REDT: Remote exploitation detection technology for network infrastructure

Xiaokang Yin, Shengli Liu*, Fangfang Jia and Da Xiao

State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, CO 450000, China

*Email: mr.shengliliu@gmail.com

Abstract. An attacker exploits a remote vulnerability to execute arbitrary code on network infrastructure. It seriously threatens the security of network infrastructure. To detect the exploitation attack in a timely and effective way, in this paper, we present REDT, a remote exploitation detection technology based on the effect of exploitation attack. This method determines whether there is an authentication bypass exploitation by detecting the remote authentication process and saves attack traffic to the anomalous traffic database. With the database, we can replay the attack and analyze the exploitation. In the experiment, the detection accuracy rate reached 97.67%. In addition, this method does not rely on the signatures of the vulnerability and shellcode, so it can detect undisclosed vulnerabilities exploitation and provide abnormal traffic for the study of unknown vulnerabilities.

1. Introduction

With the increasing exposure of vulnerabilities in network infrastructure such as routers, switches, firewalls, and Internet of Things (IoT), the security of network infrastructure has attracted widespread attention from security researchers [1-5]. Unlike the conventional software, the firmware running on the network infrastructure can’t directly be patched to repair the vulnerability. The administrator has to download the patched firmware from the vendor support site and replace the original one to repair the vulnerabilities, which is tedious and complicated. Therefore, most administrators rarely upgrade the firmware [6]. So, there are a lot of vulnerabilities in the running firmware. How to efficiently and accurately detect exploitation for embedded systems has grown up to be an important research topic [7-11].

Currently, a lot of studies have been made to detect remote exploitation. Packet vaccine mechanism was introduced into exploitation detection by [12]. This method randomizes address-like strings in packet payloads to carry out fast exploitation detection, vulnerability diagnosis and signature generation. But it also relies on the signatures of shellcode or vulnerabilities. Michalis Polychronakis et al. [13] had studied more than 1.2 million polymorphic code injection attacks targeting production systems, captured using network-level emulation. They focused on the analysis of the structure and operation of the attack code, as well as the overall attack activity in relation to the targeted services. They provide a great reference for extracting exploit signature. Alex Shenfield et al. [14] put forward a novel approach to detection of malicious network traffic using artificial neural networks. This method greatly enhances the accuracy and usability of the intrusion detection system. However, this method is hard to learn the signatures of exploitation from the encrypted network traffic. At the DefCamp 2017
security conference in Bucharest, Romania, Kirill Shipulin [15] had explained how to turn available signatures against the IDS to paralyze its operation. Attackers can construct malformed packets with public exploit signatures and send them to IDS. So, this indicates that remote exploitation detection based on signature is not perfect and can be abused by attackers.

To the best of our knowledge, most of the already known methods are based on the signatures of shellcode or vulnerabilities in essence. To address the problem of lack of samples and hysteresis problem of signature-based exploitation detection on network infrastructure, we proposed REDT, a remote vulnerability exploitation detection system based on remote login authentication bypass. To evaluate the efficacy and performance of REDT, we have implemented it as a proof-of-concept in python. We evaluated REDT in terms of accuracy in identifying the exploitation with the exploits published on the exploit-db. In the experiment, REDT obtained an average accuracy 97.67%, and an average false alarm rate 2.75%. What’s more, the novel approach can detect the zero-day exploitation attack and discovery undisclosed vulnerability on network infrastructure. It also saves the exploitation network traffic into abnormal traffic database. The database can be used to analyze the attack and extract vulnerability information.

The contributions of this paper are as follows:
1. We proposed a remote exploitation detection technology for network infrastructure.
2. We analyzed the remote login process and the difference between normal login and remote exploitation.
3. We designed and implemented REDT in python as a proof-of-concept. We verified it in LAN environments with the public exploits on the internet.

The remainder of this paper is structured as follows. Section 2 reviews the challenges of exploitation detection. Section 3 describes the proposed approach to authentication bypass exploitation attack. Section 4 presents the experiment and evaluates REDT with public exploits. Finally, section 5 provides the conclusion.

2. Challenge
In this section, we describe the challenges faced in exploitation detection.

2.1. Difficult to extract the vulnerability signature
The triggering conditions of different vulnerabilities are various. And different vulnerabilities have different triggering conditions. In addition, after the new vulnerabilities are discovered, it still takes some time to study and analyze the vulnerabilities to obtain much more accurate characteristics.

2.2. Exploit diversity
Even for the same purpose of the attack, the exploits are also different. The triggering conditions of the same vulnerability are also diverse. For example, Cisco IOS IP Options vulnerability, CVE-2007-0480, can be triggered with up to seven types of packets for this vulnerability. Therefore, attackers could develop a great number of exploits to exploit this vulnerability in different routers. For network infrastructure, such as Cisco routers, switches and IoT, due to differences in different versions, resulting in different addresses, functions, etc., exploits cannot be directly generated by using tools like Metasploit. Attackers have to develop shellcode for each version manually. So, it is difficult to extract the signature of exploit developed manually.

2.3. Obfuscation exploit signature with encryption
Figure 1 presents the structure of common shellcode. The attacker can use a simple encryption method (XOR, NOR operation, etc.) to confuse the features. Figure 2 shows the structure of the encrypted shellcode. The attacker encapsulates the decoder at the front end of the shellcode and shellcode decodes the encrypted payload through the decoder. After the encryption, the original characteristics of the shellcode no longer exist. Qinghua Zhang et al. [16] proposed an approach to detect self-decrypting exploit code. This proposed method combines static analysis and emulated instruction
execution techniques to detect such highly obfuscated polymorphic exploit code. But non-exploit code that uses code obfuscation, code encryption, and self-modification for legitimate purpose will interfere the detection process resulting in false alarm.

![Figure 1. Structure of shellcode [16].](image1)

2.4. Bypasses IDS using signatures
As mentioned in the previous section, available IDS signatures can be turned against the IDS to paralyze its defenses. Attackers construct a large number of malformed packets containing public vulnerabilities and exploits signatures and send them to IDS, which leads to IDS paralysis. So, the real attack can bypass the defense of IDS.

3. Detection process
In general, the entire process of exploitation attack on network infrastructure could be divided into five steps [17]: a) scan device and collect fingerprint, b) bypass remote login authentication and gain the control, c) check and modify the configuration, implant backdoor or update firmware and so on, d) clear attack traces, e) restore remote login authentication. The following heuristic experiences exist in the network environment: A network infrastructure configured correctly with remote login service on must have the login authentication mechanism. Attackers often use exploit to attack the network infrastructure for the purpose of gaining the control. After the network infrastructure under attack, attacker can login without password or with arbitrary password. In order to achieve the purpose of the attack and conceal their own aggressive attack, attackers will complete the attack within a short time.

3.1. Normal and abnormal authentication
This paper takes telnet as an example to illustrate the remote login authentication process and analyze the difference between normal and abnormal login. Telnet establishes connection and transfers data with the TCP protocol. During the authentication phase, the server requests login authentication password from the client and the client must answer it with correct username, password. After authentication, the client sends commands to the server, and the server executes the commands and returns the result to the client.

After attackers attacks the network infrastructure with login authentication bypass exploitation, the telnet login authentication process is destroyed. We analysed the public exploits for network infrastructure and found that the login authentication bypass attacks can be divided into two types: login authentication with an arbitrary password and login authentication without password. Arbitrary password login still contains the authentication request process, but the attacker can input any password or directly enter the ‘enter’ to login the network infrastructure. Login authentication without password returns a command shell once the telnet connection is established without asking for password. And the entire process misses the authentication phase. After login to the network infrastructure, the attacker can totally control it, view and modify the configuration, implant backdoor, replace firmware and so on.
3.2. Detection algorithm and process

We divide the entire remote exploitation detection system into five modules: traffic processing, authentication anomaly detection, anomalous traffic database, attack alert and restore attack process. Figure 3 shows the detection processes.

Traffic processing module is placed at the middle of internet and infrastructure device. It captures all the traffic passing through them, stores the network traffic into the traffic buffer queue and authentication traffic queue. Traffic collection and filtering module extracts the authentication-related packets with certain rules and forwards them to the authentication anomaly detection module. The filter criteria are based on the IP address and TCP port. Since our goal is to detect attacks against the infrastructure device, we need the IP address of infrastructure device to be protected, which can be one or more. So, the traffic buffer queue stores all the data traffic that communicates with the protected device and the authentication traffic queue stores the remote authentication login traffic. As we all known, currently, the remote login protocol mainly is Telnet, SSH and HTTP/HTTPS for infrastructure device not containing the private protocols. Those protocols have specific TCP port. The port, for Telnet, is 23, for SSH, is 22, and for HTTP/HTTPS is 80 or 443.

The Authentication anomaly detection module analyses the authentication traffic and judges whether the login authentication process is broken or not. The authentication process starts from the three-way handshake of TCP. As we mentioned above, when there is a remote authentication bypass exploitation, the authentication login process will be broken, and attacker can login the device without password or any password. To accurately detect the any password login attack, we’d better put the correct remote login password into REDT, so, it will know whether people login the device with correct password or not. Then we can know whether there is a remote authentication bypass exploitation or not.

When an attack is detected, the attack alert module will send an alert message. The message contains the attack IP address, time and the IP address of attacked device. If REDT is integrated into IDS, it can be more intelligent, for example, send an alert message to administrator with SMS or Email.

The abnormal traffic database stores the suspected attack traffic. When the authentication anomaly detection detects the authentication process is broken, it will notify the traffic buffer queue to stores all the traffic in the queue into the abnormal traffic database. In order to capture all attack traffic, there is delay storing the traffic to collect more attack packets. Then we can know what the attacker does after login the device and provide traffic to analyze the attack means. That traffic may contain the attack traffic with exploit. From that traffic, we can restore the exploit and vulnerability that attacker used to bypass the authentication login. If the vulnerability is not disclosed, we will discover the zero-day vulnerability.

![Figure 3. Exploitation detection process based on authentication bypass.](image-url)
The restore attack process is an offline program and it is to restore the attack process with the traffic stored in the abnormal traffic database. It is used to support the researcher to analyse the attack traffic and verify whether the attack used a published vulnerability or zero-day vulnerability.

4. Experiments and analysis
In this section, we first present the environment used in the experiments in detail, and then analyze the experiment result and evaluate the proposed approach.

4.1. Environment
To evaluate the proposed approach, we use an automation test program to randomly perform simulated exploitation attack, normal login, and error login. Within a period of time, the automation test program randomly executes one operation and records the operation and operation time, then writes them into the file named attack_log. The attack_log contains the operation time and the event, for example {2018.08.08.15:20:15 attack 192.168.1.1 router}. REDT will saves the detection result into file named de_attack_log with detection time and IP of attackers. The de_attack_log contains the detection time, the IP addresses of attack and the IP addresses of attacked devices, for example {2018.08.08.15:20:16 192.168.1.168 attack 192.168.1.1}. By comparing the time between attack_log and de_attack_log, we can judge the detection whether is correct or not. But in the experiment, in order to capture the communication traffic among all devices, one of the ports of switch is configured as a listening port. So, all traffic forwards to the listening port. Table 1 shows the vulnerabilities used in the experiments. The protocol in the table shows the device can’t correctly deal with this protocol leading to the vulnerability.

Table 1. Vulnerabilities used in this experiment.

| System         | CVE                  | Vulnerability Type(s)                  | Protocol | CVSS Score |
|----------------|----------------------|----------------------------------------|----------|------------|
| Cisco IOS      | CVE-2007-0480        | Denial of Service, Overflow, Execute Code | IP Option | 10.0       |
| Cisco IOS      | CVE-2007-2586        | Execute Code, Overflow                 | FTP      | 9.3        |
| Cisco IOS      | CVE-2017-6736 (36-44)| Execute Code                          | SNMP     | 9.0        |
| Cisco IOS Cisco IOSXE | CVE-2017-3881 | Execute Code                          | CMP      | 10.0       |
| Cisco ASA      | CVE-2016-6366        | Execute Code, Overflow                 | SNMP     | 8.5        |

Figure 4. Network topology in the experiment.
Figure 4 presents the experiment network topology. In this topology, Host 1, 2, 3 and 4 communicate with each other with the local network software to simulate the real network. The specific operation is as follows: 1. Host A runs the exploitation detection system and records the time and attack traffic when detects an attack happened. 2. Host B attacks router, switch and ASA firewall in turn with public exploits and records the attack time. 3. Host C logins the network infrastructure, modifies the configuration, and simulates unauthorized users’ login with error password. After the experiment, compare file attack_log with file de_attack_log.

4.2. Analysis of results
There are 473 times automated operation with program during the experiment, including exploitation attack, normal login and error login. Table 2 shows the experiment data. Among 473 times operations, there are 218 times exploitation attacks, 215 times normal login and 40 times error login. Comparing the attack_log with de_attack_log, we found that the exploitation detection system correctly detected 214 times attack, regarded normal login as attack with 6 times and considered error login as attack with 1 time by mistake.

| Table 2. Experimental data. |
|-----------------------------|
| Times Operation | Actual Operation | Attack detected |
| Exploitation attack | 218 | 214 |
| Normal login | 215 | 6 |
| Error login | 40 | 1 |
| Total | 473 | 221 |

| Table 3. Result of exploitation detection. |
|-----------------------------|
| Accuracy | 97.67% |
| Precision | 96.83% |
| Recall | 98.17% |
| False Alarm Rate | 2.75% |

Table 3 presents the result of remote exploitation detection system. The experimental result shows that this method can effectively detect the authentication bypass attack with accuracy 97.67% on network infrastructure, and in the experimental, the attack alarm can be issued as long as the attacker logins to the device. And the detection latency is very small about second level. Since we are just concerned about the authentication process packets and have filter the packets with TCP port, actually the packets that we need to analyze is very small.

This method is different from the signature-based remote exploitation detection method. It does not consider the specific vulnerability characteristics. It is not necessary to spend a lot of time analyzing the vulnerabilities and exploits. And it does not care about how the shellcode is constructed and whether the shellcode is encrypted. As long as the exploitation attempts to bypass the login authentication, it can accurately detect the exploitation attack. What’s more, it can discover that an attacker uses an undisclosed vulnerability to launch zero-day attacks on network infrastructure.

4.3. Inadequacy
As REDT is based on the abnormality of remote authentication login process after being attack, it can detect the remote authentication bypass exploitation. Due to the integrity of the login authentication process, the loss of authentication packets will lead to false alarm. For the DoS attack or other type exploitation attack, it is unable to detect. For login authentication bypass with an arbitrary password, the detection effect of this method is limited, because the authentication process is not destroyed, that
is, the login is no different from the administrator’s login under normal circumstances. So, we need to preload the correct password into the program and judge the password that has logged in the device is the correct password or not.

5. Conclusion

In this paper, we proposed a remote exploitation detection method based on the exploitation effect which bypasses the remote login authentication on network infrastructure, such as, routers, switches and firewalls. We implemented this method in python and set up a network environment to evaluate it. This approach accurately detected login authentication bypass attack on network infrastructure in time and reached accuracy 97.67%. What’s more, this method has the ability to detect the undisclosed vulnerability exploitation on network infrastructure. This approach also has the characteristics of a small load, low detection delay and does not depend on specific network infrastructure. When a new vulnerability or exploit is published, it is no need to analyze the vulnerability to extract the signature and update the exploitation detection system. The exploitation detection system also stores the exploitation traffic into abnormal traffic database to help security researcher replay the attack behavior and analyze the triggering conditions of new vulnerabilities and exploits.

Acknowledgments

The authors would like to thank the funding of the National Key R&D Plan under Grant 2016YFB0801505 and 2016YFB0801601.

References

[1] Cui, Ang, and Salvatore J. Stolfo. 2011. Defending embedded systems with software symbiotes. International Workshop on Recent Advances in Intrusion Detection. (Berlin, Heidelberg, 2011), RAID’11, Springer-Verlag, pp. 358–377.

[2] Fournaris, A. P., Pocero Friaile, L. and Koufopavlou, O. 2017. Exploiting Hardware Vulnerabilities to Attack Embedded System Devices: a Survey of Potent Microarchitectural Attacks. Electronics, 6(3), 52, 2017, doi: 10.3390/electronics6030052.

[3] Cui, Ang, Michael Costello and Salvatore J. Stolfo. 2013. When Firmware Modifications Attack: A Case Study of Embedded Exploitation. NDSS. 2013.

[4] D’Orazio, Christian J., Kim-Kwang Raymond Choo and Laurence T. Yang. 2017. Data exfiltration from Internet of Things devices: iOS devices as case studies. IEEE Internet of Things Journal 4.2, 2017, pp. 524-535, doi: 10.1109/JIOT.2016.2569094.

[5] Granjal, Jorge, Edmundo Monteiro and Jorge Sá Silva. 2015. Security for the internet of things: a survey of existing protocols and open research issues. IEEE Communications Surveys & Tutorials 17.3, 2015, pp. 1294-1312, doi: 10.1109/COMST.2015.2388550.

[6] Lindner, Felix. Cisco IOS router exploitation. Black Hat USA, 2009. Accessed: April 15, 2018. [Online] Available: https://www.blackhat.com/presentations/bh-usa-09/LINDNER/BHUSA09-Lindner-RouterExploit-SLIDES.pdf

[7] Davi, Lucas, Patrick Koebertl and Ahmad-Reza Sadeghi. 2014. Hardware-assisted fine-grained control-flow integrity: Towards efficient protection of embedded systems against software exploitation. Proceedings of the 51st Annual Design Automation Conference. ACM, 2014, doi: 10.1145/2593069.2596656.

[8] Francillon, Aurélien. 2009. Attacking and protecting constrained embedded systems from control flow attacks. Diss. Institute National Polytechnique de Grenoble-INPG, 2009.

[9] Papp, Dorothy, Zhendong Ma and Levente Buttyan. 2015. Embedded systems security: Threats, vulnerabilities, and attack taxonomy. Privacy, Security and Trust (PST), 2015 13th Annual Conference on. IEEE, 2015, doi: 10.1109/PST.2015.7232966.

[10] Shao, Zili, et al. 2006. Security protection and checking for embedded system integration against buffer overflow attacks via hardware/software. IEEE Transactions on Computers 55.4, 2006, pp. 443-453, doi: 10.1109/TC.2006.59.
[11] Parameswaran, Sri and Tilman Wolf. 2008. Embedded systems security—an overview. Design Automation for Embedded Systems 12.3, 2008, pp. 173-183.

[12] Wang, X., Li, Z., Xu, J., Reiter, M. K., Kil, C. and Choi, J. Y. 2006. Packet vaccine: Black-box exploit detection and signature generation. In Proceedings of the 13th ACM conference on Computer and communications security, ACM, 2006, pp. 37-46, doi:10.1145/1455518.1455523.

[13] Anagnostakis, K. G., and E. P. Markatos. 2009. An empirical study of real-world polymorphic code injection attacks. In Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats, 2009.

[14] Shenfield, A., Day, D. and Ayesh, A. 2018. Intelligent intrusion detection systems using artificial neural networks. ICT Express, 2018, doi: 10.1016/j.ictex.2018.04.003.

[15] Kirill Shipulin. 2018. Turning IDS signatures against an IDS itself: a new evasion technique. DefCamp9, 2018. Accessed: May 23, 2018. [Online] Available: https://def.camp/speaker/kirill-shipulin/

[16] Zhang, Q., Reeves, D. S., Ning, P. and Iyer, S. P. 2007. Analyzing network traffic to detect self-decrypting exploit code. In Proceedings of the 2nd ACM symposium on Information, computer and communications security. ACM, 2007, pp. 4-12, doi: 10.1145/1229285.1229291.

[17] Erickson, Jon. 2008. Hacking: the art of exploitation. No starch press, 2008. p.140