Defending Against the Advanced Persistent Threat: Detection of Disguised Executable Files

Ibrahim Ghafir[1,2], Mohammad Hammoudeh[2], Vaclav Prenosil[1]

[1] Faculty of Informatics, Masaryk University, Brno, Czech Republic

[2] Faculty of Science & Engineering, Manchester Metropolitan University, Manchester, UK

ghafir@mail.muni.cz, M.Hammoudeh@mmu.ac.uk, prenosil@fi.muni.cz

Abstract

Advanced Persistent Threat (APT) is one of the most serious types of cyber attacks, which is a new and more complex version of multi-step attack. Within the APT life cycle, the most common technique used to get the point of entry is spear-phishing emails which may contain disguised executable files. This paper presents the disguised executable file detection (DeFD) module, which aims at detecting disguised exe files transferred over the connections. The detection is based on a comparison between the *MIME type* of the transferred file and the file name extension. This module was experimentally evaluated and the results show successful detection of disguised executable files.

Keywords—Cyber attacks, advanced persistent threat, spear-phishing emails, disguised executable file, malware, intrusion detection system.

I. Introduction

Over the last decade, cyber attacks and intrusions have increased substantially, utilising different tools and techniques such as malware, botnets [1], [2], [3] and social engineering [4]. Furthermore, since the Internet of Things (IoT) has emerged, the number of devices connected to wireless networks is increasing rapidly [5], [6]. One of the most serious types of cyber attacks is the Advanced Persistent Threat (APT) [7], which is targeting a specific organisation and it is performed through several steps. The main aim of APT is espionage and then data exfiltration. Therefore, APT is considered as a new and more complex version

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2998v2 | CC BY 4.0 Open Access | rec: 15 Jun 2018, publ: 15 Jun 2018

of multi-step attack [8]. These APTs form a problem for current detection methods [9], [10] as they use advanced techniques like social engineering [11] and make use of unknown vulnerabilities. Moreover, the economic damages due to a successful APT attack can be very expensive. The expected cost of attacks is the major motivation for the investments in intrusion detection and prevention systems [12]. APTs are currently one of the most serious threats to the companies and governments [13].

A novel approach for APT detection is proposed in [14]. The suggested system undergoes two main phases, the first one detects eight techniques commonly used in APT life cycle. For that purpose, eight detection modules are presented, which are disguised exe file detection, malicious file hash detection [15], malicious domain name detection [16], malicious IP address detection [17], malicious SSL certificate detection [18], domain flux detection [19], scan detection, and Tor connection detection [20]. The second phase includes a correlation framework to link the outputs of the detection modules.

Within the APT life cycle, the most common technique used to get the point of entry is spear-phishing emails [21] which may contain disguised exe files. This paper presents the disguised exe file detection (DeFD) module, which aims at detecting disguised exe files transferred over the connections. The detection is based on a comparison between the *MIME type* of the transferred file and the file name extension.

The remainder of this paper is organized as follows. Section II presents the related work to APT detection. The disguised exe file detection module and its algorithm are explained in Section III. Section IV shows the evaluation results and Section V concludes the paper.

II. Related Work

A classification model for APT detection is presented in [22]. This model is built based on machine learning algorithms. First, the legitimate traffic for normal users is analysed aiming to extract the CPU usage, memory usage, open ports and number of files in the system32 folder. A piece of malware, which is previously used for the APT attack, is injected into the network and the four features are extracted. The dataset of benign and malicious features are used to train the detection model using different machine learning algorithms.

TerminAPTor, an APT detector, is described in [23]. This detector uses information flow tracking to find the links between the elementary attacks, which are triggered within the

APT life cycle. TerminAPTor depends on an agent, which can be a standard intrusion detection system, to detect those elementary attacks. A statistical APT detector, similar to TerminAPTor detector, is developed in [24]. This system considers that APT undergoes five states which are delivery, exploit, installation, C&C and actions; and several activities are taken in each state. The generated events in each state are correlated in a statistical manner.

An APT detection system based on C&C domains detection is introduced in [25]. This work analyses the C&C communication based on the observation that the access to C&C domains is independent, while the access to legal domains is correlated.

An approach for APT detection based on spear phishing detection is explored in [26]. This approach depends on mathematical and computational analysis to filter spam emails. Tokens, which are considered as a group of words and characters such as (click here, free, Viagra, replica), should be defined for the detection algorithm to separate legitimate and spam emails.

An active-learning-based framework for malicious PDFs detection is suggested in [27]. These malicious PDFs can be used in the early steps of APT to get the point of entry.

An approach based on Data Leakage Prevention (DLP) is proposed in [28]. This approach focuses on detecting the last step of APT which is the data exfiltration. A DLP algorithm is used to process the data traffic to detect data leaks and generate "fingerprints" according to the features of the leak.

A context-based framework for APT detection is explained in [29]. This framework is based on modelling APT as an attack pyramid in which the top of the pyramid represents the attack goal, and the lateral planes indicates the environments involved in the APT life cycle.

An in-depth analysis of Duqu is presented in [30]. A European organisation was targeted by attackers using the Duqu malware to steal data. The authors propose the Duqu detector toolkit, which consists of six investigation tools developed to detect the Duqu malware involved in the APT attacks.

With regards to the processing of multiple streams of events, IBM suggests a conceptual model for event processing in [31]. It describes the basic requirements to design an efficient correlation system. The work presented in [32] is based on finite state machines and uses a query language for event processing. Both systems can process the events in real time, a key limitation shared by both approaches is that they can not detect so called low & slow attacks, which take place over an extended time period.

Finally, APT detection systems face serious shortcomings in achieving real time detection [33], detecting all APT attack steps [33], balance between false positive and false negative rates [30], and correlating of events spanning over a long period of time [31] [32]. To address those weaknesses, a new approach for APT detection has been presented in [14], and this paper is a step towards developing the proposed system.

III. Disguised executable File Detection (DeFD)

According to Mandiant APT1 report [34], spear phishing is the most commonly used technique to get the point of entry in APT. The spear phishing emails contain either a malicious attachment or a hyperlink to a malicious file. The subject line and the text in the email body are usually relevant to the recipient. Executable files supposed to end in .exe are made to appear as simple document files (pdf, doc, ppt, excel) to convince the victim to click on it.

The DeFD module detects disguised exe files transferred over the connections. In other words, it detects if the content of the file is exe while the extension is not exe. Figure 1 shows the methodology of DeFD; the network traffic is processed, all connections are analysed and all exe files identified when transferring over the connections are filtered. This filtering is based on the file content. Following this, the file name extension should be checked to decide about raising an alert on disguised exe file detection.

Different security systems are used to monitor the network traffic [35] [36]. The DeFD module is implemented on top of the *Bro* [37] [35]. Bro event engine reduces the incoming packet stream into a series of higher-level *events*, more than 300 events.

Algorithm 1 shows the implementation pseudo-code of the DeFD module. The network traffic is processed; this module waits for *file_over_new_connection* event to be generated by Bro. This event indicates that a file has been seen in the process of being transferred over a connection [38]. Then *describe()* function is applied on that file; this function provides a text description regarding metadata of the file, so the file name can be extracted.

This method is able to detect disguised exe files in both cases, uploaded and downloaded,

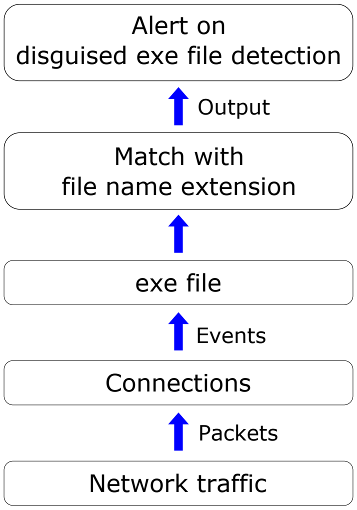


Fig. 1. Methodology of the disguised exe file detection.

but DeFD aims to detect only downloaded disguised exe files, as they are the ones used in the second step of APT. Thus, DeFD checks the current connection, in which a new file has seen being transferred, if it is established by a host from the monitored network. This is done by checking the connection source IP address through *is_ local_ addr* function; this function returns true if an address corresponds to one of the defined local networks, false if not. For this reason, the *subnet* of the monitored network should be defined.

Following this, the *mime_ type* [39], which can be extracted from *file_ over_ new_ connection* event, is checked for its presence in *t_ exe_file* table. *t_ exe_file* table contains the MIME types of the files which DeFD aims to find and filter, i.e. MIME types of exe files. Therefore, if the transferred file is an exe file (based on its *mime_ type*), DeFD checks whether the extension in the file name is exe; this file name extension is extracted from the output of the *describe()* function previously mentioned. If the file name extension is not exe, this means it is a disguised exe file. Before an alert is raised, DeFD checks if an alert regarding the same host and for the same disguised exe file has been generated during the previous day. This check is to ensure that DeFD does not generate the same alert about the same set (host, file) during one day, therefore, DeFD checks if the current set exists in the *t_ suppress_ disguised_ exe_ alert* table, this table contains all detected sets during the last day.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2998v2 | CC BY 4.0 Open Access | rec: 15 Jun 2018, publ: 15 Jun 2018

Algorithm 1 Implementation pseudo-code of DeFD

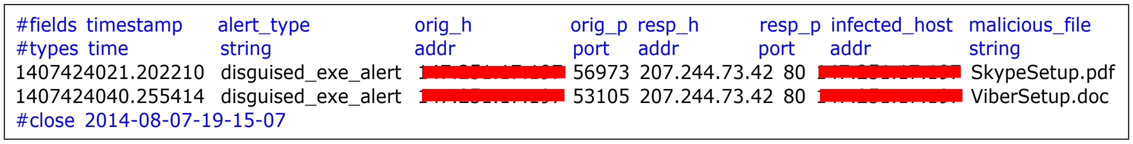
```
Get t_exe_file table
Get file_over_new_connection event
fname ← file name
if the connection is established by a host from the monitored network then
    if the file MIME type is in t_exe_file table then
        if file MIME type = fname extension then
            if the same disguised_exe_alert has been generated over the last day then
                goto End
            else
                Generate an event (disguised_exe_alert)
                Write disguised_exe_alert into disguised_exe_detection.log
                Send an alert email to RT
                Suppress the same disguised_exe_alert over the next day
            end if
        else
            goto End
        end if
    else
        goto End
    end if
else
    goto End
end if
End
```

If the current set (host, file) had not been detected during the previous 24 hours, DeFD generates *disguised_exe_alert* event to be used in the FCI correlation framework. The malicious connection information is written into a specific log *disguised_exe_detection.log*, to keep a historical record of the monitored network. An alert email regarding disguised exe file detection is sent to RT, where the network security team can perform additional forensics and respond to it. The current detected set (host, file) is added into the *t_ suppress_ disguised_ exe_ alert* table where it stays for one day to ensure that DeFD does not generate another alert about the same set during the same day. The written information into *disguised_ exe_ detection.log* is:

```
timestamp = c$start_time,
alert_type = "disguised_exe_alert"
connection  = c$id
infected_host = c$id$orig_h
malicious_file = fname
```

IV. Evaluation Results

To evaluate the effectiveness of the DeFD module, a download of disguised exe file was simulated via the campus network. An experimental server was set up, using Bro, to passively monitor the campus live traffic. DeFD was run on that experimental server for the purpose of disguised exe file detection. Two exe files were randomly selected, *SkypeSetup.exe* and *ViberSetup.exe*, and their names' extensions (*exe*) were changed into *pdf* and *doc* extensions, i.e. *SkypeSetup.pdf* and *ViberSetup.doc* respectively. The two disguised exe files were uploaded to the *speedyshare.com* public server. Then a host working on a computer connected to the Internet through the monitored network was used to connect to that public server and download the modified files. As shown in Figure 2, DeFD was able to detect both malicious downloads and write the information regarding each connection into a specific log.



```
#fields timestamp     alert_type          orig_h      orig_p resp_h        resp_p infected_host malicious_file
#types  time          string              addr        port   addr          port   addr          string
1407424021.202210  disguised_exe_alert  [redacted]  56973  207.244.73.42  80  [redacted]  SkypeSetup.pdf
1407424040.255414  disguised_exe_alert  [redacted]  53105  207.244.73.42  80  [redacted]  ViberSetup.doc
#close  2014-08-07-19-15-07
```

Fig. 2. The log produced by the DeFD module.

This experiment was repeated a hundred of times using different disguised exe files with different extensions. DeFD was able to detect all the malicious files, the average detection delay was 270 ms with a standard deviation of 54 ms.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2998v2 | CC BY 4.0 Open Access | rec: 15 Jun 2018, publ: 15 Jun 2018

V. Conclusion and Future Work

This paper presents the disguised exe file detection (DeFD) module, which aims at detecting disguised exe files transferred over the connections. The detection is based on a comparison between the *MIME type* of the transferred file and the file name extension.

For future work, the output of this module will be correlated with the outputs of other detection modules, developed to detect commonly used techniques over the APT life cycle, to raise an alert on APT detection.

References

[1] I. Ghafir, V. Prenosil, M. Hammoudeh, T. Baker, S. Jabbar, S. Khalid, and S. Jaf, "Botdet: A system for real time botnet command and control traffic detection," *IEEE Access*, 2018.

[2] I. Ghafir, V. Prenosil, and M. Hammoudeh, "Botnet command and control traffic detection challenges: A correlation-based solution," *International Journal of Advances in Computer Networks and Its Security (IJCNS)*, vol. vol. 7, no. Issue 2, pp. 27–31, 2015.

[3] I. Ghafir, J. Svoboda, and V. Prenosil, "A survey on botnet command and control traffic detection," *International Journal of Advances in Computer Networks and its security (ICJNS)*, vol. vol. 5, no. Issue 2, pp. 75–80, 2015.

[4] I. Ghafir, J. Saleem, M. Hammoudeh, H. Faour, V. Prenosil, S. Jaf, S. Jabbar, and T. Baker, "Security threats to critical infrastructure: the human factor," *The Journal of Supercomputing*, pp. 1–17, 2018.

[5] S. Walker-Roberts, M. Hammoudeh, and A. Dehghantanha, "A systematic review of the availability and efficacy of countermeasures to internal threats in healthcare critical infrastructure," *IEEE Access*, 2018.

[6] U. Raza, J. Lomax, I. Ghafir, R. Kharel, and B. Whiteside, "An iot and business processes based approach for the monitoring and control of high value-added manufacturing processes," in *Proceedings of the International Conference on Future Networks and Distributed Systems.* ACM, 2017, p. 28.

[7] I. Ghafir and V. Prenosil, "Advanced persistent threat attack detection: An overview," *International Journal of Advances in Computer Networks and Its Security (IJCNS)*, vol. vol. 4, no. Issue 4, pp. 50–54, 2014.

[8] Z. Liu, C. Wang, and S. Chen, "Correlating multi-step attack and constructing attack scenarios based on attack pattern modeling," in *Information Security and Assurance, 2008. ISA 2008. International Conference on.* IEEE, 2008, pp. 214–219.

[9] I. Ghafir, M. Husak, and V. Prenosil, "A survey on intrusion detection and prevention systems," in *Proceedings of student conference Zvule, IEEE/UREL.* Brno University of Technology, 2014, pp. 10–14.

[10] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, S. Khalid, and F. J. Aparicio-Navarro, "Detection of advanced persistent threat using machine-learning correlation analysis," *Future Generation Computer Systems*, 2018.

[11] I. Ghafir, V. Prenosil, A. Alhejailan, and M. Hammoudeh, "Social engineering attack strategies and defence approaches," in *Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on.* IEEE, 2016, pp. 145–149.

[12] T. R. Rakes, J. K. Deane, and L. Paul Rees, “It security planning under uncertainty for high-impact events,” *Omega*, vol. 40, no. 1, pp. 79–88, 2012.

[13] P. Wood, M. Nisbet, G. Egan, N. Johnston, K. Haley, B. Krishnappa, T.-K. Tran, I. Asrar, O. Cox, S. Hittel *et al.*, “Symantec internet security threat report trends for 2011,” *Volume XVII*, 2012.

[14] I. Ghafir and V. Prenosil, “Proposed approach for targeted attacks detection,” in *Advanced Computer and Communication Engineering Technology*. Springer, 2016, pp. 73–80.

[15] I. Ghafir and V. Prenosil, “Malicious file hash detection and drive-by download attacks,” in *Proceedings of the Second International Conference on Computer and Communication Technologies*. Springer, 2016, pp. 661–669.

[16] I. Ghafir and V. Prenosil, “DNS traffic analysis for malicious domains detection,” in *2nd International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE Xplore Digital Library, 2015, pp. 613–918.

[17] I. Ghafir and V. Prenosil, “Blacklist-based malicious ip traffic detection,” in *Global Conference on Communication Technologies (GCCT)*. IEEE Xplore Digital Library, 2015, pp. 229–233.

[18] I. Ghafir, M. Hammoudeh, and V. Prenosil, “Malicious SSL certificate detection: A step towards advanced persistent threat defence,” in *Proceedings of International Conference on Future Networks and Distributed Systems*. Cambridge, United Kingdom: ACM Digital Library, 2017.

[19] I. Ghafir and V. Prenosil, “DNS query failure and algorithmically generated domain-flux detection,” in *International Conference on Frontiers of Communications, Networks and Applications (ICFCNA)*. IEEE Xplore Digital Library, 2014, pp. 1–5.

[20] I. Ghafir, J. Svoboda, and V. Prenosil, “Tor-based malware and tor connection detection,” in *International Conference on Frontiers of Communications, Networks and Applications (ICFCNA)*. IEEE Xplore Digital Library, 2014, pp. 1–6.

[21] I. Ghafir and V. Prenosil, “Advanced persistent threat and spear phishing emails,” in *Proceedings of International Conference on Distance Learning, Simulation and Communication*. University of Defence, 2015, pp. 34–41.

[22] S. Chandran, P. Hrudya, and P. Poornachandran, “An efficient classification model for detecting advanced persistent threat,” in *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*. IEEE, 2015, pp. 2001–2009.

[23] G. Brogi and V. V. T. Tong, “Terminaptor: Highlighting advanced persistent threats through information flow tracking,” in *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*. IEEE, 2016, pp. 1–5.

[24] J. Sexton, C. Storlie, and J. Neil, “Attack chain detection,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 8, no. 5-6, pp. 353–363, 2015.

[25] X. Wang, K. Zheng, X. Niu, B. Wu, and C. Wu, “Detection of command and control in advanced persistent threat based on independent access,” in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[26] J. V. Chandra, N. Challa, and S. K. Pasupuleti, “A practical approach to e-mail spam filters to protect data from advanced persistent threat,” in *Circuit, Power and Computing Technologies (ICCPCT), 2016 International Conference on*. IEEE, 2016, pp. 1–5.

[27] N. Nissim, A. Cohen, C. Glezer, and Y. Elovici, “Detection of malicious pdf files and directions for enhancements: a state-of-the art survey,” *Computers & Security*, vol. 48, pp. 246–266, 2015.

[28] J. Sigholm and M. Bang, “Towards offensive cyber counterintelligence: Adopting a target-centric view on advanced persistent threats,” in *Intelligence and Security Informatics Conference (EISIC), 2013 European*. IEEE, 2013, pp. 166–171.

[29] P. Giura and W. Wang, “A context-based detection framework for advanced persistent threats,” in *Cyber Security (CyberSecurity), 2012 International Conference on*. IEEE, 2012, pp. 69–74.

[30] B. Bencsáth, G. Pék, L. Buttyán, and M. Félegyházi, “Duqu: Analysis, detection, and lessons learned,” in *ACM European Workshop on System Security (EuroSec)*, vol. 2012, 2012.

[31] C. Moxey, M. Edwards, O. Etzion, M. Ibrahim, S. Iyer, H. Lalanne, M. Monze, M. Peters, Y. Rabinovich, G. Sharon *et al.*, “A conceptual model for event processing systems,” *IBM Redguide publication*, 2010.

[32] L. Brenna, A. Demers, J. Gehrke, M. Hong, J. Ossher, B. Panda, M. Riedewald, M. Thatte, and W. White, “Cayuga: a high-performance event processing engine,” in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 1100–1102.

[33] M. Balduzzi, V. Ciangaglini, and R. McArdle, “Targeted attacks detection with spunge,” 2013.

[34] FireEye, “Mandiant apt1 report,” https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf, accessed: 26-06-2016.

[35] I. Ghafir, V. Prenosil, J. Svoboda, and M. Hammoudeh, “A survey on network security monitoring systems,” in *IEEE International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE Xplore Digital Library, 2016, pp. 77–82.

[36] J. Svoboda, I. Ghafir, and V. Prenosil, “Network monitoring approaches: An overview,” *International Journal of Advances in Computer Networks and Its Security (IJCNS)*, vol. vol. 5, no. Issue I, 2015.

[37] V. Paxson, “Bro: a system for detecting network intruders in real-time,” *Computer networks*, vol. 31, no. 23, pp. 2435–2463, 1999.

[38] B. Project, “file_over_new_connection event,” https://www.bro.org/sphinx/scripts/base/bif/event.bif.bro.html\#id-file_new, accessed: 01-06-2016.

[39] MrForms, “Mime types list,” http://www.freeformatter.com/mime-types-list.html, accessed: 07-04-2015.