

**R & D Project Report**

**Academic Year- 2021-22**

On

**EXPLOIT DEVELOPMENT**

Submitted by

1. Shalom Shreyan-BT19GCS004 CSE
2. Rosy Chadha-BT19GCS068 CSE
3. Agam Goel-BT19GCS157 CSE

Vivek Kumar Anand



=================================================================

**CERTIFICATE BY SUPERVISOR(S)**

This is to certify that the present R&D project entitled ……………………………………………….. being submitted to NIIT University, Neemrana, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology, in the area of BT/CSE/ECE/GIS, embodies faithful record of original research carried out by ----------------------------------. She / He / They has / have worked under my/our guidance and supervision and that this work has not been submitted, in part or full, for any other degree or diploma of NIIT or any other University.

Place: Vivek Kumar Anand

Date:



=================================================================

**DECLARATION BY STUDENT(S)**

I/We hereby declare that the project report entitled …………………………………. which is being submitted for the partial fulfilment of the Degree of Bachelor of Technology, at NIIT University, Neemrana, is an authentic record of my/our original work under the guidance of --------------. Due acknowledgements have been given in the project report to all other related work used. This has previously not formed the basis for the award of any degree, diploma, associate/fellowship or any other similar title or recognition in NIIT University or elsewhere.

Place:

Date:

1. Shalom Shreyan-BT19GCS004 CSE

2. Rosy Chadha-BT19GCS068 CSE

3. Agam Goel-BT19GCS157 CSE

**Introduction**

An exploit is a program or code that is structured to take advantage of a security fault or vulnerability in a computer system or different applications. Cybercriminals can use exploit-development for malicious purposes like the installation of malware. In other words, the target of an assault has a design fault that permits others to devise ways to gain access to it and utilize it in his favour.

Exploits can either work remotely or locally. An exploit that works remotely works over a network and exploits the security vulnerability without having any prior access to the vulnerable system whereas an exploit that works locally requires prior access to the vulnerable system, usually done by social engineering. Many exploits are designed to get “root” aka super user-level access to a computer system but it is possible to use multiple exploits to first achieve low-level access and then escalate privileges repeatedly until root access aka the highest administrative level is achieved.

Types of exploits: -

1. Known Exploits

When the authors of vulnerable software get notice of an exploit, the vulnerability is patched immediately, rendering the exploit ineffective. Security vendors were, therefore, given access to this information. Some organizations identify publicly known cybersecurity vulnerabilities and provide an identification number, a description, and a public reference.

1. Unknown or Zero-day Exploits

Zero-day exploits are unknown to everyone except the folks who created them. These are by far the most hazardous exploits, as they occur when a piece of software or system design contains a severe security flaw that the vendor is unaware of. The vulnerability is discovered when a hacker attempts to exploit it, hence the name "zero-day exploit." Once a cyber-attack is launched, systems running the exploit software are susceptible. Either a vendor will issue a patch to address the vulnerability, or security software will identify and stop the exploit and malware that results.

The CVE (Common Vulnerabilities and Exposures) Program exists to identify, define, and catalogue all the publicly disclosed cybersecurity vulnerabilities. There exists one CVE Record for each vulnerability present in the catalogue. The vulnerabilities are discovered, then assigned a unique ID and then published. Each vulnerability has a unique number, for example, CVE-2022-29505 where the "2022" refers to the year the vulnerability was discovered and the "29505" is the unique ID for this specific vulnerability.

**Common Vulnerabilities**

Our work started off by researching about common vulnerabilities so as to develop exploits for them. Following are the vulnerabilities that are required to get the clear understanding about the concepts.

1. Remote Code Execution: Attack based on this vulnerability allows an attacker to gain access to a victim's machine and execute system commands, write, modify, delete or read files, and can connect to databases, usually irrespective of where the machine and the attacker are physically located. The victim’s machine gets compromised as it provides the attacker with the ability to execute malicious code and take complete control of the infected system along with the privileges of the victim.
2. Denial-of-Service: Attack based on this vulnerability can cause a service or a website to become unavailable, temporarily or indefinitely, by overloading it with massive loads of traffic generated from single or multiple sources which can lead to shutting down or slowing down of entire systems on the network. There are two types of DoS attack : - flood attacks and crash attacks. In flood attacks, the system slows down and eventually stops because of the continued requests and can be caused by ICMP flood attack or SYN flood attack. With crash attacks, entire systems crash as the hackers transmit bugs that exploit flaws in the targeted system.
3. Spoofing: Attacks based on this vulnerability allows the hacker to impersonate someone in an attempt to gain confidence, get access to systems, steal data, steal money, or spread malware. It can be done via emails, websites, caller id, text message, MITM attack or changing IP.
4. Privilege Elevation: Attack based on this vulnerability allows a hacker to get illicit access to elevated rights i.e. beyond what was authorized to the user. It is of two types: Horizontal privilege escalation and Vertical privilege escalation. Horizontal privilege escalation involves gaining access to the rights of another account which has similar privileges, thus allowing the attacker to broaden their sphere. Vertical privilege escalation involves elevating privileges beyond what the user already has. It can be done by exploitation of credentials, presence of vulnerabilities and exploits, system misconfigurations, existence of malware in the system and social engineering.
5. Buffer Overflow: Attacks based on this vulnerability can cause a program to behave unpredictably and generate incorrect results, memory access errors, or crashes as the volume of data inputted exceeds the storage capacity of the memory buffer. An attacker can overwrite an existing pointer and point it to an exploit payload, which would be useful to gain control over the program. It is of two types: Stack-based & Heap-based. Stack-based buffer overflows use stack memory that only exists during execution of a function. Heap-based attacks flood the memory space allocated for a program being used for current runtime operations.

By getting a brief view about different types of vulnerabilities, our research becomes more centric to web vulnerabilities and towards their exploitation.

**Web Vulnerabilities**

So for working on and knowing about the web vulnerabilities we started our research on the OWASP top 10 web application security risks to get an idea about what web vulnerabilities or these risks are and how we can develop an application or exploit based on it.

Following is the brief description about the OWASP 10 vulnerabilities: -

1. Broken Access Control: It means when someone acts outside of the permission they are given or accessing information which is not allowed to be seen by them. With this someone can either get the data or edit or delete it or use it without any permission. The attacker can perform it by various ways like changing the URL or the html page, by gaining access to some other account who has high privileges, metadata manipulation etc.
2. Cryptographic Failures: Cryptography is necessary to cipher the data in transit or the data traveling or at rest. Data such as credit card or debit card numbers, passwords, etc., require extra security. If cryptography fails an attacker can do a MITM attack and can retrieve such sensitive information. This can happen due to many reasons such as using old cryptographic methods which are easy to break nowadays, the server certificates and trust chains are not properly validated etc.
3. Injection: This mainly includes cross-site scripting and SQL injection. SQL injection can be used by an attacker on the websites using SQL database where an attacker can retrieve information by typing several SQL commands parts or by running a payload which runs various SQL command parts and trick the interpreter to execute these commands and give access to sensitive and unauthorized information. On the other hand for XSS vulnerability, an attacker attacks the script embedded in the page on the client side. He/she injects malicious scripts on the victim’s browser with the help of which an attacker can deface websites, steal session cookies, run malware on target’s machine etc.
4. Insecure Design: This means that websites or applications are at risk due to the way they are designed. This does not mean that the app is designed poorly, a secure design can have some implementation defects that can lead to several vulnerabilities thus making the application insecure. For example, if a movie theatre is providing discounts if someone book seats in groups, for every extra one seat booked it provides a 10% discount on the total amount but the maximum number of seats can be booked are 10 before paying. The attacker can exploit the model and see if he/she can book the entire cinema in one go due to which the movie theatre will face a great loss of income.
5. Security Misconfiguration: These configurations are set for the application, database server, web server etc. and are deployed. If not configured properly the attacker can access unauthorized sensitive information such as app server version, database server version etc., and can perform other attacks on the basis of this information.
6. Vulnerable and Outdated Components: This means that applications are using older versions of components on either client side or server side which are vulnerable and have been attacked before. We can prevent it by regularly checking and upgrading the versions and by regularly scanning for vulnerabilities if the components are up-to-date.
7. Identification and Authentication Failures: When someone login on a website a session is created and the user is authorized with the help of cookies in that session. For each new session, a new cookie is created for identification and authentication. So the cookie used in the previous session should be invalid. An attacker can steal these cookies and get their hands on some sensitive data as these cookies contain usernames and passwords. It makes a possibility for brute force attack if the password is hashed or the attacker can hijack the session if the session id is the same before and after login and logout using XSS. Or if the user is using a public system he/she just closed the browser abruptly instead of logging out, if the session timeouts are not implemented properly the attacker can use the same public system and just can open the browser and he/she will be already logged in as the session was not ended.
8. Software and Data Integrity Failures: The vulnerabilities which arise due to the code which does not verify integrity properly or in other words does not protect against integrity violations. For example, an application using libraries or modules from untrusted sources or repositories. Another example can be, today a lot of applications use the feature of auto-update which downloads the application’s newer version without properly verifying the integrity. The attacker can upload his/her own updates and distribute them thus making the system vulnerable.

After reading and understanding about some of the OWASP top 10 security risks we chose to work on the injection attacks and vulnerabilities, i.e., SQL injection and XSS injection. Then we gradually moved towards incorporating other types of vulnerabilities in our application.

**Problem Statement**

Many vulnerabilities necessitate an attacker initiating a sequence of suspicious operations to set up an exploit. The majority of vulnerabilities are caused by software or system design flaws. Attackers build code to exploit these flaws to infect the machine with various sorts of malware. It is often built as a large collection of exploits and can be easily hidden in plain view on various websites. The hidden code present on a website can quickly fingerprint the user’s computer and identify the type of operating system they are using, which programs are running and check for flaws and weaknesses, if they exist. If it is found that the system has any vulnerabilities, they can be exploited to infect the computer or network with any number of malware, ranging from spyware to ransomware.

In today’s world internet is the major cause of spreading many malware, viruses and embedded scripts. Hence our problem statement will revolve around the detection of malicious and benign URLs and thus testing them to get the type of vulnerability.

**Literature Review**

“Software Crash Analysis for Automatic Exploit Generation on Binary Programs” paper wrote about a method to generate attacks automatically on binary programs using software crashes by a new symbolic input method in combination with lazy evaluation on pseudo symbolic variables to handle symbolic pointers which improved speed. Exploits are created from inputs given by the crash and is an automated process for software failures without knowing the source code while being highly scalable. Concolic testing determines if a certain path is exploitable as focus only on one path due to input constraints. Bitblaze analyzes crashes to find out which ones are exploitable. It can be triggered by a tainted instruction pointer and diverts vulnerable paths by concolic execution and exploit constraints (return oriented programming, payload and feed) to manipulate instruction pointers. Most threats are continuous, like goto statements but are turned into non-local jumps, exception handling, and user threads. EIP register is monitored as it contains control flow statements and corrupted pointers can be stored in them. Contiguous memory is needed to hold the payload. Format string exploit is caused by misuse of printf and syslog function as an address can be overwritten with a random value but CRAX supported this attack only for Linux.  CRAX is a feasible and powerful exploit generation tool which is usable for real environments and can be extended to larger programs like Microsoft Office and popular web browsers. As CRAX can be applied to the whole system, it can be used to attack web systems apache with the php module and mysql server.

“A Network Based Vulnerability Scanner for Detecting SQLI Attacks in Web Applications” paper talked about detecting SQL injection attacks on web applications with a low false positive rate and consuming less time. SQL injection attacks have shown themselves to be the most threatening attack as the attacker gains the ability to execute any SQL commands on the server database. URLs need to be properly sanitized to prevent this type of attack. Three tools available on the market were tested, which were Acunetix, Netsparker and WebCruiser. All of them had a high false positive rate and took a lot of time due to high overhead. The proposed new system is divided into three parts, Crawler, Attack Simulation and Network Setup. This system was able to provide maximum coverage while providing zero false positives and taking the least amount of time. The efficiency is heavily dependent upon the number of systems which are connected within the network.

“Diagnosing Software System Exploits” paper mentioned how to get different exploits other than those which they tested before to make the system diagnose more efficient and fast.They used fuzzer for this operation as the did not take some inputs till an exploit is found but they manipulated the given exploit until found a new one that can bypass their security. But there was a problem, fuzzer was giving similar outputs thus, reducing the efficiency so they used a technique called under-tracing in which they trace the exploits with low granularity level to get better results at higher levels. As we will be using two vulnerabilities from OWASP i.e. SQL injection and Cross Site Scripting (XSS), this fuzzer will not be of much use for us.

“Integrated Exploit Kit for Web Application” by Aryya Dwisatya Widigdha and Yudistira Dwi Wardhana Asnar, 2019. During this time, data and information became the most critical assets in the organization to safeguard. Exposed vulnerabilities put assets like data and information at risk, so they must be corrected as soon as feasible. Vulnerability identification requires time and occasionally results in a large number of failures. By combining vulnerability identification tools as an exploit kit, this study offered a technique to reduce false positives in vulnerability identification findings and speed up the process. The solution may reduce vulnerability detection time by 50% for two targets and boost vulnerability identification certainty by using manual analysis and proof of concept features. Lastly, the paper talked about integrating it with machine learning for processing vulnerability identification to fully remove the need for human intervention which will lead to reduction of analysis time and false positive rates.

“Detecting Cross-Site Scripting Attacks Using Machine Learning” by Fawaz A. Mereani and Jacob M. Howe, 2018. Cross site scripting is one of the most widely recognized sorts of internet based application attacks, subsequently it is significant in data security. XSS happens when an aggressor infuses noxious code, typically JavaScript, into a web application to be executed in the client's program. Recognizing malignant contents is a significant part of an internet based application's protection. This review concentrates on the utilization of SVM, k-NN, and Random Forests to identify and restrict known and unseen attacks on JavaScript code by creating classifiers.

It shown that utilizing a charming list of capabilities that joins language linguistic structure and conduct data produces in classifiers with great exactness and accuracy on immense true informational indexes without zeroing in exclusively on confusion.

**Proposed methodology**

The workflow of our project is divided into two broad categories. In first category detection of malicious and benign websites takes place. For the second category, web scraping is done using python libraries so as to get the content of the site and then we’ll set up a connection with the site to send our payload and check whether the site is exploitable or not.

Following is the list of the machine learning algorithms that are used: -

1. Logistic Regression
2. SVM
3. Random Forest
4. K-Nearest Neighbors

* Logistic Regression

Logistic Regression comes under supervised learning in machine learning. It is used to forecast the overall subordinate variable using a given set of autonomous components. It majorly predicts the categorical dependent variable’s output. Hence the result would be in the form of a categorical value or discrete value. This can be given as 0 or 1, Yes or No but it never gives the exact value as 0 and 1 instead the result will be a probabilistic value which lie between 0 and 1. This algorithm is an important machine learning approach because it can categorize data as discrete or continuous and generate probabilities.

Types of logistic regression:

1. Binomial

The reliant factors in binomial logistic regression must be one of two sorts: 0 or 1.

1. Multinomial

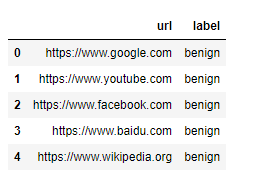
In multinomial logistic regression, the reliant variable could incorporate at least three unordered sorts, for example, "trees," "herbs," or "plants".

1. Ordinal

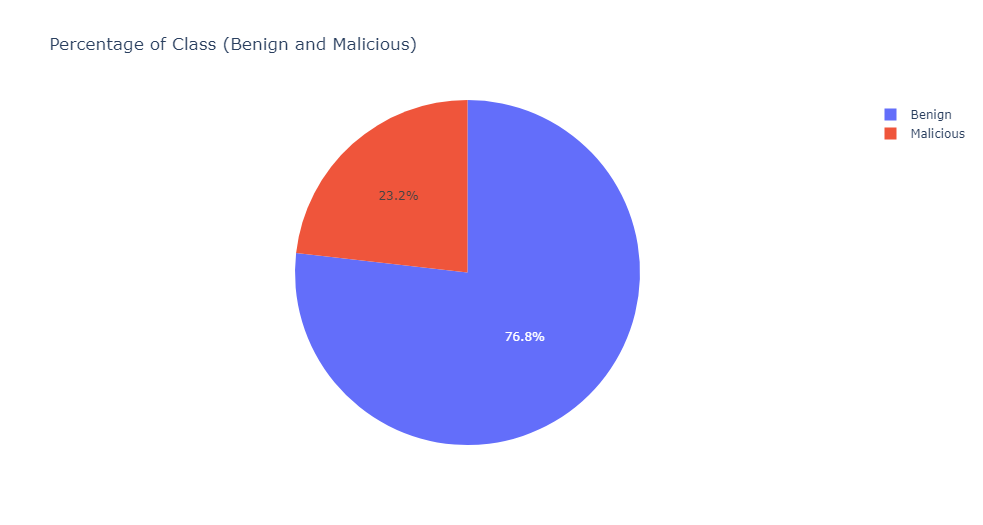
Ordinal Logistic regression can have at least three arranged kinds of ward factors, for example, "low," "medium," or "high."

**Malicious URL Detection**

The dataset used would be of different types of URLs marked as “malicious” and “benign”. Malicious URL are those which are harmful and have no security whereas benign URLs are secure and are well intentioned.

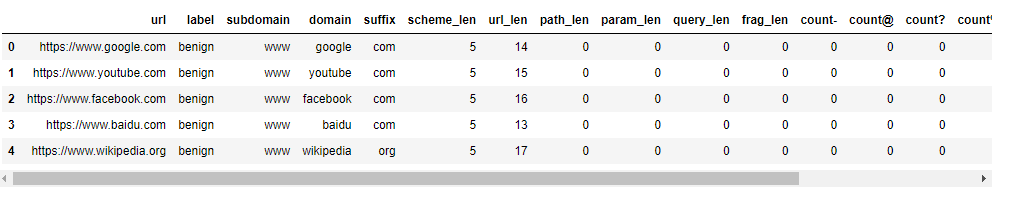


We can visualize the data by using python libraries.

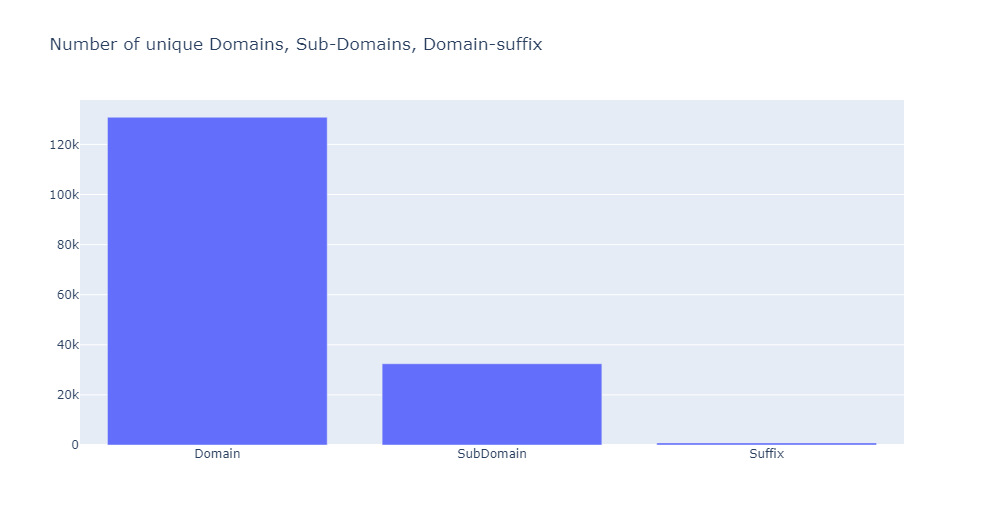


Now the data will be processed and it can will be divided into following features would be extracted: -

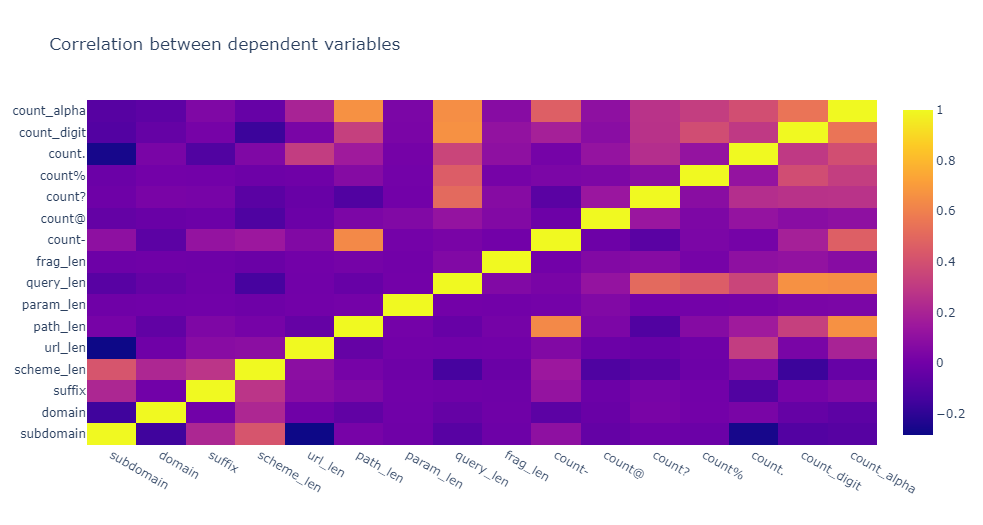
* Sub Domain
* Domain
* Suffix
* Scheme length
* Path length
* Parameter length
* Query length
* Fragment length
* Count of '-'
* Count of '&'
* Count of '?'
* Count of '%'
* Count of '.'
* Count of digits
* Count of alphabets



The figure below depicts the number of domains, sub-domains, and domain-suffix that are generated after extraction.

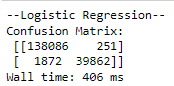


Heat map is generated for showing the correlation between dependent variables.

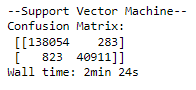


For model testing confusion matrix is generated for every algorithm:-

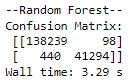
1. Logistic Regression



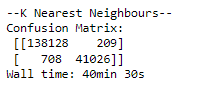
1. SVM



1. Random Forest



1. K-Nearest Neighbors

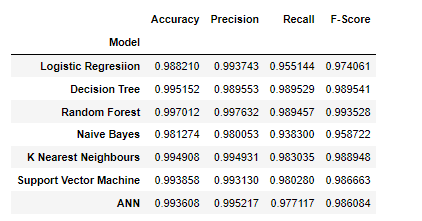


Following figure shows the comparison between the algorithms



**Result**

Precision denotes how many websites the model classified as malicious are actually malicious. The most import metric in this scenario is Recall, because Recall is calculated as out of all the websites that are malicious, how many were predicted as malicious and our main task is to identify malicious websites.



According to the above table random forest is the best algorithm to find out the result and it took less time as compared to others.

**Future Scope**

We are more focused towards cross-site scripting and SQL injection which are part of injection as per OWASP 10 vulnerabilities. In future, we will try to incorporate all other types of web vulnerabilities.

**References**

1. <https://owasp.org/www-project-top-ten/>
2. <https://www.exploit-db.com/docs/50577>
3. <https://ieeexplore.ieee.org/document/8984449>
4. <https://ieeexplore.ieee.org/document/9430543>
5. <https://ieeexplore.ieee.org/document/8970609>
6. <https://ieeexplore.ieee.org/document/6717039>
7. <https://link.springer.com/chapter/10.1007/978-3-319-74690-6_20>
8. <https://www.guru99.com/web-security-vulnerabilities.html>
9. <https://www.beyondtrust.com/blog/entry/privilege-escalation-attack-defense-explained>
10. <https://www.tutorialspoint.com/python_penetration_testing/python_penetration_testing_dos_and_ddos_attack.htm>
11. <https://www.malwarebytes.com/spoofing>
12. <https://cyware.com/news/remote-code-execution-vulnerability-what-is-it-and-how-to-stay-protected-from-it-12a1b250>
13. <https://securiumsolutions.org/the-most-common-types-of-vulnerability-in-cybersecurity/>

Source code/hardware description/equipment description (As Annexure)