Web Sitesi Saldırılara Karşı Hızlı Çözüm Üreten Bir Uygulama

Murat Arslan¹, Burak Çarkçı¹, and Murat Erten²

¹ İzmir Yüksek Teknoloji Enstitüsü
² Bâkircay Üniversitesi
{muratarslan,burakcarikci}@std.iyte.edu.tr;{murat.erten}@bakircay.edu.tr


Anahtar Kelimeler: Ağ güvenliği, uygulama güvencesi, siber saldırı istihbaratı.
An Application for Quick Response to Website Attacks

Murat Arslan\textsuperscript{1}, Burak Çarıkçı\textsuperscript{1}, and Murat Erten\textsuperscript{2}

\textsuperscript{1} Izmir Institute of Technology
\textsuperscript{2} Bakırçay University
{muratarslan,burakcarikci}@std.iyte.edu.tr;{murat.erten}@bakircay.edu.tr

Abstract. When web pages undergo attacks, it is just as important to collect information and statistics regarding the attack as protecting the site. In this work a system was developed where a clone of the web page being attacked is created using docker and the attacks are diverted there. While the attacker thinks that he is performing an attack on the web page, statistical information is collected regarding his/her activities but the web page continues to function normally. The system was successfully implemented and tested, and the results are presented.

Keywords: network security · application security · cyber threat intelligence.

1 Introduction

In our changing and developing world, everything is getting connected with networks. Websites and application servers have the highest percentage in that space. Hence, website security is becoming much more critical parallel with these improvements and, at the same time, hackers become equipped with the most capable tools and talents. Similar tools and approaches are required on the side of security domain to protect the sites and to gather as much intelligence on the attacks as possible.

Firewalls are one of the tools used to stop attackers from penetrating into servers and networks. Web Application Firewalls (WAF) do not deceive hackers but if an attacker is caught by WAF, payloads can not reach the web application because he is banned or that payload is banned. Hackers, on the other hand, detect the WAFs and apply developed techniques to bypass these [1].

Intrusion Detection/Prevention Systems (IDS/IPS) are also being used against attacks which exploit this vulnerability in servers and networks. They only offer protection against known attacks and are vulnerable to false positive or false negative detections. Anomaly based detection is also possible, but network performance is adversely affected when this approach is adopted [2,3].

As mentioned before, it is important to protect websites but gathering statistical and other information on these attacks is just as important. The above mentioned approaches do not provide any detailed information regarding the
attack or attackers. Honeypots, on the other hand, are tools usually used to attract intruders and collect cyber intelligence. Vulnerable applications are installed on Honeypots with the purpose of attracting hackers. Honeypots, however, are standalone servers dedicated for this purpose, they are usually memory hungry and require considerable resources to maintain. They are also created once and need to be updated once compromised [4].

Our proposed approach involves introducing a clone of the web site under attack and divert the attackers to this clone. By doing so we do not only protect the web site but also gather intelligence. Using the clone website, the hackers are deceived and redirected to the docker container. The attacker continues to attack under the impression that he/she is successful. Even if the attackers find a way to bypass the docker, they need to change certain files for web server applications which causes extra work for them.

In the following paragraphs we shall describe the proposed system and present the results obtained through simulating cyber attacks to a web server.

2 System Description

This study offers a method to protect the elements connected to the Internet from web-side and network-side cyber-attacks. Our first aim is to detect attacking vectors to the website. Intrusion detection system (IDS) is used to detect these attacks. Special characters and log analysis can be used for this purpose. There are many IDS products for network analysis (Snort, OSSEC, Sagan, etc.). We have chosen to use Snort which offers the benefits presented in [5]. After detecting the attack, victim website is cloned with the help of virtualization technology. Cloned website is deployed and attacker is redirected to it. Meanwhile fake data/databases are produced and shown to the attacker. Attacker thinks that he/she is successful and keeps digging. Attacker's every move is logged and monitored by administrator of the server. The aim of this clone website is to protect the main website and analyse the behaviour of the attackers.

In our project we are using Docker as a virtualization technology. Python, PHP and Javascript are chosen for scripting languages.

To summarize the above actions; when an attacker tries to hack our web application by sending malicious payloads repeatedly, our IDS system (i.e. Snort) detects the first malicious payload and it produces an alert. These alerts are parsed by our engine (written in Python3). The engine then creates a docker container cloning the actual website. After deployment, the attacker is now redirected to clone docker container. He/she thinks that he/she is successful and keeps sending packets. Docker container is isolated from the real web server so while the attacker is hacking fake-website, real web site continues to perform its operations. Every move/payload is collected by the system and stored to the database (which is hosted on Amazon Web Services (AWS) and running as Postgresql). Database is accessible to both the clone machines and our web application which is used for sharing intelligence with customers. Customers, in our case, are the subscribers who are using the described system to protect their
web sites. Database transfers the new entries to the web application and these entries are parsed and depicted in the front end.

As stated above, once the IDS system detects an attack it throws an alert and a docker container is deployed for this intruder only, and the .htaccess file is modified for redirection. After this point attacker is redirected to the cloned web page. For each attack a separate docker container is created and as they only occupy a small memory area, this approach does not create too much load for the server. Attacks are also prevented from affecting each other through this approach. As an example, if the intruder has an IP address as 123.1.2.3, then a container is activated for this particular IP and the logs of the attacks from this IP are kept in the database under this identity.

When a new attack entry is detected and this information is inserted into the database, the system sends it to our web application. Web application parses it and shares it with the customer. An email notification is also sent to the customer. The system architecture is shown in Fig. 1

![Architecture Diagram](image)

**Fig. 1.** The system architecture

### 3 Results

Testing the proposed architecture is performed using a victim machine which is hosted on AWS. All attacks are sent from our local machine. Victim machine can normally be found by automatic bots which scans all the internet.
Snort is configured to throw alerts to be processed further. If Cross-site Scripting (XSS) attack is detected (\textless script alert(1)\textgreater payload is sent), for example, Snort throws an alert, “XSS Detected”, using the pre-defined rules. This alert consists of attacker’s IP address, attacker’s port number, affected parameters, victim’s IP address, victim’s port number, severity and activity of the attacker. This information is passed on to our core engine running on our server. In this study only four attack types, cross-site scripting (XSS), form based sql injection (SQLi), error based SQLi, and Internet control message protocol (ICMP) attacks were implemented.

Alert is then parsed by the core engine. Core engine is continuously communicating with the database server where there is a table in the database for storing parsed alerts. Database stores every attack in detail.

Database communicates with our web application which is used for sharing cyber threat intelligence. All data are sent to the web application. Python/Django is using ORM (Object Relational Model) query models to query the database. If it detects a new entry, this log is parsed and shown in the user interface to the particular client who has been attacked. The displays depicting the attack severity and attack types, as presented to the clients, are shown in Fig. 2 and Fig. 3.

Following are the list of actions performed:

- Payloads are logged.
- Core engine parses log
- Logs are written in database
- Backend sends query to database
- Information is shown on the user interface in less than 1 second, thanks to the efficient infrastructure of AWS and Django.

The logs kept by the system is shown in Fig. 4 and the intelligence information provided to the clients are shown in Fig. 5.
4 Discussions

The results presented demonstrate the data obtained using the proposed system. The system works successfully both for protecting the web site and for collecting the necessary intelligence on the attacks performed. The IT managers of the respective clients would be able to access our server and display the types of attacks they are being subjected to and from where, hence they may be able to take precautions accordingly.

An alternative approach would be a different architecture where the product is running on a dedicated server. In this case, user traffic will first pass from this server and then routed to the user website. Container can now be deployed on product server as soon as an attack is detected. Through this approach the user web site will be protected from Denial of Service (DoS) attacks and its normal operation will not be affected due to bandwidth being exhausted. In this case, however, the static web pages will have to be kept at our server and shown to the attacker during the cloning process. As the application will be hosted on a service such as AWS, there will not be bandwidth or space restrictions but the cost will obviously increase. The alternative architecture is shown in Fig. 6

One other problem with the current approach arises when the intruder performs an attack to an online shopping site, for example, and starts shopping, and starts performing the attack after putting a few items into the cart. In this case, because the clone would be showing the static pages, he/she would not be able to see the cart hence will be able to detect that he/she has been compro-
mised. Experimental work should also be carried out to measure the overheads introduced by this system, and the data to be gathered should also be decided, may be case by case. We intend to study these issues as well as the application to new attack types as future work.

References