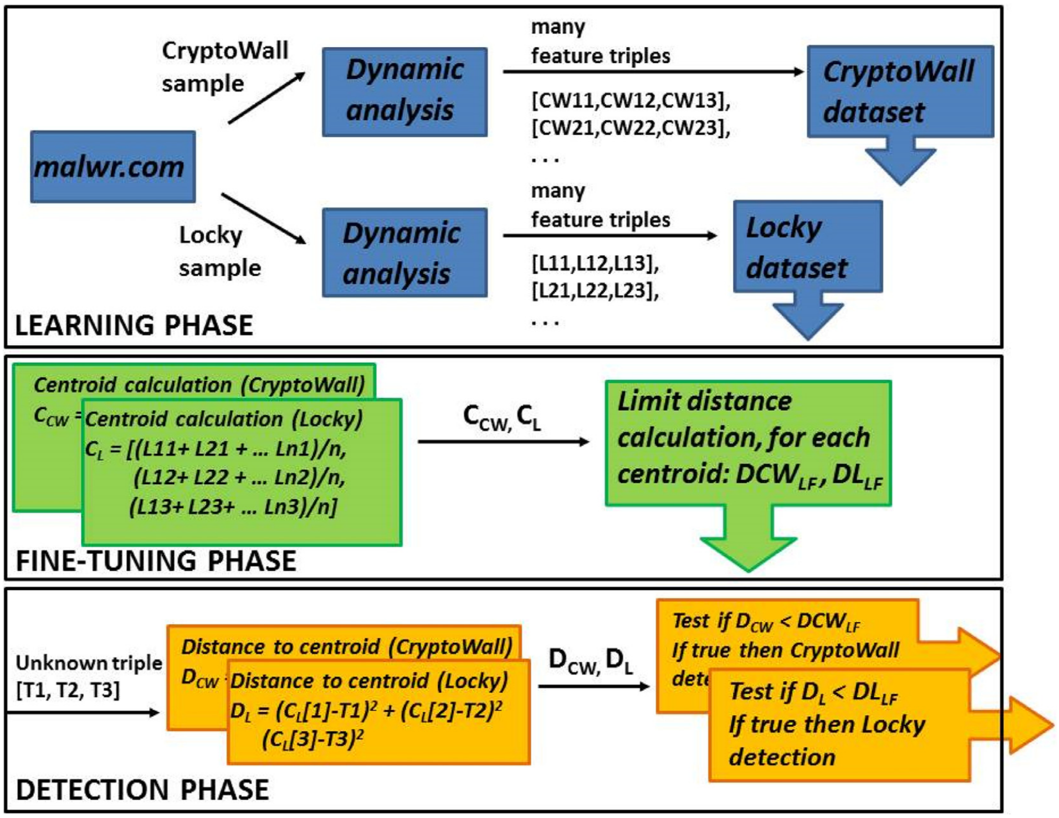

CryptoWall sample
Dynamic analysis
many feature triples
[CW11,CW12,CW13],
[CW21,CW22,CW23],
...
CryptoWall dataset
malwr.com
Locky sample
Dynamic analysis
many feature triples
[L11,L12,L13],
[L21,L22,L23],
...
Locky dataset
LEARNING PHASE
Centroid calculation (CryptoWall)
Centroid calculation (Locky)
C_L = [(L11+ L21 + ... Ln1)/n, (L12+ L22 + ... Ln2)/n, (L13+ L23+ ... Ln3)/n]
C_CW, C_L
Limit distance calculation, for each centroid: DCW_LF, DL_LF
FINE-TUNING PHASE
Unknown triple [T1, T2, T3]
Distance to centroid (CryptoWall)
Distance to centroid (Locky)
D_L = (C_L[1]-T1)² + (C_L[2]-T2)² (C_L[3]-T3)²
D_CW, D_L
Test if D_CW < DCW_LF If true then CryptoWall detection
Test if D_L < DL_LF If true then Locky detection
DETECTION PHASE

Fig. 7. Main phases of proposed detection system.

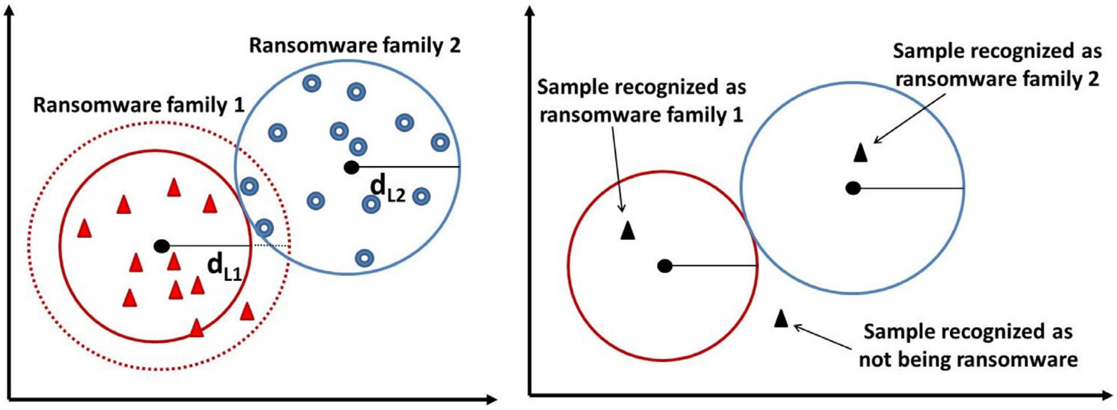

Ransomware family 2
Ransomware family 1
d_L2
d_L1
Sample recognized as ransomware family 1
Sample recognized as ransomware family 2
Sample recognized as not being ransomware

Fig. 8. Proposed detection method fine-tuning phase (left) and detection phase (right).

4.1.3. Detection phase

The HTTP connections that are the subject of the monitoring and potential ransomware detection must be initially preprocessed. During our experiments we developed two variants of the preprocessing software. The first one, mainly for experimental use, extracted data from the traffic traces (files in the .pcap format). The second one was directly integrated within the SDN controller and can be used for real-time detection of the existing ransomware infections. Both preprocessing solutions analyzed incoming TCP segments that contain HTTP traffic and reassembled the outgoing messages. When the whole request was reassembled, then the size of the data sent to the server and host IP or domain address were extracted from the HTTP header. The extracted host IP or domain name was then used for defining the destination HTTP server, which possibly could be a ransomware C&C or a proxy server. If confirmed as being malicious, information about such servers can be used, for example, to feed the ransomware countermeasures based on IP/domain blacklisting like the one proposed in [30] or directly block such traffic using the SDN-based system proposed in this paper.

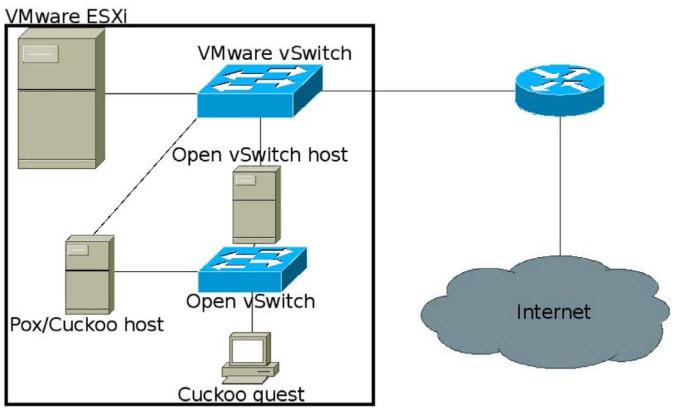


Fig. 9. Experimental testbed.

In the next step, for each HTTP server identified, as indicated above (malicious or not), a list was created that consisted of the generated HTTP messages sizes. Any new HTTP message was thus added at the end of the list associated with a given server's IP address or a domain name. If after insertion of a new value to the list, its size was equal to or greater than three (as for the proposed detection method we analyze three consecutive HTTP POST messages' sizes), then the detection of possible ransomware was performed. For this aim, the test vector was constructed using the last three values from the list. Then the distance between this vector and the centroid for a given ransomware family was measured. If the calculated value was smaller than the limit distance (d_{Lf}) established during the learning phase, the system decided that this HTTP message sequence was a sign of ransomware infection (Fig. 8, right). For example, consider the HTTP POST messages sequence presented in Fig. 5 for Locky. In this case, the extracted test vector was [910, 810, 770]. If during the learning phase the obtained centroid vector for this ransomware family had a value of [900, 800, 775] and the limit distance was fine-tuned to the value 20, then this traffic will be detected as ransomware, because the calculated Euclidean distance for this sample was 15.

4.2. Experimental test-bed

In the experimental evaluation we decided to utilize an OpenFlow protocol that enables typical networking devices to be supervised by an external controller, which is currently the standard for implementing the SDN paradigm. The experimental environment used during the experiments is depicted in Fig. 9. All operating systems were installed on virtual machines, which were managed by VMware ESXi hypervisor installed at a server with an Intel Xeon E5-2450 1.9 GHz (24 logical CPUs, 24 GB RAM). To exclude any interfering factors during the experiments, no other virtual machines were active on the hypervisor.

To securely execute the malware samples during the experiments with SDN detection, an application Cuckoo Sandbox [32] was used. In more detail, we utilized:

- A Cuckoo guest with Microsoft Windows 7 SP1 $\times$ 86_64 installed. We decided to use this OS as currently it is dominant with a market share of almost 50% of desktops [33]. Therefore, it is likely that it would be targeted by cybercriminals. It must also be noted that the update and firewall services were intentionally disabled on this host so the malware samples could be executed with no interruption.
- A Pox/Cuckoo and Open vSwitch (OVS) host using Debian 8.5 $\times$ 86_64 with kernel 3.16.36.

For separation purposes, two additional subnetworks were used: first for the Pox/Cuckoo traffic (using vNICs connected to Open vSwitch) and second for the management traffic (vNICs connected to VMware vSwitch).

The functioning of the proposed test-bed was as follows: after the Cuckoo agent (listener) was started on the guest machine, a snapshot with memory was taken. The OVS was acting as a default gateway for the Cuckoo guest, so the traffic directed towards the Internet was always passing through it. The OVS was controlled using the OpenFlow protocol by a Python-based SDN controller – Pox. The SDN application (based on the Layer 2 learning switch) forced the OVS to forward every HTTP packet to the controller for inspection. Next, at the controller the decision on the packet was made. During the experiments, the Cuckoo host first contacted the ESXi API to restore the Windows host from the snapshot. Then ransomware samples were executed. As a consequence, the traffic directed towards the C&C server was generated based on which SDN detection and prevention system made the decision to block it or not.

As stated before, the Pox application is based on a Layer 2 learning switch. The first requisite was to configure the OVS to send the whole packet to the SDN controller instead of the default 120 bytes. It was needed for the decision about rejecting or allowing traffic that was made based on the HTTP POST message size. Next, for every event (incoming packet) the handler function was invoked to check if:

Table 3
Characteristics of datasets used for evaluation of proposed detection method.

Dataset name	No. of HTTP POST	No. of POST triples	No. of domains
Ransomware traffic for CW 4.0	750	250	250
Ransomware traffic for Locky	750	250	250
Alexa traffic	22 579	11 950	8187
MACCDC traffic	17 249	15 862	761

- the packet is valid (can be parsed),
- the TCP is used in the transport layer,
- the TCP destination port is 80 (HTTP traffic),
- the packet payload length is greater than 0 (to exclude TCP segments with no data, for example, segments only with an ACK flag).

If any of the checks failed, no blocking rule was set as such a packet was not considered to be malware traffic. If all checks were positive, the payload was passed to the custom "oracle" function where the introduced detection algorithm was implemented (see Section 4.1). If a ransomware infection was detected, then the appropriate blocking flows (bidirectional using only the hostile C&C server's IP address) were inserted into the OVS switch (otherwise an infection on any other machine cannot be not prevented).

4.3. Utilized evaluation datasets

As described in the previous section, as well in Sections 3.1 and 3.2, our own datasets were used during the learning phase. This artificially generated data were prepared during the dynamic analysis. The evaluation phase required independent data that contained traffic with and without hostile activity. This section describes how the relevant malicious and benign network traffic was gathered. We collected malware samples from two main locations: the *malwr.com* website, from which it is possible to directly download ransomware samples, and *ransomtracker.abuse.ch*, where one can obtain the addresses of malicious distribution servers (the ones from which the ransomware binary is downloaded to the attacked host). Gathered samples were then executed in the controlled environment: the one described in Section 4.2 and Maltester [31] where the generated test traffic was captured. This formed the first dataset with malicious HTTP traffic.

To confirm that the proposed detection method is valid and does not generate too many false-positives, we utilized normal, benign HTTP traffic. As our proposed detection approach relies on HTTP POST message characteristics, we needed representative network dumps. However, it must be noted that it is difficult to obtain unencrypted HTTP traffic captures as they might contain sensitive and private data (logins, passwords, or cookies), thus such network dumps are usually not publically shared. Typically these repositories contain only metadata without upper layer payloads. Therefore, we generated benign HTTP traffic using 200,000 of the most popular websites from the Alexa ranking (extracted from http://s3.amazonaws.com/alexa-static/top-1 m.csv.zip on 19.10.2016). We wrote a bash script that launched each of the top-ranked Alexa websites in the Google Chrome web browser. It is worth noting that many pages are coded to generate a lot of POST messages without any users' interaction. This is to gather knowledge about the users' environment (operating system, web browser, setup, etc.) and not for usual POST purposes (like submitting forms or cookies). Such behavior is normal and can be observed world-wide, therefore the obtained traffic should not be considered as artificial or synthetic. As every website needs a certain amount of time to be fully loaded, to speed up the process, 25 pages were launched in parallel for 25 seconds. To exclude the possibility of influencing the interaction between different pages, the Google Chrome web browser was executed in Incognito Mode. Additionally, after every iteration, the web browser's cache was cleared. Therefore, the whole process of benign HTTP traffic generation took almost 56 hours, but as a result we obtained real-life HTTP traffic as it is typically experienced by Internet users.

The last part of the HTTP traffic was obtained from the 2012 MACCDC (maccdc.org) cyber defense competition data dumps. It is a capture the flag type event, which means that the HTTP data are not fully realistic, but they provided examples of various types of network attacks and contained many POST messages. Table 3 summarizes the statistics of the gathered HTTP POST and domains values.

4.4. Experimental results

This section describes in detail the process for the proposed detection method and its fine-tuning, and it presents the obtained experimental results. The experiments were performed independently for CryptoWall 4.0 and Locky ransomware families, and the results are described in the following subsections.

4.4.1. Locky ransomware

Initial data for the Locky family consisted of more than 250 traces of successful infections with recorded data exchanges between the infected victim and the C&C server. The complete dataset was divided into two distinct parts: the first, which

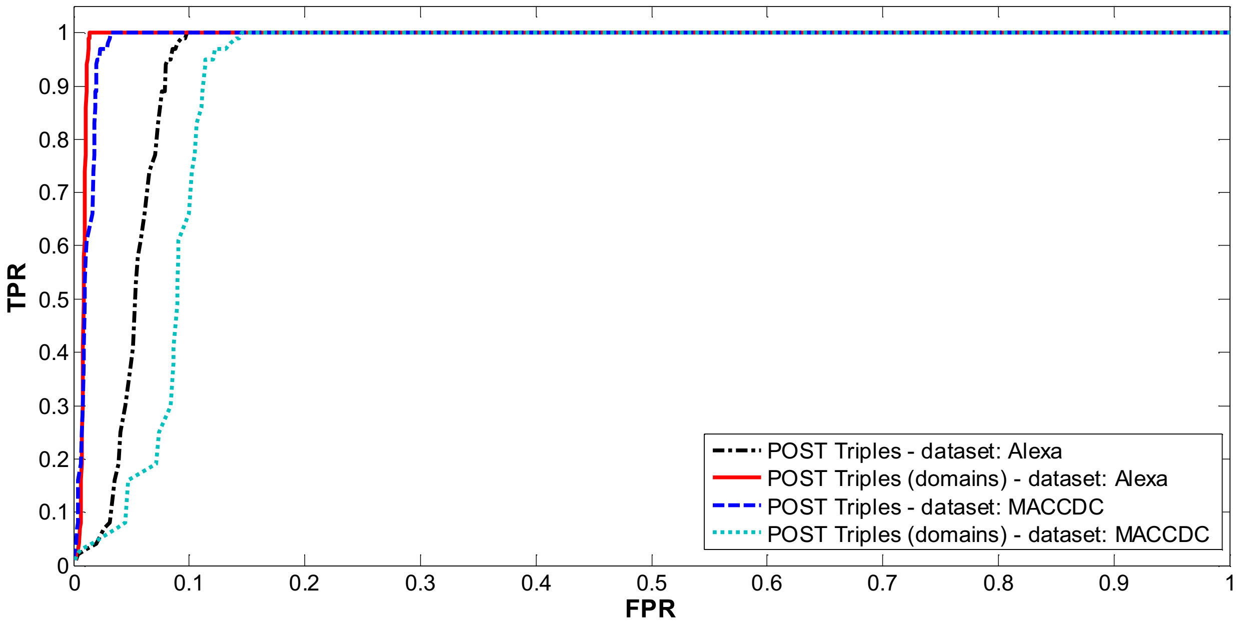


Fig. 10. ROC curves for Locky detection for Alexa and MACCDC datasets.

contained 150 samples, was used during the learning phase for the centroid parameters and the limit distance calculations. For each point in this dataset, the distance to the centroid was calculated. The results of the average, minimal, and maximum distance values were then used to determine the range of acceptable values for the limit distance parameters. For performance reasons, all calculations of the distance use the distance square, which avoids CPU intensive calculations of the square root. The results for the learning phase include the minimal distance (actually distance square) of 6914 and maximal of 274,370.

The second part of the original dataset consisted of 100 samples and it was used for the calculation of the TPR (True Positive Rate), which was needed to prepare the ROC curve for the proposed detection method. This was a reasonable decision as this data were generated from real infections (performed using the controlled and protected environment), and we are confident that they represent the real characteristics of the Locky communication traffic. During this phase the number of traces flagged as containing Locky infections was analyzed for various distance values, i.e., average, minimal, and maximum. Due to the fact that all the data contains infections, under an ideal situation we should flag all of them as malicious. The TPR was calculated as a ratio of flagged traces and the total number of samples in the dataset. The process of the FPR (False Positive Rate) calculation was similar. We obtained clean traces using two distinct datasets, described in Section 4.3, and we referred to them as Alexa and MACCDC. In these datasets, contrary to the previous dataset, which contained ransomware traffic, we should not detect any Locky infections. Therefore, all the connections recognized as Locky were false positives. For the ROC curve preparation purpose we checked these datasets using the same distance range as used for the TPR calculations. During our research we investigated two types of ROC curves. The first one was associated with all tested consequent triples of POST messages, regardless how many triples were associated with this domain and how many were flagged as malicious. This was mostly true for benign domains, because in successful ransomware traffic only three or four messages were observed. These plots were denoted as POST Triples. The second one associated all calculations with domains, for which at least one triple was flagged as a Locky transmission. In this case we could count at least one for a given domain. These plots were denoted as POST Triples (domain). Fig. 10 presents the ROC curves independently for Alexa and MACCDC datasets and both features, i.e., POST Triples and POST Triples (domain).

The ROC plots for both datasets were similar, so we decided to merge them for the final fine-tuning of the Locky detector. The final ROC curve is presented in Fig. 11. The results show that the most suitable limit distance (d_L) for our method is 260,000, because it offers the best trade-off between true positives (97%) and false positives (4.95%) and (2.2%) respectively for analyzed POST triples and domains.

4.4.2. CryptoWall 4.0 ransomware

During our research we performed similar analyses as presented in Section 4.4.1 for CryptoWall 4.0. From our previous studies [30,31] we had obtained almost 270 traces containing communication traffic between the infected host and the CryptoWall C&C servers. As this value was very similar to the number of traces used for the Locky experiments we decided to split this dataset in a similar way, i.e., used 150 traces for the learning phase and 100 for the testing phase. It turned out that the HTTP message sizes sent by CryptoWall 4.0 were much smaller than those of the Locky family. After the centroid

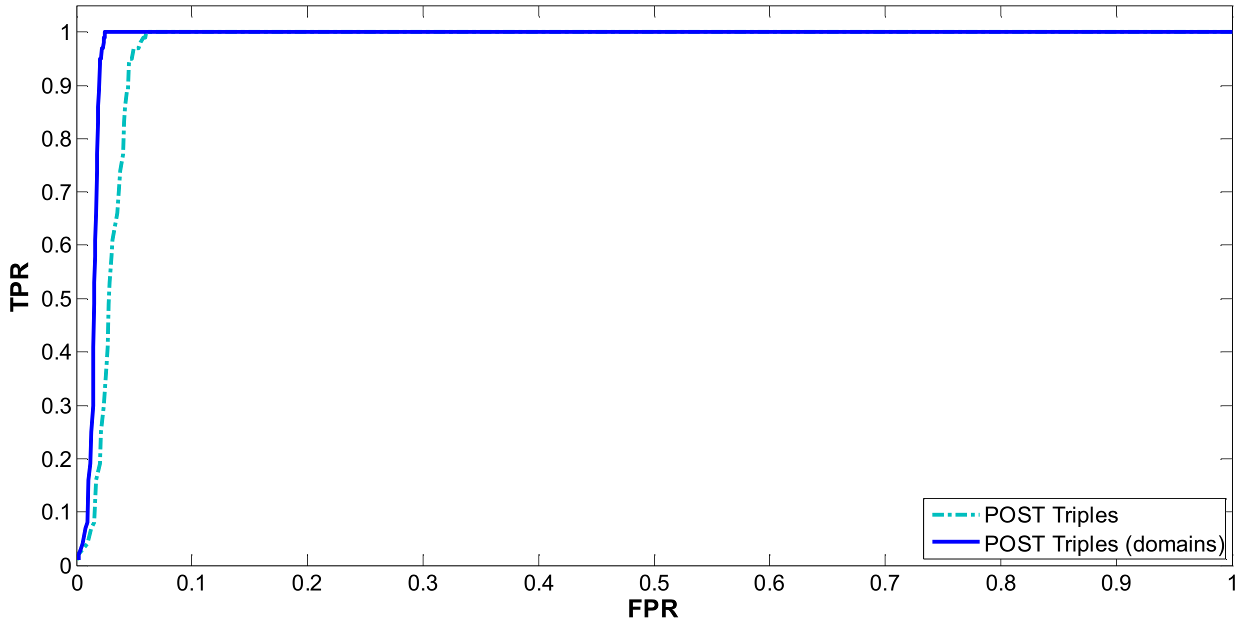


Fig. 11. ROC curves for Locky detection for merged Alexa and MACCDC datasets after final fine-tuning of detector.

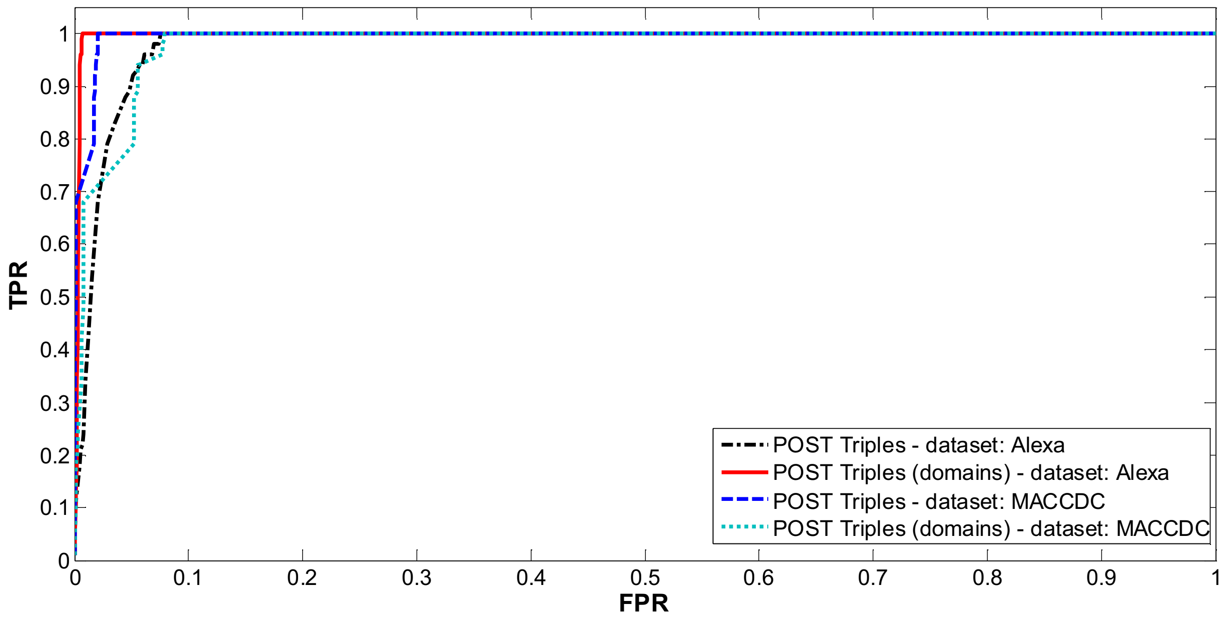


Fig. 12. ROC curves for CryptoWall 4.0 detection for Alexa (left) and MACCDC (right) datasets.

calculation we discovered that the minimal distance (for CPU performance reasons it was actually the distance square) was 25 and the maximal distance was 893. These values were then used during the TPR and FPR calculations. For TPR, to obtain 100% detection accuracy we used a limit distance of 1050. Using the second part of the traces containing the CryptoWall C&C communications, the Alexa and MACCDC datasets we gathered data for the ROC curves are illustrated in Fig. 12.

The ROC curves for both datasets were similar, so we decided to merge them for the final fine-tuning of the CryptoWall detector. The final ROC plot is illustrated in Fig. 13. The results show that the best limit distance (d_L) is 900, because it offers the best trade-off between true positives – 98% and false positives – 4.2% and 1.2% respectively for analyzed POST triples and domains.

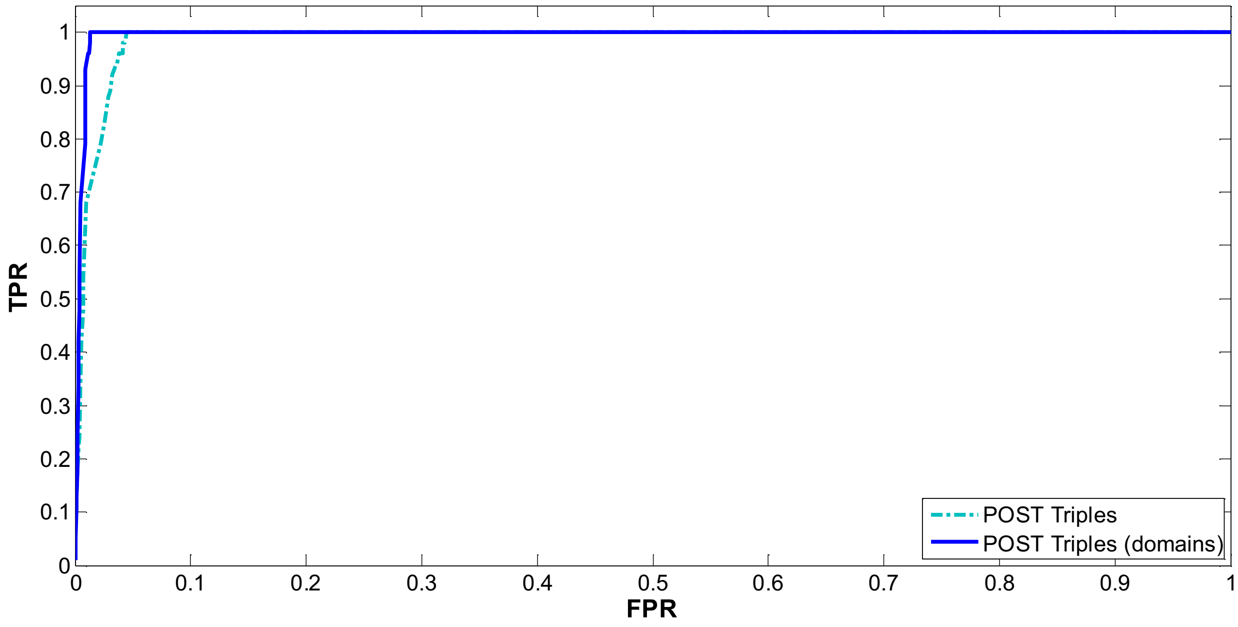


Fig. 13. Final ROC plots for CryptoWall 4.0 detection for merged Alexa and MACCDC datasets and after final fine-tuning of detector.

5. Conclusion

In this paper we proposed a novel SDN-based ransomware detection system that relies on the characteristics of the malware's communication. By performing network measurements for two very popular ransomware families, i.e., CryptoWall and Locky, we observed that a promising approach would be to detect the malicious communication between an infected host and the attacker C&C server using the HTTP traffic characteristics (HTTP message sequences and their corresponding sizes). Therefore, we designed and developed a detection system that uses the SDN solution to provide a rapid reaction to the discovered threat. The experimental results obtained using real ransomware samples proved that even such a simple approach is feasible and offers good efficacy. We were able to achieve detection rates of 97–98% with 1–2% or 4–5% false positives when relaying on domains or POST triples, respectively.

To counter ransomware, in general, we believe that it is vital to try to target and break the business model of the malware developers/exploiters by, for example, determining the most profitable malware families and disrupting their infrastructure. A complementary approach is to educate users to pay more attention to what types of files they open or what types of websites they visit. It is clear that less infections result in a lower profit and higher operation costs for the cybercriminals. Future work on ransomware detection should involve taking into account the sizes of the HTTP responses from the C&C server. Moreover, it is possible to combine the proposed approach with others, e.g., those based on blacklisting of malicious IP addresses and domains. Furthermore, it is vital to monitor ransomware development trends to provide effective countermeasures as fast as possible to limit the number of infected machines.

References

[1] Europol. Internet organised crime threat assessment 2016 (IOCTA); September 2016. URL: https://www.europol.europa.eu/content/internet-organised-crime-threat-assessment-iocta-2016.
[2] Savage K, Coogan P, Lau H. The evolution of ransomware. Symantec security response; August 2015. URL: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf.
[3] McAfee Labs. Meet 'Tox': ransomware for the rest of Us; May 2015. URL https://blogs.mcafee.com/mcafee-labs/meet-tox-ransomware-for-the-rest-of-us/.
[4] Symantec. Internet security threat report; April 2015. URL https://www4.symantec.com/mktginfo/whitepaper/ISTR/21347932_GA-internet-security-threat-report-volume-20-2015-social_v2.pdf.
[5] Kreutz D, Ramos FV, Verissimo P, Rothenberg C, Azodolmolky S, Uhlig S. Software-defined networking: a comprehensive survey. Proc IEEE January 2015;103(1):14–76.
[6] Mehdi SA, Khalid J, Khayam SA. Revisiting traffic anomaly detection using software defined networking. In: Proc. of the 14th international conference on recent advances in intrusion detection (RAID 2011); 2011. p. 161–80.
[7] Tofilski A, Couvillon M, Evison S, Helantera H, Robinson E, Ratnieks F. Preemptive defensive self-sacrifice by ant workers. Am Nat 11.2008,;172(5):E239–43.
[8] Mazurczyk W, Rzeszutko E. Security - a perpetual war: lessons from nature. IEEE IT Prof 2015;17(January/February 1):16–22.
[9] Gu G, Perdisci R, Zhang J, Lee W. Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Proceedings of the 17th USENIX security symposium; 2008.
[10] Wurzinger P, Bilge L, Holz T, Goebel J, Kruegel C, Kirda E. Automatically generating models for botnet detection. In: Backes M, Ning P, editors. ESORICS 2009. LNCS, 5789. Heidelberg: Springer; 2009. p. 232–49.

[11] Bailey M, Oberheide J, Andersen J, Mao ZM, Jahanian F, Nazario J. Automated classification and analysis of internet malware. In: Kruegel C, Lippmann R, Clark A, editors. RAID 2007. LNCS, 4637; 2007. p. 178–97.
[12] Bayer U, Comparetti PM, Hlauschek C, Kruegel C, Kirda E. Scalable, behavior-based malware clustering. Network and distributed system system security symposium; 2009.
[13] Jacob G, Hund R, Kruegel C, Holz T. Jackstraws: picking command and control connections from bot traffic. 20th USENIX security symposium; 2011.
[14] Idika N, Mathur AP. A survey of malware detection techniques. Purdue University; 2007. Technical Report.
[15] Rieck K, Schwenk G, Limmer T, Holz T, Laskov P, Botzilla. Detecting the phoning home of malicious software. In: Proceedings of the 25th ACM symposium on applied computing (SAC), March; 2010.
[16] Rossow C, Dietrich CJ. ProVeX: detecting botnets with encrypted command and control channels. In: Proc. of 10th international conference, DIMVA 2013, July 18–19; 2013. p. 21–40.
[17] Celik ZB, Walls R, McDaniel P, Swami A. Malware traffic detection using tamper resistant features. Military communications conference (MILCOM), October Tampa, FL, USA; 2015.
[18] Andronio N. Heldroid: fast and efficient linguistic-based ransomware detection M.Sc. thesis. University of Illinois at Chicago; 2015.
[19] Kharraz A, Robertson W, Balzarotti D, Bilge L, Kirda E. Cutting the Gordian knot: a look under the hood of ransomware attacks. 12th conference on detection of intrusions and malware & vulnerability assessment (DIMVA 2015), July 9–10 Milan, Italy; 2015.
[20] Scaife PTN, Carter H, Butler KR. Cryptolock (and drop it): stopping ransomware attacks on user data. In: 2016 IEEE 36th international conference on distributed computing systems; 2016. p. 303–12.
[21] Sgandurra D, Muñoz-González L, Mohsen R, Lupu EC. Automated dynamic analysis of ransomware: benefits, limitations and use for detection. Computing research repository (CoRR), Ithaca, NY (USA): Cornell University; September 2016. abs/ 1609.03020, arXiv.org E-print Archive.
[22] Zaalouk A, Khondoker R, Marx R, Bayarou K. OrchSec: an orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions. In: Proc. of network operations and management symposium (NOMS); 2014. p. 1–9.
[23] Shin S, Gu G. CloudWatcher: network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In: Proc. of 20th IEEE international conference on network protocols (ICNP); 2012. p. 1–6.
[24] Jin R, Wang B. Malware detection for mobile devices using software-defined networking. In: Proc. of GENI research and educational experiment workshop (GREE '13); 2013. p. 81–8.
[25] Ceron JM, Margi CB, Granville LZ. MARS: an SDN-based malware analysis solution. In: IEEE symposium on computers and communication (ISCC); 2016. p. 525–30.
[26] Perdisci R, Lee W, Feamster N. Behavioral clustering of HTTP-based malware and signature generation using malicious network traces. In: Proc. of the USENIX symposium on networked systems designs and implementation (NSDI); April 2010.
[27] Zegers R. HTTP header analysis M.Sc. thesis. University of Amsterdam; 2015.
[28] Kheir N. Analyzing HTTP user agent anomalies for malware detection. In: Proc. of 7th international workshop, DPM 2012, and 5th international workshop, SETOP 2012, September 13–14; 2012. p. 187–200.
[29] Grill M, Rehak M. Malware detection using HTTP user-agent discrepancy identification. In: IEEE international workshop on information forensics and security (WIFS); 2014. p. 221–6.
[30] Cabaj K, Mazurczyk W. Using software-defined networking for ransomware mitigation: the case of cryptowall. IEEE Netw. 2016(November/December). doi:10.1109/MNET.2016.1600110NM.
[31] Cabaj K, Gawkowski P, Grochowski K, Osojca D. Network activity analysis of CryptoWall ransomware. Przeglad Elektrotechniczny 2015;91(11):201–4 URL http://pe.org.pl/articles/2015/11/48.pdf.
[32] Cuckoo sandbox: automated malware analysis, URL: https://cuckoosandbox.org.
[33] Net market share, URL: https://www.netmarketshare.com/operating-system-market-share.aspx.

Krzysztof Cabaj holds M.Sc. (2004) and Ph.D. (2009) in computer science from Warsaw University of Technology (WUT). Assistant Professor at WUT. Instructor of Cisco Academy courses at International Telecommunication Union Internet Training Centre (ITU-ITC). His research interests include: network security, honeypots and data-mining techniques. He is the author or co-author of over 30 publications in the field of information security.

Marcin Gregorczyk is Ph.D. student and holds M.Sc. (2012) degree in telecommunications from the Warsaw University of Technology (WUT), Warsaw, Poland. RedHat Certified Architect and instructor of RedHat Enterprise Linux courses. His research interests include: Linux operating system security and network security especially cloud related.

Wojciech Mazurczyk received the M.Sc., Ph.D. (Hons.), and D.Sc. (habilitation) degrees in telecommunications from the Warsaw University of Technology (WUT), Warsaw, Poland, in 2004, 2009, and 2014, respectively. He is currently an Associate Professor with the Institute of Telecommunications, WUT. His research interests include network security, bio-inspired cybersecurity and networking and information hiding.