Sri Lanka Institute of Information Technology

Assignment 01

Malware Detection Using Machine Learning

It20609580 W.A.C.S Weerasinghe

IE3042 – Secure Software Systems

B.Sc. Honors Degree in Information Technology

Specialized in Cyber Security

Contents

Acknowledgement

I would like to express my heartfelt gratitude to Dr. Lakmal Rupasinghe, for their invaluable support and encouragement throughout this research. Their contributions have been instrumental in the successful completion of this work.

Purpose

The purpose of the research article on "Malware Detection Using Machine Learning" is to investigate and explore the application of machine learning techniques for detecting and combating malware threats. The primary objective is to contribute to the field of cybersecurity by developing effective methods and algorithms that can accurately identify and classify malicious software. By leveraging machine learning, the research aims to enhance the detection capabilities, improve the efficiency of malware detection systems, and ultimately contribute to the development of more robust and proactive cybersecurity measures. The purpose is to aid in the protection of computer systems and networks from potential cyber threats, thereby safeguarding sensitive information, ensuring data privacy, and mitigating the risks associated with malware attacks**.**

Introduction

The proliferation of malware poses a significant threat to the security and integrity of computer systems and networks. Malicious software, commonly known as malware, encompasses various forms such as viruses, worms, Trojans, and ransomware, designed to exploit vulnerabilities and compromise the confidentiality, availability, and integrity of data. Traditional signature-based detection methods have proven to be insufficient in keeping up with the rapidly evolving malware landscape, necessitating the exploration of alternative approaches.

Machine learning has emerged as a promising technique for malware detection due to its ability to analyze large volumes of data and identify patterns that distinguish between benign and malicious software. By leveraging machine learning algorithms, it becomes possible to develop robust and adaptive models capable of effectively identifying and mitigating the risks associated with malware.

The objective of this research paper is to investigate the application of machine learning in the context of malware detection. By employing various machine learning algorithms and techniques, we aim to develop accurate and efficient models that can detect and classify malware with a high level of precision and recall. The focus is on exploring the potential of machine learning to enhance the effectiveness of malware detection systems, improve detection rates, and reduce false positives and false negatives.

In this paper, we will first provide an overview of the current landscape of malware threats and the limitations of traditional detection approaches. We will then delve into the fundamentals of machine learning and its suitability for addressing the challenges posed by malware. Various machine learning algorithms, such as decision trees, support vector machines, and neural networks, will be explored in the context of malware detection, highlighting their strengths and weaknesses.

Furthermore, we will discuss the selection and preprocessing of features, including static and dynamic analysis techniques, to effectively represent malware samples in the machine learning models. The importance of feature engineering and dimensionality reduction techniques will also be emphasized.

To evaluate the performance of the proposed machine learning models, we will employ publicly available datasets and benchmark them against state-of-the-art malware detection systems. The evaluation will consider metrics such as accuracy, precision, recall, and F1-score to assess the efficacy and robustness of the models.

The findings of this research paper aim to contribute to the field of cybersecurity by providing insights into the capabilities and limitations of machine learning techniques for malware detection. By understanding the potential of machine learning in combating malware threats, we can pave the way for more effective and proactive strategies to protect computer systems and networks from the ever-evolving landscape of malware.

Netsparker

The overall risk rating of this website is at Medium.

Vulnerability Summary

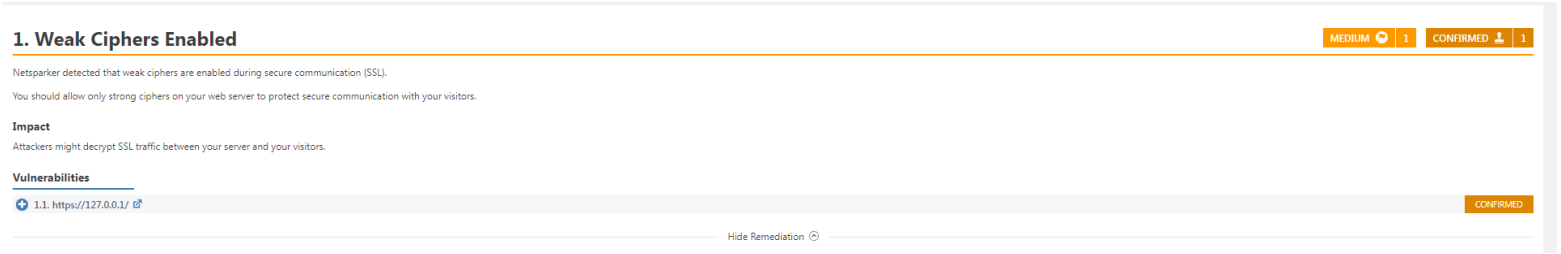
Vulnerabilities By OWASP 2017

SEVERITY FILTER : ☑ CRITICAL ☑ HIGH ☑ MEDIUM ☑ LOW ☑ BEST PRACTICE ☑ INFORMATION

CONFIRM	VULNERABILITY	METHOD	URL	SEVERITY
A3 - SENSITIVE DATA EXPOSURE				
	Weak Ciphers Enabled	GET	https://127.0.0.1/	MEDIUM
	Insecure Transportation Security Protocol Supported (TLS 1.0)	GET	https://127.0.0.1/	LOW
	Insecure Transportation Security Protocol Supported (TLS 1.1)	GET	https://127.0.0.1/	BEST PRACTICE
	Referrer-Policy Not Implemented	GET	http://127.0.0.1:5000/	BEST PRACTICE
A5 - BROKEN ACCESS CONTROL				
	[Possible] Cross-site Request Forgery	GET	http://127.0.0.1:5000/	LOW
A6 - SECURITY MISCONFIGURATION				
	Missing X-Frame-Options Header	GET	http://127.0.0.1:5000/	LOW
	OPTIONS Method Enabled	OPTIONS	http://127.0.0.1:5000/	INFORMATION
A9 - USING COMPONENTS WITH KNOWN VULNERABILITIES				
	Out-of-date Version (Bootstrap)	GET	http://127.0.0.1:5000/static/js/bootstrap.min.js	MEDIUM
	Out-of-date Version (jQuery)	GET	http://127.0.0.1:5000/static/js/jquery.min.js	MEDIUM

0
0
3
3
2
1

CONFIRM medium Vulnerability.



Impact

Attackers might decrypt SSL traffic between your server and your visitors.

Actions to Take

1. For Apache, you should modify the SSLCipherSuite directive in the `httpd.conf`.

```
SSLCipherSuite HIGH:MEDIUM:!MD5:!RC4
```

2. Lighttpd:

```
3. ssl.honor-cipher-order = "enable"
```

```
ssl.cipher-list = "EECDH+AESGCM:EDH+AESGCM"
```

For Microsoft IIS, you should make some changes to the system registry. **Incorrectly editing the registry may severely damage your system. Before making changes to the registry, you should back up any valued data on your computer.**

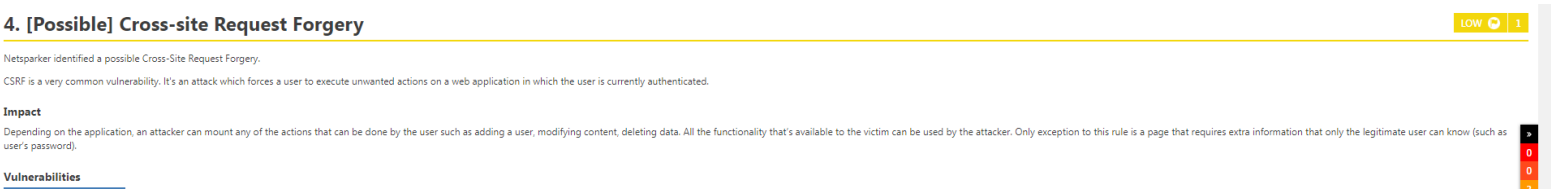
a.Click Start, click Run, type `regedt32`or type `regedit`, and then click OK.
b.In Registry Editor, locate the following registry key: `HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders`
c.Set "Enabled" DWORD to "0x0" for the following registry keys:

```
SCHANNEL\Ciphers\DES 56/56
SCHANNEL\Ciphers\RC4 64/128
SCHANNEL\Ciphers\RC4 40/128
SCHANNEL\Ciphers\RC2 56/128
SCHANNEL\Ciphers\RC2 40/128
SCHANNEL\Ciphers\NULL
SCHANNEL\Hashes\MD5
```

Remedy

Configure your web server to disallow using weak ciphers.

Conform low vulnerability



Impact

Depending on the application, an attacker can mount any of the actions that can be done by the user such as adding a user, modifying content, deleting data. All the functionality that's available to the victim can be used by the attacker. Only exception to this rule is a page that requires extra information that only the legitimate user can know (such as user's password).

Vulnerabilities

4.1. http://127.0.0.1:5000/

Hide Remediation

Remedy

- Send additional information in each HTTP request that can be used to determine whether the request came from an authorized source. This "validation token" should be hard to guess for attacker who does not already have access to the user's account. If a request is missing a validation token or the token does not match the expected value, the server should reject the request.
- If you are posting form in ajax request, custom HTTP headers can be used to prevent CSRF because the browser prevents sites from sending custom HTTP headers to another site but allows sites to send custom HTTP headers to themselves using XMLHttpRequest.
    - For native XMLHttpRequest (XHR) object in JavaScript;

```
xhr = new XMLHttpRequest();

xhr.setRequestHeader('custom-header', 'valueNULL');
```

For JQuery, if you want to add a custom header (or set of headers) to

a. **individual request**

```
$.ajax({
    url: 'foo/bar',
    headers: { 'x-my-custom-header': 'some value' }
});
```

b. **every request**

```
$.ajaxSetup({
    headers: { 'x-my-custom-header': 'some value' }
});
OR
$.ajaxSetup({
    beforeSend: function(xhr) {
        xhr.setRequestHeader('x-my-custom-header', 'some value');
    }
});
```

External References

- OWASP Cross-Site Request Forgery (CSRF)

Remedy References

- OWASP Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet