Evaluation of CEBRAS: A Case-based Reasoning Adversary Emulation System

Cedric Klosa[1,2], Jakob Michael Schoenborn[1,2,3], and Klaus-Dieter Althoff[2,3]

[1] Exploit Labs UG
Eschborn Business Park
Mergenthaler Allee 15-21
65760 Eschborn, Germany
klosa@exploitlabs.de
[2] University of Hildesheim
Samelsonplatz 1
31141 Hildesheim, Germany
schoenborn@uni-hildesheim.de
[3] German Research Center for Artificial Intelligence (DFKI)
Trippstadter Str. 122
67663 Kaiserslautern, Germany
kalthoff@dfki.uni-kl.de

Abstract. "*Never change a running system.*". This might be true for certain circumstances, but in general, not applying new patches to computer systems usually leads to security issues - especially when the system is (indirectly) connected to the internet. One of the usual first steps of an attacker is to scan open ports for version numbers and accessible services to establish a foothold inside their targeted system. We provide a system called CEBRAS which uses a combination of the scanning tool Nmap and the case-based reasoning methodology to identify possible vulnerabilities and to raise the awareness to patch these. To identify these vulnerabilities, we use a preliminary simple case structure with heavy weight on the service description as the most dominant factor. We compare the service descriptions by using the Jaro-Winkler distance metric since service descriptions usually differ at the later positions, making this a perfect fit to our needs. Using the Exploit-DB database, we achieve first promising results, but also remain with future work to further improve our accuracy.

Keywords: Case-Based Reasoning · Adversary Emulation · Penetration Testing

1 Introduction

Due to the increasing level of digitalisation in almost any domain, for example, medical, industry, daily life, we gain more access to technology to gather data and each device is becoming more and more connected to other devices. Nevertheless, the aspect of security is often left behind during the development

cycle of those devices, despite the attack vectors are very well known and have not changed drastically during the last years [1]. Additionally, the increasing amount of attacks especially by using (spear-)phishing e-mails is noteworthy [2] and furthermore has been investigated by using case-based reasoning (CBR) as a detection method by Lansley et al. [3]. Phishing e-mails try to lure the user to click a certain link pointing to a malicious website - developed by the attacker - to gain more information about the victim, for example, sensitive data stored in cookies or sessions. These e-mails usually do not have a certain target in mind but rather send a massive amount of e-mails, hoping *someone* clicks on their link. In contrast, spear phishing e-mails are targeting individual persons by using additional information. For instance, the correct first- and lastname, the date of birth - which can often be found in social media accounts - and other information are used to increase the chance that the targeted victim might think that it is a valid e-mail and that the victim clicks on the link.

This contribution aims to increase the awareness of security issues by focusing on the issue of systems which have not been patched properly. In an industrial setting, companies refrain from patching their systems due to fear of incompatibility, for instance, due to changes in the API of a program, and rather follow the rule of thumb "Never change a running system". This behaviour can be seen by an article of Jeff Dunn which suggests the market share of computers using Windows XP with 7 % despite Windows XP has not been officially supported since 04/08/2014 and therefore contains multiple unpatched security issues, such as allowing the well-known incident of the ransomware "WannaCry" to happen and spread [4]. Reevaluating the underlying data provided by netmarketshare for 10/2018 up until the point of writing, the market share has been decreased to 2,83 % which are in relation still millions of devices [5]. On the other side, companies are increasingly aware of security issues and that they might not have caught every possible bug. This led to the so-called bug bounty programs. These programs define the scope where every individual is allowed to try to find security issues. The amount of the bounty is usually defined depending on the risk level, the likelihood of exploitation, and the quality of the written report, for example, Microsoft offers up to 300.000 USD for issues in their Azure services [6].

We propose the system "Case-Based Reasoning Adversary Emulation Systems" (CEBRAS) to reveal known security issues depending on the operating system (OS), the running processes, and services. The targeted audience is on both ends which we described before: On the one hand, the system can be used by individuals to assess their current setup regarding known vulnerabilities. On the other hand, bug bounty hunters can be supported by managing their knowledge bases and provide them with similar attack vectors to find novel bugs. As a disclaimer, we do not claim to provide complete security (since this state is not feasible to reach) but rather try to make the average working space more secure. We begin by describing the underlying architecture, the information retrieval and interpretation in section 2, move over to the setup of the evaluated testing

environment in section 3, discuss the results in section 4 and close with pointing towards future work in the last section 5.

2 Architecture

The system contains two main components: The first component is considering the information retrieval (scanning). To retrieve information, we scan each software port of the targeted computer. A software port is a communication interface of the computer ranging from port 0 to 65535 and allows the communication between two computers. For example, a well-known port is 443 which is the standard port for a secure internet connection (https-protocol). To scan all these ports automatically, we use the tool Network Mapper (Nmap) [7]. Without going too much into technical detail, the multitude of possible scans allows us not only to see which services are available, but additionally see the exactly used version - if not manually prevented by a well-configured firewall. This also assumes that we do not need physical access to the network but rather use our internet access. However, having physical access to a local network drastically increases the accuracy and possibilities of information gathering. The gathered data will be stored in an XML-file, which is easily interpretable for a human and a machine due to its hierarchical structure.

The second component considers the interpretation of the gathered information by using CBR. CEBRAS uses the structural CBR approach by using attribute-value-pairs since this is the most intuitive way to interpret the provided structural information, for instance, port = 80. The case problem contains three attributes: The service description, the port number, and the operating system. The service description and the operating system is compared by using the token-based Jaro-Winkler distance function with P as the length of the longest common prefix of s_1 and s_2 and $P' = max(P, 4)$ [10]. For clarification, we transfer the distance function to a similarity function by calculating $Jaro_{sim}(s_1, s_2) = 1 - Jaro(s_1, s_2)$.

$$Jaro_{sim}(s_1, s_2) = \begin{cases} 0, & \text{if m=0} \\ \frac{1}{3} \cdot \left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right), & \text{otherwise} \end{cases}$$

with

$\|s_1\|, \|s_2\|$	length of the input strings
m	number of matching characters
t	half the number of transitions

As an example, we compare s_1 = ProFTPD 1.3.6a, $|s_1| = 14$ and s_2 = ProFTPD 1.3.3, $|s_2| = 13$. We compute $t = \left\lfloor \frac{m}{2} \right\rfloor - 1$ by comparing s_1, s_2 and counting the number of matching characters $m = 12$, thus $t = \left\lfloor \frac{12}{2} \right\rfloor - 1 = 5$.

This results in overall in

$$Jaro_{sim}(s_1, s_2) = \frac{1}{3}\left(\frac{12}{13} + \frac{12}{14} + \frac{12-5}{12}\right) = 0.78785$$

We use the result of the Jaro similarity for the Jaro-Winkler similarity.

$$\begin{aligned} Jaro - Winkler_{sim}(s_1, s_2) &= Jaro_{sim}(s_1, s_2) + \frac{P'}{10} \cdot (1 - Jaro_{sim}(s_1, s_2)) \\ &= 0.87271 \end{aligned}$$

Since the Jaro-Winkler similarity metric locates on which position the dissimilarity occurs by punishing dissimilarities on earlier positions, this distance metric fits very well to our application domain, since different version numbers usually occur at the later positions, for example, ProFTPD 1.3.6a and ProFTPD 1.3.3. Thus, we retrieve a better result by using Jaro-Winkler instead of the Jaro similarity by a more favorable rating due to matching characters at the beginning of the strings. Intuitively, one might argue that these two versions are not that dissimilar to each other, but especially in the IT-security domain, attack vectors (or the so-called "payloads") are usually written for the current, specific version of a service and might not be anymore applicable for the next updated version. Thus, dissimilar terms are heavily punished by a strong descending similarity.

Regarding the port number, a system administrator might shuffle the ports which certain services use to further obfuscate the underlying structure of the network, thus, the similarity of the port number is treated as an integer polynominal with 2. The operating system is treated as a symbol matrix, containing the values "linux", "macOS", "windows" while linux is more similar to macOS than to windows and vice versa. To fill the case base, we use the publicly available database Exploit-DB which as the point of writing hosts 41.641 entries [8]. Exploit-DB is a well-known service where novel exploits will be published so that administrators can read about them and initiate the appropriate countermeasures. The exploits are usually reported, maintained, and evaluated by security experts - mostly by members of the training company Offensive Security [8]. These results can be filtered by the application domain to show only exploits which do not require physical access to the network (remote) or can be categorized as a web application vulnerability (webapps), which we will primarily consider in the following. These are currently 21.306 entries, that is, 21.306 cases. Additionally to the raw information, Exploit-DB also provides the corresponding exemplary exploit to further validate the result.

Fig. 1 shows the process of the two components described above. After we have gathered and sanitized our information collected in step 1-4, we begin to operate on a case-based level. For building and maintaining our CBR system, we used myCBR which has been developed by the German Research Center for Artificial Intelligence in cooperation with the University of West London [9]. We used our proposed case structure in step 5 by using the gathered information to create the query cases. During the Retrieve-phase of the CBR cycle, we gather the data from Exploit-DB and use a custom parser to bring these information

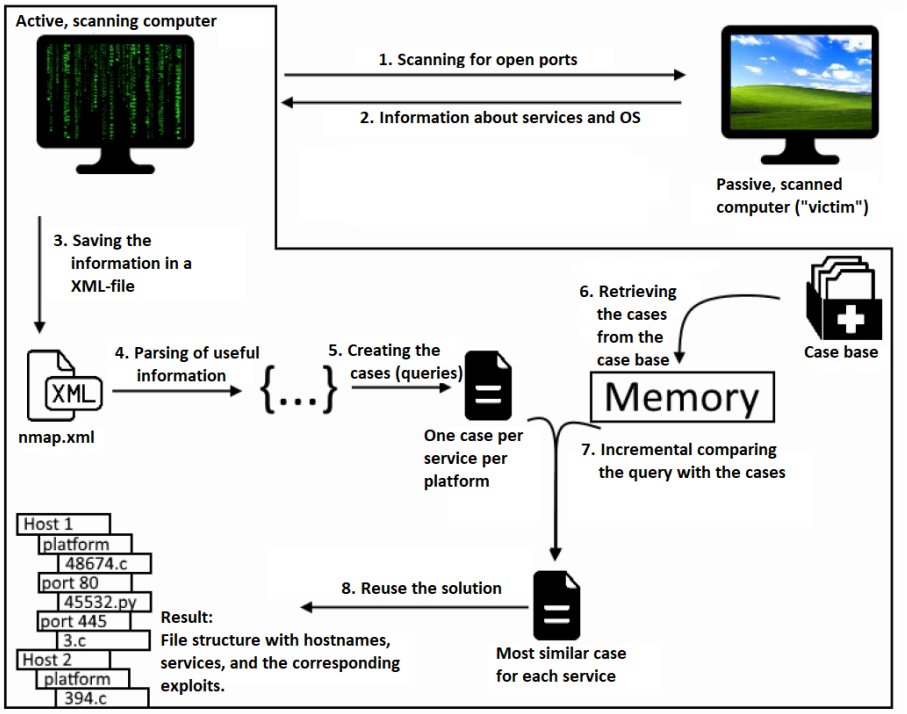

Fig. 1. Architecture and process of CEBRAS.

```
-------------------------------------
CasE-Based Reasoning Adversary System (Cebras)

IP-Range: 192.168.2.0-255
Host-System: Linux-Based
-------------------------------------
Delete previous Scans and Results...

-------------------------------------
Start Nmap to scan the ip Range
Scan complete! Saved in results/nmap.xml
-------------------------------------
Informationgathering completed.
Searching for Vulnerability...

Host-ID: 1:
IP: 192.168.2.92
Platform: Windows
        Found Port: 80
        ->Service: http
        Found Port: 135
        ->Service: Microsoft Windows RPC
        Found Port: 139
        ->Service: Microsoft Windows netbios-ssn
```

```
-------------------------------------
Searching for a similar case to: [['Microsoft HTTPAPI httpd 2.0', '5357', 'Windo
ws']]
No Case found!
Searching for a similar case to: [['ProFTPD 1.3.3c', '21', 'Linux']]
        The most similar case is:
        Index in CaseBase: 1200
        Attributes:
                Description: ProFTPd 1.3.3c - Compromised Source Backdoor Remote
 Code Execution
                Port: 21
                Platform: linux
        Solution:
                Path: exploits/linux/remote/15662.txt
        Similarity: 1.0

-------------------------------------
Searching for a similar case to: [['Radicale calendar and contacts server', '808
0', 'Linux']]
No Case found!

-------------------------------------
Done.
```

Fig. 2. Output of CEBRAS. Left: Start of the system and scanning; right: Query and retrieval of the most similar case.

into our case structure to build our case base. After building the case base and retrieving the most similar cases based on the similarity metrics described above, we move into the reuse phase by suggesting our case to the user in step 8. Fig. 2 depicts the extracted information by step 1-4 in addition to the whole retrieval process of searching for a similar case up to presenting the most similar case by providing the index in the case base, the attributes (and corresponding payload), the path to the solution text file, and the calculated similarity. Revise and retain have not been evaluated, yet. However, for the retain phase we do not expect huge corrections regarding maintainability, since we do not expect that a lot of new cases will be generated (which, to emphasize, is not our primary goal) due to Exploit-DB providing a very good foundation. However, Exploit-DB is not (and can not be) complete, so saving cases is still an important step for learning. As a result of the most similar case, for each scanned host we receive a file structure with the platform or service and the corresponding vulnerability and exploit (see the result of Fig. 1). At this point, we assume the targeted user to be familiar with vulnerabilities and exploitation. A more "average-user"-friendly version is a point for future work.

3 Testing environment

Fig. 3 depicts the testing environment. Each machine is running at least one service while computers with IP addresses ending on 90-98 are running services with known vulnerabilities, ranging from certain FTP services over malicious e-mail transport systems to outdated Apache servers. The vulnerabilities have been randomly chosen out of the pool of Exploit-DB. As most network administrators can easily identify, the computers are all within the same subnetwork.

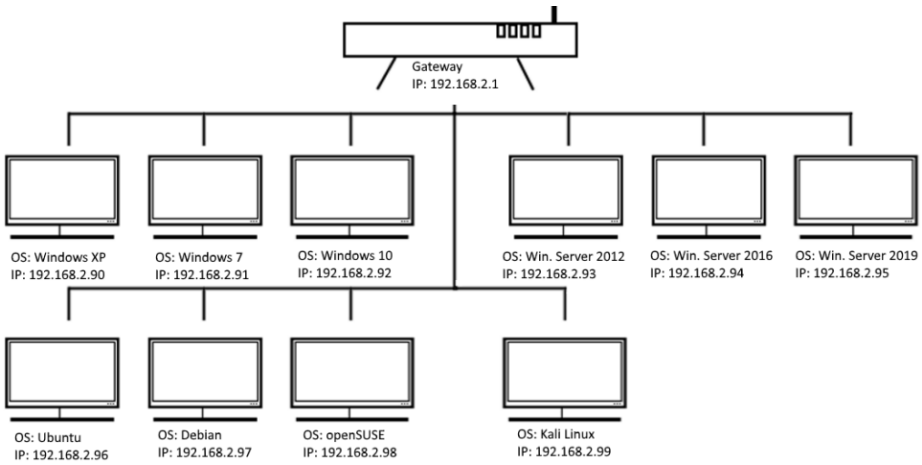

Fig. 3. Testing environment. A total of 10 computers (virtual machines) with the attacker using the Kali Linux OS (IP: 192.168.2.99).

We have kept the network as simple as possible to provide a first proof of concept regarding scanning for vulnerabilities and exporting the results. This assumes we have already passed a firewall and made our way within a subnetwork. As a point for future work, we might include exploring other subnetworks.

4 Evaluation

Using our presented test environment and the results of the Nmap scan by extracting the information from the XML file, we scanned 32 ports whereas 10 of them are vulnerable. CEBRAS flagged 21 services correctly. 6 services could not be recognised, and 5 services have been later identified as false-positives (we either detected a wrong program or the corresponding version was incorrect), which leads to an accuracy of 65 %. We added another step and improved the information retrieval by correcting the retrieval result manually (out of curiosity: what would happen if Nmap would not fail - or if we can find another, better scanner). CEBRAS flagged 24 services correctly, but for the remaining 8 services, we still went for the wrong solution, thus, leading to a slightly better, but not drastically better result.

We consider four points for further improvement:

1. Acceptance of results
   As briefly discussed above, in case we do not retrieve 100 % similarity, the border for accepting the most similar case still needs to be set very high, at least above 90 % since small differences, for example, in the version, can have a huge impact. The exact value varies depending on the underlying OS and vulnerability.
2. Missing adaptation knowledge
   Since this is our very first step into this domain, we are still lacking some domain expert knowledge to adapt the solution of similar problems. Yet, it has to be determined if this is even applicable given the very specific problem-solution pairs. Additionally, the adaptation knowledge regarding the similarity of attributes needs to be reevaluated.
3. Training of weights
   As we are using a very reduced amount of attributes, the training of the global weights becomes more important. We started with using a weighted sum of (2,1,1) corresponding to case structure (service, port, OS). Preliminary training showed that (9,9,2) led to better results, however we ended up with (10,1,1) as the best weights which leads to the service description as a very dominant factor.
4. False-positives
   Despite Nmap being one of the best scanning tools, some services do not publish their information which leads us to treat this service as unknown and possibly vulnerable.

5 Conclusion

We discussed our first step into the IT security domain by using CBR to detect outdated and thus possibly vulnerable services. We used common ways of penetration testing to identify those weaknesses as a real attacker would probably do as well. By using our scanning results, we still retrieved a few false-positives and additionally, for some services we applied the wrong solution despite the service has been identified correctly, leading to an overall precision of 65 %. This is highly questionable whether this result is sufficient, but the results we retrieved during our training phases leave us with multiple promising paths to further decrease the rate of false-positives and further increase the overall precision. Especially moving into more detail with comparing the service description seems to be a promising alternative. Furthermore, we might add more attributes in the future to reach a higher complexity of the case structure and thus benefiting more of the CBR approach in general. As a closing note, it might also be beneficial to split the program between the two targeted audiences and develop those separately to further consider the needs of the corresponding stakeholders.

References

1. Wichers, D. (2015): *OWASP Top Ten: Project Activity?* Mailing List, Website. Last validation: 10/15/2019. URL: `https://lists.owasp.org/pipermail/owasp-topten/2015-June/001310`, 2015.
2. Issac, B., Chiong, R., Jacob, S. M. (2014): *Analysis of Phishing Attacks and Countermeasures.* IBIMA, Bonn, Germany, ISBN 0-9753393-5-4, pp.339-346, 2014.
3. Lansley, M., Polatidis, N., Kapetanakis, S., Amin, K., Samakovitis, G., Petridis, M. (2019): *Seen the villains: Detecting Social Engineering Attacks using Case-based Reasoning and Deep Learning.* Paper presented at Workshops Proceedings for the Twenty-seventh International Conference on Case-Based Reasoning, 2019.
4. Dunn, J. (2017): *A huge number of PCs still use ancient Windows software that puts them at risk..* Business Insider. Website. Last validation: 10/15/2019. URL: `https://www.businessinsider.com/how-many-people-use-windows-xp-chart-2017-5?IR=T`, 2017.
5. Netmarketshare (2019): *Operating System Share by Version.* Website. Last validation: 10/15/2019. URL: `https://www.netmarketshare.com/operating-system-market-share.aspx`. 2019.
6. Microsoft (2019): *Microsoft Bug Bounty Program.* Website. Last validation: 10/15/2019. URL: `https://www.microsoft.com/en-us/msrc/bounty`. 2019.
7. Nmap (2019): *Network Mapper.* Website. Last validation: 10/15/2019. URL: `https://nmap.org/`. 2019.
8. Exploit-DB (2019): *Exploit-Database.* website. Last validation: 10/15/2019. URL: `https://www.exploit-db.com/`. 2019.
9. myCBR (2019): *myCBR.* Website. Last validation: 10/21/2019. URL: `http://www.mycbr-project.org/index.html,2019`
10. Cohen, W. W., Ravikumar, P., Fienberg, S. E. (2003): *A comparison of string distance metrics for name-matching tasks.* In Proceedings of the 2003 International Conference on Information Integration on the Web (IIWEB'03), Subbarao Kambhampati and Craig A. Knoblock (Eds.). AAAI Press 73-78. 2003.