Phishing Attack Detection Using Enhanced Classification Method

N.JAYAKANTHAN
Department of Computer Applications, Kumaragurur College of Technology, Coimbatore

Abstract - Phishing attack is major issue. The attacker steals the information from the user's machine or insert a malware in that system. When the users are attracted to visit the web page the script is executed to carried out the its task. To detect and prevent these attack various tools and methods are developed. Various phishing detection techniques are proposed. But the phishing attacks are not completely detected because the attacker dynamically changing their approaches. In this paper we propose a dynamic approach which solution to any type of phishing attack. Our idea is to analyze the static and dynamic features of the web page to differentiate phishing and genuine web page using machine learning algorithms. The proposed approach correctly detects all phishing and genuine website without any false positive and negatives. It overcomes many drawbacks of the existing signature-based approaches.

Index Terms- Phishing, Web, Malware, Detection

I. INTRODUCTION

The goal of phishing attacks is to steal user identities and credentials. Phishing is the act of attempting to acquire information such as usernames, passwords, and credit card details (and sometimes, indirectly, money) by masquerading as a trustworthy entity in an electronic communication. Communications purporting to be from popular social web sites, auction sites, online payment processors or IT administrators are commonly used to lure the unsuspecting public. Phishing emails may contain links to websites that are infected with malware [1]. Phishing is typically carried out by e-mail spoofing or instant messaging and it often directs users to enter details at a fake website whose look and feel are almost identical to the legitimate one. Phishing is an example of social engineering techniques used to deceive users and exploits the poor usability of current web security technologies. Attempts to deal with the growing number of reported phishing incidents include legislation, user training, public awareness, and the goal of phishing attack

Unfortunately, no existing technical mechanism fully solves this problem. For example, SSL only authen- ticates a web server's IP address or hostname to a browser and protects the communication channel as well. Nonethe- less, it provides no guarantee the HTML files sent by the web server are not misleading. [2]Some schemes are proposed to enable a user to authenticate a server with a priori secu- rity association. However, those schemes are not applicable to websites which a user visits for the first time. Moreover, it requires user awareness of existence of the authentication Nonetheless, if the user is already alerted, a simple URL checking can prevent the phishing attack. (HTTP Transaction) A request sent from the browser to the server and the corresponding response from the server to the browser, both. When a phishing site maliciously claims a false identity, it always demonstrates abnormal behaviors compared to a honest site, which are indicated by some web DOM objects in the page and HTTP transactions.

The advantage of this approach includes that it does not rely on any prior knowledge of the server or users' security expertise, the adversary has much less adaptability since the detection is independent of any specific phishing strategy and it causes no changes on users' existing navigation behavior.

Given the rising threat posed by malicious web pages, it is not surprising that researchers have started to investigate techniques to protect web users. Currently, the most widespread protection [3] is based on URL blacklists. These blacklists (such as Google Safe Browsing) store URLs that were found to be malicious. The lists are queried by a browser before visiting a web page. When the URL is found on the blacklist, the connection is terminated, or a warning is displayed. Of course, to be able to build and maintain such a blacklist, automated detection

mechanisms are required that can find on the Internet web pages containing malicious content.

Unfortunately, for obvious reasons, very few details have been revealed about Google's filter. In particular, the authors only provide examples of three page features and report that they use a proprietary machine-learning framework. The very existence of Google's blacklist provides evidence that the overall system i.e. combining the filter with the back-end analysis tools works.

II. LITERATURE SURVEY

In the last few years, the detection of web pages that launch drive-by-download attacks has become an active area of research and several new approaches have been proposed. Dynamic approaches. Dynamic approaches use honey client systems to visit web pages and determine if they are malicious or not. In high-interaction honey clients, the analysis is performed by using traditional browsers running in a monitored environment and detecting signs of a successful drive-by-download attack (e.g., changes in the file system, the registry, or the set of running processes) [4,7,9,10]. In low-interaction honey clients, the analysis relies on emulated browsers whose execution during the visit of a web page is monitored to detect the manifestation of an attack (e.g., the invocation of a vulnerable method in a plugin) [5,6,8, 11]. Both high- and low-interaction systems require to fully execute the contents of a web page. This includes fetching the page itself, all the resources that are linked from it, and, most importantly, interpreting the associated dynamic content, such as JavaScript code [12]. These approaches usually yield good detection rates with low false positives, since, by performing dynamic analysis, they have complete "visibility" into the actions performed by an attack. The down-side is that this analysis can be relatively slow, because of the time required by the browser (either simulated or real) to retrieve and execute all the contents comprising a web page, taking from a few seconds to several minutes, depending on the complexity of the analyzed page. Scalability issues with today's honeyclient systems (relatively slow processing speed combined with relatively high hardware requirements) motivated our work on a filtering system[13]. Our filter achieves higher performance by forgoing dynamic analysis (e.g., the interpretation of JavaScript code), and relying instead on static analysis only. Static approaches [14] to the detection of drive by download attacks rely on the analysis of the static aspects of a web page, such as its textual content, features of its HTML and JavaScript code, and characteristics of the associated URL. Understand the scientific terms and jargon related to your research work.

III. METHODOLOGY

Now it is the time to articulate the research work with ideas gathered in above steps by adopting any of. In this section, we evaluate both the two-stage correlate-and filter algorithm and the underlying decentralized CIDS architecture. We first conduct a study of the feasibility of using the proposed two-stage correlate-and-filter algorithm by comparing it against a fully centralized scheme. We compare these two schemes in terms of their detection accuracy and message exchange rate using a simulation based on a real-world intrusion dataset. We then evaluate the fully decentralized CIDS architecture by conducting a large-scale experiment on Planet Lab [3] using a real-world worm outbreak dataset. In this section, we first introduce the real-world intrusion dataset used in the experiments. Then we report on the simulation results of the proposed two stage correlate-and-filter algorithm for non-stealthy attack scenarios using the naive threshold selection scheme and in stealthy attack scenarios using the probabilistic threshold selection scheme respectively.

Algorithm
Detection system d_i:
for each time interval **do**
collect raw alerts r_i locally
SA_i = local alert report on d_i
SA_i (correlate-and-filter r_i
for each $v_{ij} = SA_i$ **do**
look up the destination node for p_{ij}
dt = lookup(srcIP of v_{ij})
subscribe(v_{ij} , n_{ij} , d_i) on d_t
end for
end

Our simulation program is written in Java and run on a Sun T-2000 server, which contains an 8-core 1 GHz Ultra SPARC CPU, 16 GB of RAM, and the Solaris

10 operating system. The *dataset* is stored in a *mysql* relational database.

We simulate 2n participating IDSs by varying n from 3 to 7. Each simulated IDS is assigned a unique *provider-id* which is a field of the alert table in the database. Each IDS periodically queries the database using this *provider-id* and a specified time interval. Then these raw alerts will be processed by two different simulated processes: the two-stage correlate and filter process and the centralized correlate-and-filter process. In this simulation, we consider the alert patterns and messages generated by the centralized process as our *gold standard* for calculating the *detection accuracy* and *message exchange rate*. According to performance analysis in normal case, in attack case and in IDS case we observe that DDOS attack affected the network and our scheme is successfully defense the network and also provides the protection against them. In case of attack we observe that the routing load is very high because attacker node are continuously transmit the routing packets to their neighbored and every node in network are reply to attacker node by that heavy congestion is occur. Packet delivery fraction and end to end delay are also goes low, which shows that packets are not deliver accurately and number of dropped data is goes high approximately twice to the normal condition.

IV. IMPLEMENTATION AND ENVIRONMENT

We implemented Dynamic attack detector, and we used it as a for our existing dynamic analysis tool, called Wepawet. It can be used unchanged as a filter for any of the other, publicly available honey client systems. The crawls are seeded by using the current Twitter, Google, and Wikipedia trends as search terms. These trends are used as a basis for the searches because most malware campaigns use Search Engine Optimization (SEO) techniques to increase their ranking in the search engines' results associated with popular terms [15, 16]. Another source of seeds for our crawler is a list of links extracted from a feed of spam emails. The list of links is updated daily and provides us with an average of 2,000 URLs per day. We modified the crawler to be able to set the "Referrer" header when fetching a seed URL. This header has to be set to the search engine from which the seed URL was extracted. This is necessary because some malicious web pages deliver malicious content only when the request appears to be the result of a user clicking on the search results. The crawler fetches pages and submits them as input to Prophiler, which analyzes each page and extracts and stores all the features. Once all features have been extracted from a page, Prophiler uses the models learned in the previous training phase to evaluate its maliciousness. If a page has been identified as likely malicious, it is forwarded to the dynamic analysis tool (Wepawet, in our case). This tool then confirms that the page is indeed malicious, or it flags it as a false positive.

V. CONCLUSION

The proposed mechanism eliminates the need for a centralized trusted authority which is not practical in ADHOC network due to their self organizing nature. The results demonstrate that the presence of a DDOS increases the packet loss in the network considerably. The proposed mechanism protects the network through a self organized, fully distributed and localized procedure. The additional certificate publishing happens only for a short duration of time during which almost all nodes in the network get certified by their neighbors. After a period of time each node has a directory of certificates and hence the routing load incurred in this process is reasonable with a good network performance in terms of security as compare with attack case. We believe that this is an acceptable performance, given that the attack prevented has a much larger impact on the performance of the protocol. We introduce a probabilistic approach to estimate the optimal threshold for local correlation in stealthy attack scenarios. In comparison to a centralized correlate-and filter algorithm, our evaluation using a real-world intrusion dataset shows that our decentralized approach reduces alert messages significantly with little degradation in detection accuracy in most attack scenarios. The proposed probabilistic scheme achieves a significant improvement in detection accuracy compared to a naive threshold selection scheme that uses the same local and global threshold. For future work, we will consider how to optimize the load distribution in the fully decentralized CIDS architecture, and how to make the support threshold adaptive to different types of attack scenarios.

REFERENCES

[1] G. O. C. Estan, S. Savage, and G. Varghese, "Automatically inferring patterns of resource consumption in network traffic," in Proceedings of the 2003 Conference on Applications,Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), 2003, pp. 137–148.

[2] W.-K. "CERT Incident Note IN-99-07," 1999. [Online]. Available: http://www.cert.org/incident

[3] H. "Planetlab testbed," http://www.planet-lab.org.

[4] M. Locasto, J. Parekh, A. Keromytis, and S. Stolfo, "Towards Collaborative Security and P2P Intrusion Detection," in Proceedings of the 2005 IEEE Workshop on Information Assurance and Security, 2005.

[5] E. H. Miller, ―A note on reflector arrays (Periodical style—Accepted for publication), ‖ IEEE Trans. Antennas Propagate., to be published.

[6] J. Y. Hu, D. M. Chiu, and J. C. Lui, "Adaptive Flow Aggregation - A New Solution for Robust Flow Monitoring under Security Attacks," in *Tenth IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2006, pp. 424–435.

[7] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites," in *Proceedings of the 11th International Conference on World Wide Web (WWW)*, 2002, pp. 293–304.

[8] J. "CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks," 1996. [Online]. Available: http://www.cert.org/advisories/CA-1996-21.html.

[9] CERT Advisory CA-2003-20 W32/Blaster worm," 2003. [Online]. Available: http://www.cert.org/advisories/CA-200320.html

[10] S. Gibson, "Distributed Reflection Denial of Service - Description and analysis of a potent, increasingly prevalent, and worrisome Internet attack," 2002. [Online]. Available: http://www.grc.com/dos/drdos.html.

[11] S. S. Katti, B. Krishnamurthy, and D. Katabi, "Collaborating against common enemies," in Internet Measurement Conference, 2005.

[12] S. S. S. Axelsson, "The Base-Rate Fallacy and Its Implications for the Difficulty of Intrusion Detection," in ACM Conference on Computer and Communications Security, 1999, pp. 1–7.

[13] S. G. K. Bhattacharyya and R. A. Johnson, Statistical concepts and methods. New York: Wiley, 1977.

[14] S. G. K. D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, 1997, pp. 654–663.

[15] "Dshield org," http://www.dshield.org.

[16] C. V. Zhou, S. Karunasekera, and C. Leckie, "Evaluation of a Decentralized Architecture for Large Scale Collaborative Intrusion Detection," in *the Tenth IFIP/IEEE International Symposium on Integrated Network Management (IM), Germany*, 2007, pp. 80–89.