A DIVE INTO THE DEEP: DEMYSTIFYING WANNACRY CRYPTO RANSOMWARE NETWORK ATTACKS VIA DIGITAL FORENSICS

Aaron Zimba[1,2], Mwenge Mulenga[2]

[1]Department of Computer Science and Technology, University of Science and Technology Beijing
[2]Department of Computer Science and Information Technology, Mulungushi University
e-mails: azimba@xs.ustb.edu.cn, mmwenge@mu.ac.zm
China[1], Zambia[2]

Abstract: Ransomware has emerged as a resilient type of malware to reckon with which has inadvertently forced security researchers and companies into a game of cat and mouse. The media has not helped matters even as the inaccuracies and myths surrounding ransomware continue to deepen. In this paper, we endeavor to debunk WannaCry ransomware through its network interactions. We employ reverse engineering to perform an autopsy on the underlying malware program logic and further use dynamic analysis to capture the corresponding network actions associated with such logic. Based on the uncovered artifacts, we recommend preventative measures to thwart such attacks.

Key words: Ransomware, WannaCry, Encryption, Static Analysis, Dynamic Analysis.

1. INTRODUCTION

The malware landscape has seen a paradigm shift from nuisance and destructive malware to resilient and robust crimeware [1]. In the latter, to which ransomware identifies, the attacker has motivation for monetary gain and will use a myriad of attack vectors including social engineering to lure the benign victim. In January of 2018, Maersk, the largest container shipping company in the world revealed that they had reinstalled over 50,000 computers after a ransomware attack [2]. The incident cost the company about $300 million. The aforesaid implies that ransomware can no longer be ignored as it has come to strike the core of the economy in the world today. Almost all cyber breaches today involve the use of a malware depending on the infection vector. Ransomware, however is another breed of malware in that it employs the philosophy extortion to achieve its goal [3]. The basic idea behind ransomware is to hold a victim's file hostage not until a ransom demand is met. The most commonly used approach is encrypting a victim's files with resilient encryption and only releasing the decryption key upon payment of the demand.

The use of anonymous Bitcoin for such payments and transactions has made the tracking of cyber criminals a mammoth battle [4]. Such attractive prospect has seen the rise of various strains of ransomware [5] and Ransomware as a Service [6]. The latter is a cyber-crime service where cyber criminals offer ransomware software to non-technical users at a stipulated

fee. One of the most devastating ransomware attacks was WannaCry which affected over 150 countries in a short period of time [7]. It's the impact and effectiveness of this magnitude that has motivated this paper. The insights thereof would be very useful to both security researchers and systems administrators.

In light of the aforesaid, we perform static code analysis where we do not actively run the ransomware but dissect it to analyze its internal code. Furthermore, we carry out another experiment where we actively run a number of ransomware samples in a controlled environment and capture all the associated network activities. There are many behavioral features which can be extracted from the ransomware such as the encryption process, file interactions, registry alterations etc. Notwithstanding the aforementioned, our scope is limited to network related behaviors. This implies that in as far as static analysis is concerned, we concentrate on network-related source code directives. In the same light, the dynamic features that we capture are network related such beaconing to a Command and Control (C2) servers, detection of a kill-switch domain, exfiltration of encryption keys etc. [9]. Even though recent ransomware strains exhibit worm-like capabilities of self-network propagation, we contend that the initial infection vector could be malware-less (malware-free). Thus, the considered attack model is based on both malware-based and malware-less intrusion. The former encompasses spam emails, exploit kits, URL redirects etc. whilst the latter includes DNS tunneling, system misconfigurations, hardcoded default credentials etc. Despite spear fishing being one of the widely used ransomware delivery method, we use the latter for delivery of the ransomware payload in our dynamic analysis experiments due to ethical reasons.

The remainder of the paper is structured as follows: Section 2 expounds the fundamentals of ransomware and the related literature thereof. The methodology and experiment setup are brought forth in Section 3 while the corresponding results and analysis are discussed in Section 4. The conclusion is drawn in Section 5.

2. RELATED LITERARTURE AND CONCEPTS

There are three main components to a ransomware attack process: infection phase, encryption phase and the C2 communication phase. The diagram below in figure 1 illustrates the generic ransomware attack process.

2.1. Infection phase (1)

In this stage, the attacker specifies the encryption algorithm and generates a corresponding key pair K_p, K_s, of which the public key K_p might be implanted into the ransomware payload whilst the private key , K_s is retained by the attacker. Whether or not to implant the public key is dependent on the implemented attack structure. This occurs at the attacker's end or botnets. The ransomware is then delivered to the victim via any of the earlier discussed infection vectors.

2.2. Encryption phase (2)

This is the core of the ransomware business model because failure to actualize an effective encryption methodology renders all other aspects futile. The ransomware attacks by encrypting user files such as .doc, .pdf, .jpg, .xlsx etc. This is denoted by the operation $E_k(m_i, K_{secret}) = C_i$. If the encryption key was not implanted in the ransomware payload, the ransomware communicates with the C2 server as denoted by $C2\ Comms\ 1$ to retrieve the key. This is was a common feature in earlier ransomware families such as Cryptowall [13]

though this feature is absent in recent variants thus enabling the encryption process without the need to communicate with the C2 servers as in the case of WannaCry. In the case where the public key pair is generated on the infected host, some attack structures facilitate the exfiltration of the private key K_s as denoted by $C2\ Comms\ 1$. Some attack structures [14] are known to further encrypt the symmetric key K_{secret} in the event that the operating system's Crypto API for actual file encryption. This is denoted by the operation $E_k(K_{secret}, K_p) = C_j$ in the encryption phase.

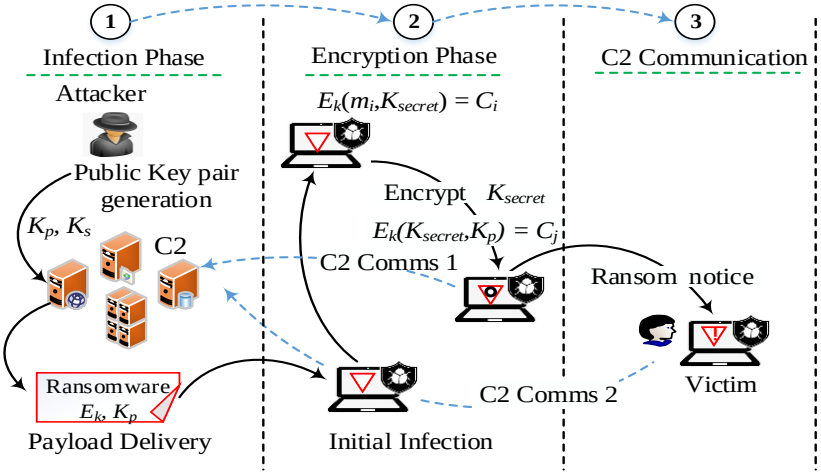

Fig. 1. Ransomware attack-chain process flow

2.3. C2 Communication phase

This phase denotes the state where encryption of targeted files has been successful and the victim is given a notice that their files have been encrypted and they need to pay. Most ransomware attacks required payments in terms of Bitcoin crypto-currencies [15]. Such transactions could be handled by the C2 infrastructure hence the operation $C2\ Comms\ 2$. In this stage, some ransomware also notify the attacker about the infected target. Ransomware are known to implemented secure C2 communications by using the Tor network and implementation of secure protocol suites such as SSL. However, communications of this nature reveal the domains and IP addresses associated with the ransomware in question. It is from this perspective of digital forensics that we base our dynamic analysis.

2.4. Related Works

Due to the interactive nature of ransomware, it's always certain that the malware will at point or the other interact with the network. This inadvertently entails that the underlying ransomware source code, whether in the attack structure or otherwise, harbors instructions on how to interact with the network. However, most research on ransomware seeks to holistically consider behavioral characteristics of different strains and families. Kharraz et al. [16] present a long-term study of ransomware attacks based on behavioral analysis. Authors contend that despite a continuous improvement in the encryption methodologies, C2 communications and deletion techniques, the amount of ransomware variants with destructive capabilities remains small. They further argue that most ransomware families employ superficial

techniques. However, their observations are solely based on behavioral analysis [8]. Furthermore, very little is addressed regarding network communications of the associated ransomware families.

Ahmadian et al. [17] present a novel methodology to detecting highly survivable ransomware based on a principal feature discovered in the encryption key exchange protocol. They also present a ransomware taxonomy with a clear distinction between those that are public-key cryptosystem and those that are private-key cryptosystem based. Nonetheless, their work is entirely based on behavioral features without any regard to static analysis. Furthermore, the ransomware network activities therein only pertain to activities between the C2 and the DGA.

Scaife et al. [18] present a detection methodology that uses behavioral indicators found in most ransomware families. Their proposed system can be parameterized for quick detection of ransomware attacks with low false positives. However, the dynamic analysis thereof does not include network behavior and neither any static analysis. Cabaj et al. [13] present detailed a network analysis of Cryptowall ransomware. However, there's no inspection of static code to uncover the basis of such behaviors or any sandbox evasion techniques if present.

3. METHODOLOGY

In order to extract digital forensics evidence from the ransomware, we statically analyze multiple WannaCry samples. The corresponding process flow of the reverse engineering is shown in figure 2 below. Furthermore, in order to deduce how the ransomware behaves on the network, we actively run the ransomware in a contained sandbox environment (Cuckoo) [19] as shown in figure 3 below. *Static Analysis (figure 2)*: In step 1, we select the ransomware variant to be analyzed, WannaCry in this case. We select two types of strains; one with a kill-switch domain and one without.

The unique identities of these ransomware is determined in step 2 where we run the corresponding cryptographic hashes; MD5, SHA-1 and SHA-256. Since some malware sample employ sandbox evasion techniques via obfuscation, we check the packing to determine whether or not internal program logic is disguised in step 3. To find out which strings point to the different network routines, we search for network activity associated strings in step 4. In step 5, we check for the corresponding meta-data and only extract that which is useful to network operations. We disassemble the ransomware source code using the interactive disassembly IDA Pro and Ollydbg [20, 21]. Upon disassembling the code, we hunt for instructions that enable the ransomware to carry out effective network activities. The results are presented in the proceeding section.

Dynamic Analysis (figure 3): The setup for dynamic analysis comprises a Cuckoo server running in Linux and victim machines running different versions of Windows operating system under virtualization, hence Virtual Machine Hosts (VM Host). The connection to the public Internet is sink-holed using different configurations of automatic DNS responders. We further monitor all network activities using Wireshark. Process Hacker is also deployed on VM Hosts to deduce which ransomware spawned sub-processes invoke network activities. Furthermore, the Cuckoo server aggregates the ransomware activities of all the configured guest VMs. The ransomware is deployed to the victim machines using malware-less intrusion as earlier clarified for ethical reason. The results of this analysis are likewise presented in the next section.

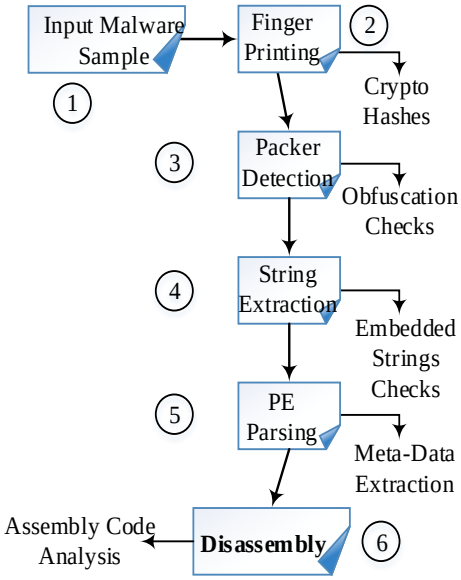

Fig. 2. Static analysis workflow

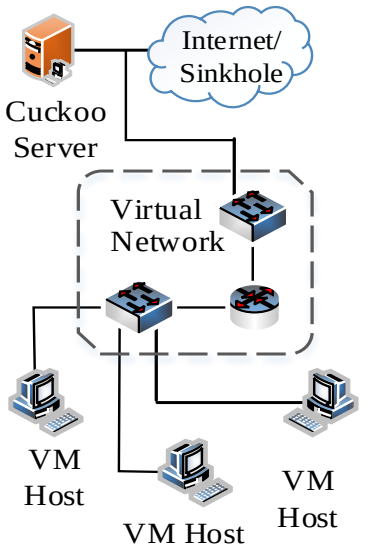

Fig. 3. Dynamic analysis setup

4. ILLUSTRATIVE RESULTS AND ANALYSIS

Here, we present the results from the experiments discussed in the previous section. Upon successful execution, in as far as network activity is concerned, we observe that the ransomware spawns multiple threads to initiate network related system calls. Particularly, one thread is created to scan IP addresses in the three private IP classes A, B and C as per RFC1918 specification [22]. The scan is initiated after determining whether it's residing in a local network by acquiring the relevant information using the GetAdaptersInfo() API. The ransomware code snippet for LAN IP scans is shown in figure 4 below.

```
int __cdecl InternalIPCheck(u_long hostlong)
{
  u_long v1; // eax@1
  int result; // eax@3

  v1 = htonl(hostlong);
  if ( v1 < 0xA000000 || v1 > 0xAFFFFFF )       // 10.0.0.0 ~ 10.255.255.255
  {
    if ( v1 < 0xAC100000 || v1 > 0xAC1FFFFF )   // 172.16.0.0 ~ 172.31.255.255
      result = v1 >= 0xC0A80000 && v1 <= 0xC0A8FFFF;// 192.168.0.0 ~ 192.168.255.255
    else
      result = 1;
  }
  else
  {
    result = 1;
  }
  return result;
}
```

Fig. 4. Code snippet for local IP addresses scans

We observe from the code that if the ransomware detects that it's running on a public network, 128 threads are spawned which scan public IP addresses generated by the cryptographically secure pseudo-random number generator (*CSPRNG*) from the CryptGenRandom() API by default. The first octet of the generated range cannot be equal to 127 as this is only usable for local loopback and also cannot be equal to or greater than 224 as any IP address in this range is not usable publicly. We show this in figure 5 below.

```
}
while ( num % 0xFF == 127 || first_octet >= 224 );// first octet not 127 or >= 224
if ( secong_octet_flag && threadID < 32 )
{
  v9 = PRNG(seed);
  seed = 255;
  second_octet = v9 % 0xFF;
}
third_octet = PRNG(seed) % 0xFFu;
fourth_octet = PRNG(0xFF);
sprintf(&targetIPAddress, aD_D_D_D, first_octet, second_octet, third_octet, fourth_octet % 0xFF, v17);
hInet = inet_addr(&targetIPAddress);
if ( CheckPort445(hInet) > 0 )              // check if it can connect to port 445
  break;
```

Fig. 5. Code snippet for public IP addresses scans

After this scan, it checks to determine whether SMB port 445 is listening for connections as depicted in the last part of the code in figure 5.

If connection to SMB port 445 is successful, the entire IP address block for that network is scanned and exploit attempts are made upon discovery. This is the worm [10] propagation component of WannaCry that enable it to attack a lot of hosts in such a short period of time. Such network propagation was possible due to the vulnerabilities CVE-2017-0143 - CVE-2017-0148 discovered in the SMBv1. The code snippet below shows the *EternalBlue* exploitation of this SMB vulnerability.

```
int __cdecl scan_IP(int a1)
{
  void *v1; // eax@2
  void *v2; // esi@2

  if ( can_connect_to_port_445(a1) > 0 )
  {
    v1 = (void *)beginthreadex(0, 0, MS17_010_attempt_pwn_thread, a1, 0, 0);
    v2 = v1;
    if ( v1 )
    {
      if ( WaitForSingleObject(v1, 600000u) == WAIT_TIMEOUT )
        TerminateThread(v2, 0);
      CloseHandle(v2);
    }
  }
  InterlockedDecrement((volatile LONG *)&FileName[268]);
  endthreadex(0);
  return 0;
}
```

Fig. 6. Port 445 SMBv1 vulnerability exploitation

It is worth noting that these scans are only valid for IPv4. Furthermore, the WannaCry ransomware comes in two variants; one with a kill-switch domain and another version without it. The former version first beacons to a non-existent domain upon successful execution. If that fake domain is reachable, the ransomware exits and does not attack, hence the kill-switch. This inherently is a sandbox evasion technique since only sandboxes (utilities such as FakeDNS) issue "valid" replies or domain lookups without actually connecting to the Internet. This means that a normal host connected to the Internet will give a genuine invalid domain lookup reply since the domain is non-existent. However, some of the kill-switches have been sink-holed implying that even a normally connected host to the Internet will give a positive reply meaning the ransomware won't actually run. This explains the variations in the newer strains which excluded the kill-switch.

It's also worth noting that for systems residing behind an upstream proxy, the strain with the kill-switch would still work owing to the nature of Internet accessibility under such a setting.

For initiation of network services, we observe that the ransomware actually launches system services after acquisition of required privilege level and loops through established RDP session of the victimized host in this manner running as the logged in user. Captured Wireshark traffic shows a high presence of Link-Local Multicast Name Resolution (LLMNR) on the local network. Evidently, the ransomware initiates a scan of the LAN upon determination that it is residing in a private IP address scope.

We observe that the variant with the kill-switch probes a name resolution of the kill-switch before any encryption takes place. The kill-switch name resolution request is shown in the diagram below in figure 7.

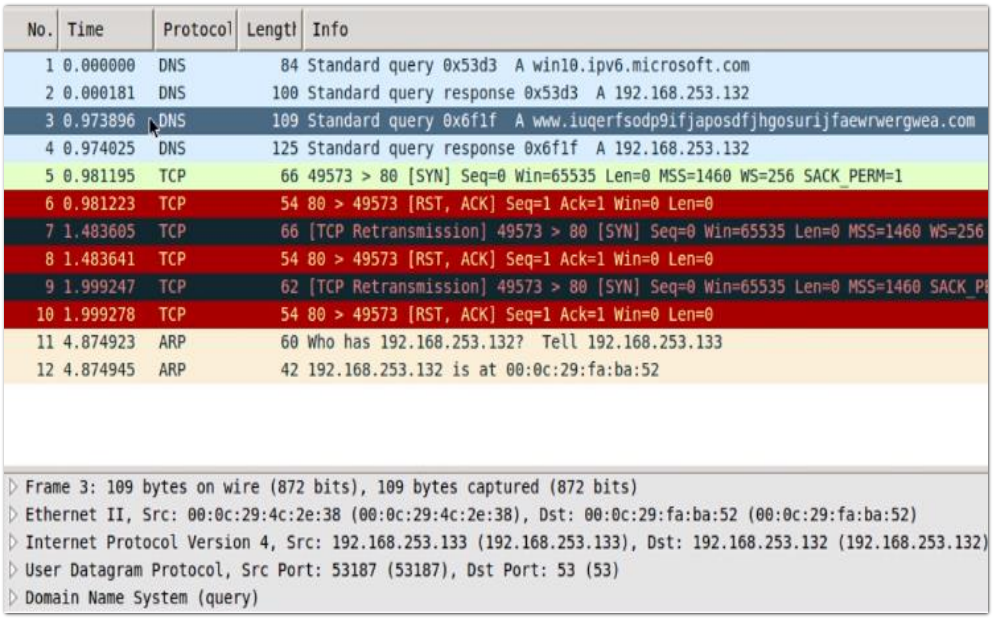

Fig. 7. Kill-switch name resolution at time

The kill-switch is shown in frame number 3 at time 0.973896 and it's quite evident that this is a randomly generated domain. Traffic logs with such request are indicators of compromise (IOCs). Note that the source and destination IP addresses have been omitted in the main capture but they are shown in the lower pane of the window. The diagram below in figure 8 shows a positive name resolution lookup from the FakeDNS [24] which in essence is a redirect. In this case, FakeDNS sink-holes the kill-switch and the encryption does not take place even though the host is already infected with a couple of spawned and actively running WannaCry child processes. This is evidenced in the Process Hacker tracking all the spawned processes.

Fig. 8. Sink-holed killswitch DNS lookup request

Here, we present the graphs we obtained from network traffic analysis. Conversely, when the ransomware runs successfully and gets access to the local network, it scans for SMB port 445 to search for the vulnerability. This is evidenced by a spike in SMB network traffic in the capture. It's worth noting that only the SMBv1 version is the only one susceptible to WannaCry attacks as it's the version contains the vulnerability. The diagram below in figure 9 shows average network traffic before SMB probe requests on the network.

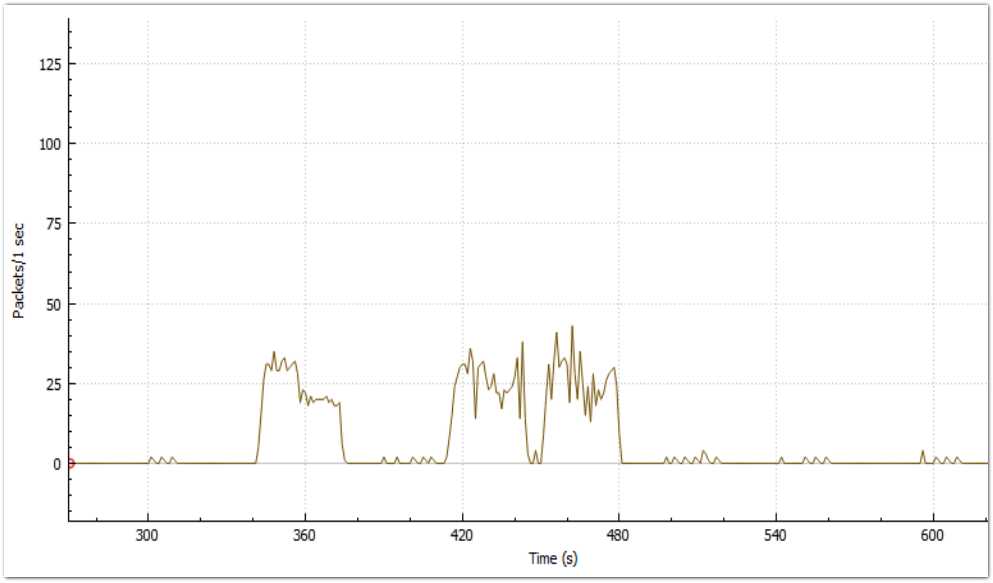

Fig. 9. Average network traffic before SMB network scans

After the ransomware runs on the local network, there's a sharp spike in network traffic in the same range. The earlier traffic before SMB scans is related to name lookups, ARP, DHCP requests etc. The diagram below in figure 10 shows network traffic capture with SMB traffic denoted in red. The presence of source huge amounts of SMBv1 traffic is already an IOC seeking to exploit the earlier mentioned vulnerability in this SMB version. It's clear from the network traffic capture that digital evidence lurks the network each time WannaCry ransomware attacks a local or public network. These artifacts left behind by the ransomware act as a red flag which can be fed into an intrusion detection system (IDS). The most vivid IOC in the variant with sandbox evasion techniques in the presence of the kill-switch domain which is rather a randomly generated domain name. Another apparent IOC is huge volume of SMB traffic from a single host probing other hosts for port 445 connections. In such a case, the host originating such requests could be isolated and investigated further as it would risk infecting other hosts on the same or adjacent subnets. WannaCry was able to inflict substantial damage as opposed to other ransomware because it utilized a worm component to self-propagate by exploiting a vulnerability in SMBv1. The WannaCry decryptor further tries to connect to the Tor network for secure Bitcoin transactions [23]. Presence of the associated Tor addresses is a further IOC. User awareness is a strong line of defense against WannaCry attacks in that like most ransomware, it utilizes social engineering for initial infection vectors. Malware-free infections [11, 12] though mostly associated with APTs are other possible infection sources entailing that awareness should also be extended to technical personnel tasked to secure information systems.

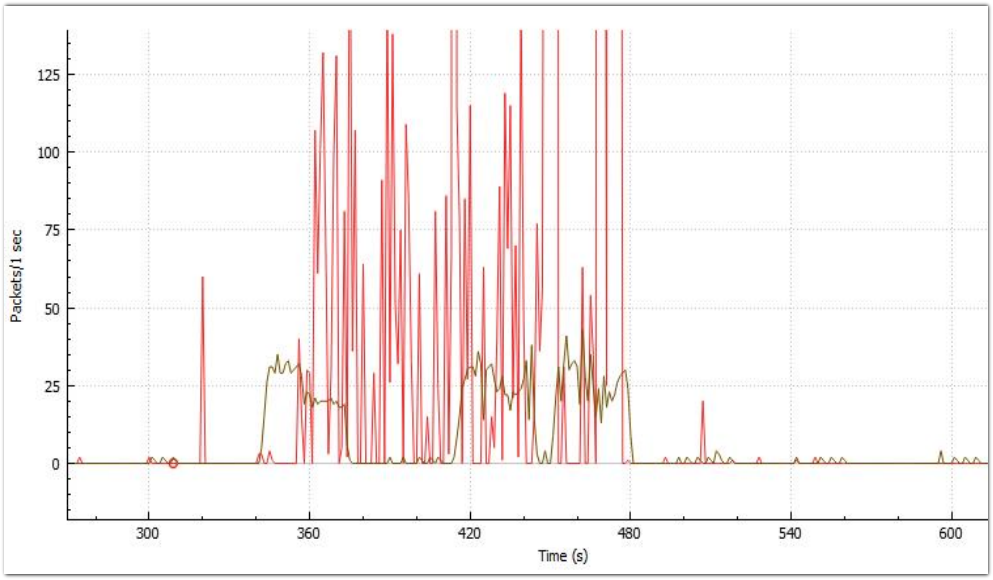

Fig. 10. Network traffic capture with SMB scans on port 445

Patching up Windows systems with the MS17-010 critical update likewise thwarts any potential attacks. Internet facing devices and DMZ ought to be properly segmented in the corporate or enterprise network as they would act as entry point into the entire network and to mapped network drives should SMBv1 vulnerability be discovered.

5. CONCLUSIONS

In this study, we have examined the network characteristics of WannaCry ransomware both from a static and behavioral perspective. The reverse engineering of the source code shows that the ransomware utilized a couple of techniques to effectuate substantial network propagation and subsequent damage. Particularly the ransomware fetches the network adapter properties to determine whether it's residing in a private or public subnet. This is a feature not seen in earlier ransomware as it gives the ransomware an effective methodology to find victims in either IP address scopes. The APIs used in the source code show that the ransomware is purely based on the Windows operating system. This explains why critical systems such as hospitals, transport systems, research centers [25] commonly known to use legacy software fell victim to this family of ransomware. Therefore, what made WannaCry effective over other ransomware was its effective network attack and propagation technique although efforts by the cyber criminals (ransomware authors) to implement sandbox evasion via the kill-switch led to its demise.

REFERENCES

[1] Jakobsson M, Ramzan Z. *Crimeware: understanding new attacks and defences*. Addison-Wesley Professional; 2008 Apr 6.

[2] Kovacs E. Maersk Reinstalled 50,000 Computers After NotPetya Attack. *Security Network*, 28 January, 2018.

[3] Richet J.L. *Extortion on the internet: the rise of crypto-ransomware*. Harvard. 2016.

[4] Richardson R, North M. Ransomware: Evolution, mitigation and prevention. *International Management Review.* 2017;13 Vol. (1):10.

[5] Mercaldo F, Nardone V, Santone A. Ransomware inside out. In *11th IEEE International Conference on Availability, Reliability and Security (ARES),* 2016 Aug 31 (pp. 628-637).

[6] Young A.L, Yung M. Cryptology: The birth, neglect, and explosion of ransomware. *Communications of the ACM*. 2017 Jun 26;60(7):24-6.

[7] Ehrenfeld J.M. Wannacry, cybersecurity and health information technology: A time to act. *Journal of Medical Systems.* Springer, 2017 Jul 1;41 (7):104.

[8] Ganame K, Allaire MA, Zagdene G, Boudar O. Network Behavioral Analysis for Zero-Day Malware Detection–A Case Study. In *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments* 2017 Oct 25 (pp. 169-181). Springer, Cham.

[9] Rafique M.Z, Chen P, Huygens C, Joosen W. Evolutionary algorithms for classification of malware families through different network behaviors. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* 2014 Jul 12 (pp. 1167-1174). ACM.

[10] Mansfield-Devine S. Ransomware: the most popular form of attack. *Computer Fraud & Security*. Elsevier, 2017 Oct 31; 2017 (10):15-20.

[11] Zimba A. Malware-Free Intrusion: A Novel Approach to Ransomware Infection Vectors. *International Journal of Computer Science and Information Security*. 2017 Feb 1; 15(2):317.

[12] Andreev S, Balandin S, Koucheryavy Y. *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. Springer International Publishing; 2014.

[13] Cabaj K, Gawkowski P, Grochowski K, Osojca D. Network activity analysis of CryptoWall ransomware. *Przeglad Elektrotechniczny*. 2015; 91(11):201-4.

[14] Zimba A, Simukonda L. and Chishimba M. Demystifying Ransomware Attacks: Reverse Engineering and Dynamic Malware Analysis of WannaCry for Network and Information Security. *Zambia ICT Journal*. Vol 1 No 1. Dec 11, 2017.

[15] Liao K, Zhao Z, Doupé A, Ahn G.J. Behind closed doors: measurement and analysis of CryptoLocker ransoms in Bitcoin. In *IEEE APWG Symposium on Electronic Crime Research (eCrime)*, 2016 2016 Jun 1 (pp. 1-13).

[16] Kharraz A, Robertson W, Balzarotti D, Bilge L, Kirda E. Cutting the Gordian knot: A look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* 2015 Jul 9 (pp. 3-24). Springer, Cham.

[17] Ahmadian M.M, Shahriari H.R, Ghaffarian S.M. Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomwares. In *12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, 2015 Sep 8 (pp. 79-84). IEEE.

[18] Scaife N, Carter H, Traynor P, Butler K.R. Cryptolock (and drop it): stopping ransomware attacks on user data. In *IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, 2016 Jun 27 (pp. 303-312). IEEE.

[19] Cuckoo Sandbox - Automated Malware Analysis. (Jan 2018) [Online] Available:https://cuckoosandbox.org/

[20] IDA: About - Hex-Rays. (May 27, 2015). [Online] Available: https://www.hex-rays.com/products/ida/

[21] Gagnon M.N, Taylor S, Ghosh A.K. Software protection through anti-debugging. *IEEE Security & Privacy*. 2007 May; 5(3).

[22] Rekhter Y, Moskowitz B, Karrenberg D, de Groot GJ, Lear E. *Address allocation for private internets*. (No. RFC 1918). 1996.

[23] Reid F, Harrigan M. An analysis of anonymity in the Bitcoin system. In *Security and privacy in social networks* 2013 (pp. 197-223). Springer, New York, NY.

[24] Janbeglou M, Zamani M, Ibrahim S. Redirecting outgoing DNS requests toward a fake DNS server in a LAN. In *IEEE International Conference on Software Engineering and Service Sciences (ICSESS)*, 2010 Jul 16 (pp. 29-32). IEEE.

[25] Mattei T.A. Privacy, Confidentiality, and Security of Health Care Information: Lessons from the Recent WannaCry Cyberattack. *World Neurosurgery* 104, pp. 972-974. Elsevier 2017.

Information about the authors:

Aaron Zimba is currently a PhD candidate at the University of Science and Technology Beijing in the Department of Computer Science and Technology. He holds Master and Bachelor of Science degree from the St Petersburg Electrotechnical University in St Petersburg. He's a member of the IEEE and his main research interests include Network and Information Security, Cloud Computing Security and Network Security Models.

Mwenge Mulenga is a lecturer of Computer Science and the associate head of the School of Science, Engineering and Technology at Mulungushi University. He holds a Master's degree from the St Petersburg State Electrotechnical, Russia. He has vast experience in major software projects implementing both proprietary and open-source technologies. His main research interests include software engineering and machine learning.

Manuscript received on 28 January 2018