VOLUME-7, ISSUE-4, APRIL-2018 • PRINT ISSN No 2277 - 8160

Original Research Paper

Computer Science

AUTOMATIC COLLECT AND ORGANIZED VARIOUS APPLICATIONS LOG EVENT DATASET

P. Saravanan — Asst.Professor, Department of computer science, Govt. Arts College, Dharmapuri, Tamilnadu

Dr. V. Sangeetha — M.Sc., Mphil., P.hD, Asst.Professor, Department of computer science,PRUCAS, Dharmapuri, Tamilnadu

ABSTRACT

Centralizing event logs in a single location enables faster log analysis. Event Log Manager helps centralize logs from various applications, servers, network devices spread across the organization[8]. Normalization better analyze Event Log detail, without getting interrupt with unreadable machine data. Syslog, event logs, and flat files normalized to provide a detailed account of the specific event name, insertion detection time, source machine IP and destination account. A number of pre-defined filters organized by categories during a firefight[1]t. The filters provide real-time visibility event activities. The drill down to the specific event get detailed listing of the source destination, ports, create new filters, conditions enable in-console notifications investigate a specific network, server, application suspicious user activity.

KEYWORDS : Detect, track, XSS client/server and analyze malicious events network.

1. Introduction:

Correlate all events from the network devices, applications, servers, storage, security appliances, and other systems real-time with Log & Event Manager[5]. This Manager includes hundreds of built-in event correlation rules. Time transaction-based event allows to simultaneously perform multiple actions. An ordering permutation issues are covered with non-linear event correlation tools notified of malicious events inside network, which helps remediate threats in less time[10]. Often, a single log viewed on its own may seem completely normal, but when viewed with a group of other related logs, could form a potential attack pattern[2]. EventLog Analyzer's correlation engine discovers sequences of logs coming from devices across the network that indicate possible attacks, and quickly alerts the threat. It empowers to go beyond just analyzing logs and start taking proactive steps against attacks.

2. Log correlation with Event Log Analyzer

Event Log Analyzer's powerful correlation engine efficiently identifies defined attack patterns the logs.

1. Predefined rules
2. Overview dashboard
3. Timeline view
4. Intuitive rule builder
5. Field-based filters
6. Instant alerts
7. Rule management
8. Column selector
9. Scheduled reports

Utilize over 20 predefined attack patterns rules that come packaged with product. Navigate easy-to-use correlation dashboard, provides detailed reports for each attack pattern as well as an overview report of all discovered attacks, facilitating in-depth analysis[3]. View a timeline diagram showing the chronological sequence of logs for every identified attack pattern. New attack procedure rule builder, provides a categorical list of network actions allows to drag and drop desired order Set constraints log fields for fine-grained control over the defined attack patterns[7]. Set up email or SMS notifications immediately alerted whenever the system picks up a suspicious pattern. Enable, disable, remove, or edit rules and their notifications from the single page[6]. Control the information shown in each report by selecting the columns needed and renaming required.

2.1 Create rules with an intuitive interface

1. Define new attack patterns using over one hundred network actions.
2. Drag and drop rules to rearrange which actions comprise a pattern.
3. Restrict certain log field values with filters.
4. Specify threshold values for triggering alerts.
5. Add a name, category, and description for each rule.
6. Edit existing rules to fine-tune alerts.
7. A specific rule is generating too many false positives or fails to identify an attack.

3. Features

In the memory, cross platform event processing instant notification and remediation without waiting on data queries[11]. Mollify issues in real-time with active responses that will block IPs, change privileges, disable accounts, and kill applications. ability to prove the limited impact of the security from fines, penalties, and even legal action. The security data investigate incidents for remediation or audit reporting purposes[4]. File modifications, deletions, and permission Monitor and alert on registry, file, and folder activity to detect suspicious and malicious behavior.

3.1. spear phishing

The success of a spear phishing attack is dependent an end user clicking the link embedded in a crafty email unlimited nature of human creativity together with Social Engineering have strengthened the odds that at least one target in a phishing attack will click on that security compromising link[12].

3.2. Protect sensitive data

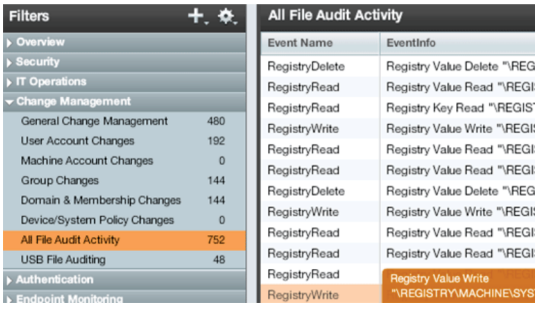

Fig:-1 File monitoring

Use File Integrity Monitoring (FIM) to detect and alert changes to the files, folders, and registry settings. this methods need to monitor file changes to stay compliant with PCI, SOX, or HIPAA standards. The methods simply ensure the security methods IT environment, Log

and Event Manager help the methods protect confidential files.

4. Secure privileged accounts

Most external breaches involve an attempt to compromise credentials. Important to establish controls to proactively monitor changes and usage patterns associated with privileged accounts[9]. Log and Event Manager will alert and respond in real-time suspicious activity related to permission changes or abuse. Eliminate threats faster with instantaneous detection of suspicious activity and automated responses[12]. Mitigate security threats Conduct security event investigations and forensics for mitigation and compliance.

4.1. Web application: XSS

Many Web applications contain vulnerabilities that allow attackers to use Cross Site Scripting (XSS) to misrepresent a website. As a result attackers are often able to get victims who interact with these illegitimate web pages to online click on a capture login credentials[5]. Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur uses a web application to send malicious code, generally in the form of a browser side script, the different end user[7]. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it. An attacker use XSS to send a malicious script to an unsuspecting use[16]r. The end user's browser has no way to know that the script should not be trusted, and will execute the script. The script came from a trusted source, the malicious script access any cookies, session tokens, or other sensitive information retained by the browser For more details on the different types of XSS flaws.

1. Server XSS
2. Client XSS

Server XSS occurs when untreated user supplied data is included in an web response generated by the server. The source of this data could be from request, and stored location[12]. such have both Reflected Server XSS the entire vulnerability server-side code and the browser is simply rendering the response and executing any valid script embedded.

The client XSS data is used to update the document object model(DOM) with an unsafe Script call. Introduce valid Script into the DOM source of this data it could have been sent by the server via an AJAX call page load[14]. The ultimate source of the data could have been from a request, to stored location on the client or the server. Both Reflected Client XSS With these new definitions[11]. The definition of DOM Based XSS doesn't change DOM Based XSS is simply a subset of Client XSS.

SSX	Server	Client
Stored	Stored server SSX	Stored Client SSX
Reflected	Reflected server SSX	Reflected Client SSX

Fig:-2 Untrusted data used

4.2. Recommended Server XSS Defenses

Server XSS is untrusted data in the web response easiest and strongest defense against Server XSS in most cases. Context-sensitive server side output encoding the details of how to implement Context-sensitive server side output encoding presented in the OWASP XSS (Cross Site Scripting) Prevention Cheat Sheet in great detail[8]. Input validation data sanitization also be performed to help prevent Server XSS much more difficult to get correct than context-sensitive output encoding.

4.3. Recommended Client XSS Defenses

Client XSS is data used to update the DOM with an unsafe Script call the easiest and strongest defense against Client XSS. Developers frequently methods in their favorite JavaScript library are safe some information jQuery methods are safe and unsafe is presented Based XSS presented at OWASP[7]. A Script method is unsafe, primary recommendation is to find an alternative safe method to use context sensitive output encoding done in the browser, before passing that data unsafe JavaScript method. OWASP's guidance how do this properly is presented in the DOM based XSS Prevention Cheat Sheet. JavaScript and jQuery methods safe and unsafe is presented DOM Based XSS talk presented at OWASP.

A primary recommendation is to find an alternative safe method to use[9]. The context sensitive output encoding done in the browser, before passing that data to the unsafe JavaScript method. OWASP's guidance on how do this properly is presented in the DOM based XSS Prevention Cheat Sheet.

5. A positive XSS prevention model

A web page like a template, with slots where a developer is allowed to get untrusted data. These slots cover the vast majority of the common places where a developer might want to put untrusted data. Putting untrusted data in other places in the web is not allowed[15]. The way browsers parse web, each of the different types of slots has slightly different security rules and methods get untrusted data into these slots, need to take certain steps to make the data does not break out of that slot into a context that allows code execution[14]. the web document like a parameterized database query data kept in specific places and is isolated from code contexts with escaping. Sets out the most common types of slots and the rules for putting untrusted data into them safely. Based on the various specifications, known XSS vectors, and a great deal of manual testing with all the popular browsers to determined that the rules.

5.1. XSS Prevention Rules

The following rules are intended to prevent all XSS in the methods application. While these rules do not allow absolute freedom in putting untrusted data into an web document, they should cover the vast majority of common use cases[7]. Many organizations may find that allowing only Rule-1 and Rule-2 are sufficient for their needs The interrupt rules here have been carefully designed to provide protection even against future vulnerabilities introduced by browser changes.

Rule-1 Insert untrusted Data Exception Locations
<script>…never put untrusted data here..</script> directly in a script

<!–… never put untrusted data here…> inside a html comment

<…div never put untrusted data here…=test/> in an attribute name

< never put untrusted data here…href=”/test”/> in a tag name
<style>… never put untrusted data here…</style>

Attribute Escape before insert untrusted data

Rule-2 is for putting untrusted data into typical attribute values. This should not be used for complex attributes of the event handlers extremely important that event handler attributes. Rule-4 Untrusted Data in to Script Data values concerns dynamically generated JavaScript code - both script blocks and event-handler attributes[16]. The only safe place to put untrusted data into this code is inside data value. Including untrusted data inside any other data context is quite dangerous and extremely easy to switch into an execution context with characters.

<script>alter'...escape untrusted data before putting here…'</script>

<script>x='... escape untrusted data before putting here…'</script>one side of a quoted expression

<script>window.setinterval('…even escapeuntrusted data are XSSED here…');

Web escape JSON values in an web context and read the data with JSON. parse In a Web 2.0 world, the need for having data dynamically generated by an application in a javascript context is common[9]. One strategy is to make an AJAX call to get the values often an initial block of JSON is loaded into the page to act as a single place to store multiple values. This data is tricky, though not impossible, to escape correctly without breaking the format and content of the values Ensure returned Content-Type header is application. This shall instruct the browser not misunderstand the context and execute injected script.

Bad HTTP response:
HTTP/1.1 200
Date: wed, 07 Feb 2013 10:27:55 GMT
Server: Microsoft-IIS/8.5…
Content-Type: text/html; charset=utf-9 <-- bad…
Content-Length: 375
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive

{"Message":"no HTTP resource was found that matches the request URI'dev.net.ie/api/pay/.html?HouseNumber=9&AddressLine=The+Gardens<script>alter(1)</script>&AddressLine2=foxload+woods&TownName=Meath'","MessageDetail":"No type was found that matches the controller named 'pay'."}

Good HTTP response
HTTP/1.1 200
Date: Wed,06 Feb 2013 10:28:54 GMT
Server:Microsoft-IIS/7.5

Content-Type: application/json; charset=utf-8 <-- goodJSON entity encoding the rules found in the Output Rules this will not allow the methods to use XSS protection provided by CSP entity encoding[11]. This technique has the advantage that html entity escaping is widely supported and helps separate data from server side code without crossing any context boundaries. Consider placing the JSON block on the page as a normal element and then parsing the inner web to get the contents that reads the span live in an external file making the implementation of CSP enforcement.

Js file dataElement = document.getElementByid('int-data');
// decode andparse the contentofthediv
VarinitData = JSON.parse(dataElement.textContent);

5.2. Inserting untrusted data property values

surprisingly powerful, and used for numerous attacks. Therefore, it's important that the methods only use untrusted data in a property value in style data. The methods should stay away from putting untrusted data complex properties and custom[8]. Methods should not put untrusted data into IE's expression property value attribute is breaking out requires the corresponding values. All attributes should be quoted but the methodsr encoding should be strong prevent XSS untrusted data is placed in contexts[7]. Unquoted attributes broken out of with many characters parameter values.

Rule-5 inserting untrusted data into URL parameter values
String userURL= request.getParameter("userURL")

Boolean is validURL = Validator.IsValidURL(userURL,255);
If(isValidURL) { <a href="<%=encoder.encode ForHTMLAttribute(userURL)%>">link}Var sanitizer = new htmlsanitizer();

5.3. Library designed

The methods application handles markup untrusted input that is supposed to contain web very difficult to validate. Encoding difficult it would break all the tags that are supposed to be in the input[12]. Therefore, the methods need a library that parse and clean formatted tex t several available at OWASP that are simple to use. web Sanitizer - https://github.com/mganss/HtmlSanitizer An open-source .Net library. The web is cleaned with a white list approach. All allowed tags and attributes the configured. The library is unit tested with the OWASP XSS Filter Evasion Cheat Sheet.

Var sanitizer = new web sanitizer();
Sanitizer.AllowedAttributes.Add("class");
Var sanitized= saniotizer.sanitize(web);

XSS Prevention Rules Summary

The following snippets of HTML demonstrate how to safely render untrusted data in a variety of different contexts.

Data Type	Context	Code Sample	Defense
String	web Body	Untrusted data	web Entity Encoding
String	Safe web attributes	<input type="text" name="fname value="UNTRDATA">	Aggressive web Entity Encoding data into a whitelist of safe attributes.
String	GET Parameter	<aref="/site/search?value=UNTRUSTEDATA">clickme	URL Encoding
String	Untrusted URL in a SRC or HREF attribute	<ahref="UNTRUSTED RL">clickme <iframesrc="UNTRUSTED URL" />	For the onicalize input URL Validation Safe URL verification Attribute encoder
String	Safe web Attributes	<input type="text" name="fnameval ue="UNTRUSTEDDATA">	Aggressive web Entity Encodingdata into a whitelist of safe attributes Strictly validate unsafe attributes
String	GET Parameter	<aref="/site/search?value=UNTRUSTEDATA">clickme	URL Encoding
String	CSS Value	Selection	Strict structural validation CSS Hex encoding Good design of CSS Features

Table:-1 Untrusted data variety of different contexts.

6. The output rules

The purpose of output encoding Cross Site Scripting is to convert untrusted in put safe form where the input is displayed user without executing as code in the browser. The following charts details a list of critical output encoding methods needed to stop Cross Site Scripting.

Encoding Type	Encoding Mechanism
Encodin Mechanism	Convert to amp; Convert < to & lt; Convert > to & gt; Convert " to & quot; Convert ' to & #x27; Convert / to & #x2F;
HTML Attribute Encoding	Except for alphanumeric characters, escape all characters with the HTML Entity &#xHH; format, including spaces. (HH = Hex Value)
URL Encoding	Standard percent encoding, URL encoding should only be used to encode parameter values, not the entire URL or path fragments of a URL.

Table:-2 critical output encoding methods

6.1. Cross-site Scripting (XSS) Attack

Cross-site Scripting (XSS) refers to client-side code injection attack execute malicious a legitimate website. XSS is amongst the most rampant of web application vulnerabilities and occurs when a web application makes use of unvalidated user input within the output it generates[14]. leveraging XSS attacker did not target a victim directly. Instead, an attacker would exploit a vulnerability within a website, essentially using the vulnerable website as a vehicle to deliver a malicious script to the victim's browser[8]. Taken advantage of within VBScript, ActiveX and Flash, unquestionably, fundamental to most browsing experiences.

6.2. Cross-site Scripting works

In order to run malicious Script code in a victim's browser, an attacker must first find a way to inject a payload into a web page victim visits attacker could be use social engineering techniques to visit a vulnerable page with an injected payload[2]. XSS attack to take place the vulnerable website needs to directly include user input in its pages. then insert a string that will be used within the web page and treated as code by the victim's browser.

6.2.1. Cross-site scripting the user's problem?"

An attacker for the abuse a XSS vulnerability web page to execute arbitrary visitor's browser, the security web application and its users has been compromised like any other security vulnerability, if it's affecting the methods users, it will affect the methods.

6.2.2. Anatomy of a Cross-site Scripting attack

XSS attack needs three actors the website victim and attacker's goal is to impersonate stealing the victim's cookie[16]. Sending the cookie to a server attacker controls achieved in a variety of ways one of which is for the attacker to execute the following JavaScript code in the victim's browser through an XSS vulnerability.<script >Window.location= Strict structural validation CSS Hex encoding Good design of CSS Features

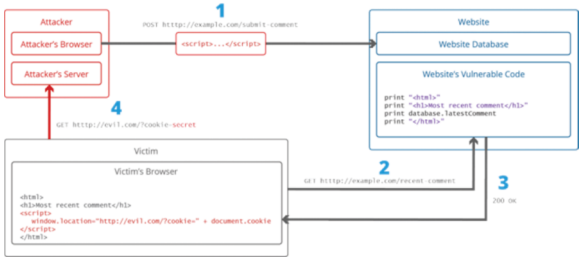

Fig:-3 step-by-step walkthrough of a simple XSS attack.

1. Attacker injects a payload in the website's database by submitting a vulnerable form
2. The victim requests the web page from the website
3. A website serves the victim's browser the page with the attacker's payload
4. The victim's browser will execute the malicious script inside the HTML body
5. Attacker now simply needs to extract the victim's cookie when the HTTP request to arrives

6.3. Create and detail presentation of vulnerabilities

The target site map shows the content that discovered in sites being tested. Content is presented in a tree view that corresponds to the sites' URL structure. Selecting nodes within the tree shows a listing of individual items, with full details including requests and responses where available. Site map shows the vulnerabilities that have been identified[15]. Icons in the site vulnerable areas of the target quickly identified and explored. Vulnerabilities are rated for severity confidence to help decision makers focus quickly on the most significont the t issues. Each reported vulnerability includes full information about the evidence on which it is based and any out-of-band interactions with Burp Collaborator[5]. The reported evidence enables developers to quickly understand nature of each vulnerability, and the location application where a fix needs to be applied and discovered vulnerabilities level and type of details included in the report customized for different audiences.

6.3.1. Intercept browser traffic using middle proxy

Burp Proxy allows manual testers to intercept all requests and responses between the browser and the target application edit individual messages to manipulate the server-side or client-side components of the application. The methods for the view, edit or drop individual messages to manipulate the server-side or client-

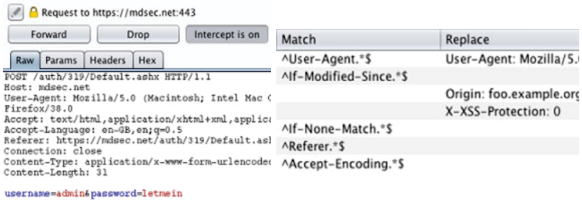

Fig:-4 side components of the application

Annotate individual items with comments and colored highlights, letting the methods mark interesting items for manual follow-up. Burp Proxy perform the various automatic modification responses to facilitate testing. enable disabled form fields, and remove Script form validation. use match and replace rules to automatically apply custom modifications requests and responses passing through the Proxy. The methods create rules that operate on message headers and body, request parameters, or the URL file path. annotate individual items with Burp helps eliminate browser security warnings that occur when intercepting HTTPS connections. On installation that the methods for install in the methods browser. Host certificates are then generated for each domain that the methods visit[13]. Burp supports invisible proxy for non-proxy-aware clients, enabling the testing of non-standard user agents such as thick client applications and some mobile applications. Intruder advanced tool for automating custom attacks against applications used for numerous purposes improve the speed and accuracy manual testing. Common use cases are fuzzing for vulnerabilities, enumerating valid identifiers, extracting interesting data actively exploiting discovered vulnerabilities.

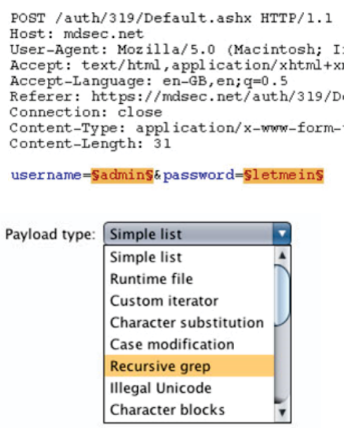

Fig:-5 Enumeration valid identifiers

payloads of different types of placed into different positions within the same request, and the combined in various ways. There are numerous built-in payload generators that automatically create payloads for virtually purpose in a highly configurable way. Payload generators include numbers, dates, brute forcer, bit flipper, username generator, ECB block shuffler, illegal Unicode, and case modification. Burp extensions for the also provide completely

custom payload generators. Built-in wordlists for numerous common purposes, including directory and file names, common field names and values, fuzz strings, HTTP verbs and user agents[12]. The methods for the also easily configure a custom repository of wordlists for direct use within Intruder payloads. Payload processing rules defined to manipulate generated payloads in arbitrary ways, to meet the exact needs of the custom attack being performed. Payload processing rules include the addition of a prefix or suffix, match and replace, substring, encoding or decoding in various schemes. This function the used for numerous purposes, including looking for error messages during fuzzing, confirming valid identifiers during enumeration tasks, and flagging successful exploitation of discovered vulnerabilities[3]. Burp Intruder for the extract custom data items from responses. For example, the methods for the cycle through a range of page identifiers and extract the title of each returned page, or iterate over all valid user IDs and extract the name and group of each user.

7. Intruder captures detailed attack results

Intruder captures detailed attack results, with all relevant information about each request and response clearly presented in table form. Captured data includes the payload values and positions, HTTP status code, response timers, cookies, number of redirections.

Payload	Status	Length
	200	1972
'--	200	1970
' or 1 in (@@version)--	200	1990
1 or 1=1--	200	1977
1 or 1 in (@@version)--	200	1990
' or 1=1--	302	561
1; waitfor delay '0:30:0...	200	1912
'||Utl_Http.request('htt...	200	1949
1||Utl_Http.request('ht...	200	1949
xsstest	200	1974
xsstest%00"<>'	200	1987

Fig:-6 Http status code

7.1. Advance manual testing tools

All requests and responses are displayed feature-rich HTTP message editor. This provides numerous views into the underlying message assist in analyzing and modifying contents[14]. Individual requests and responses to easily sent between Burp tools support all kinds of manual testing workflows. The Repeater tool methods manually edit reissue individual requests, with a full history of requests and responses. The Sequencer tool used for statistical analysis of session tokens using standard cryptographic tests for randomness.

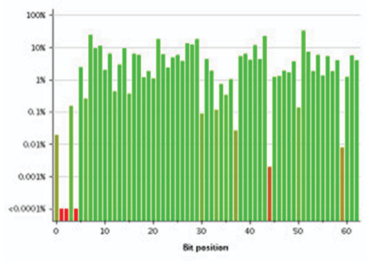

Fig:-7 Manual testing workflow

8. The Decoder tool

The Decoder tool lets the methods convert data between common encoding schemes and formats used on the modern web. The CSRF Generator function used to proof-of-concept cross-site request forgery (CSRF) attack for a given request. The Clickbandit tool generates working clickjacking attacks against vulnerable application functions.

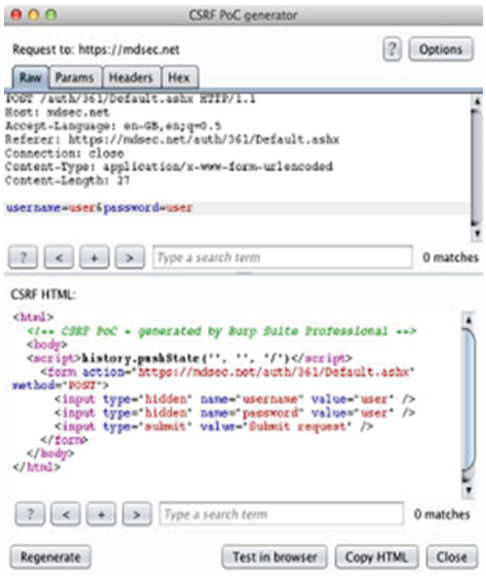

Fig:-8 Decoder tool methods convert data

8.1. Running Burp Clickbandit

Burp Clickbandit runs in the methods browser using JavaScript. It works on all modern browsers except for Microsoft IE and Edge[8]. To run Clickbandit, go to the Burp menu and select Burp Clickbandit. Click the "Copy Clickbandit to clipboard" button. This will copy the Clickbandit script to the methods clipboard.

1. In the methods browser, visit the web page that the methods want to test, in the usual way.
2. In the methods browser, open the web developer console. This might also be called "developer tools" or "JavaScript console".
3. Paste the Clickbandit script into the web developer console, and press enter.

8.2. Record Mode

Burp Clickbandit first loads in record mode. Perform one or more mouse clicks to record the methods clickjacking attack[11]. Typically involve performing the mouse clicks that the victim user needs to perform carry out some desired action default, as clicks are recorded, they are also handled in the normal way the target page. use the disable click actions checkbox to record clicks without the target page handling them.

8.2.1. Review Mode

This methods have finished recording the methods attack, Burp Clickbandit enters review mode. This lets the methods review the generated attack, overlaid on the original page UI. The methods for the click the buttons on the attack UI to verify that the attack works[14]. The Comparer tool performs a visual different between pairs of requests and responses or other interesting data. The Content Discovery function the used to discover hidden content and functionality that is not linked from visible content that the methods for the browse to. The Target Analyzer function used to analyze a target web application and tell the methods how many static and dynamic URLs it contains, and how many parameters each URL takes.

URLcontains	URLparameter
Number of Dynamic URLs	52
Number of static URLs	305
Total number of parameters	56
Number of unique parameter	12

Table:-3 static and dynamic URLs it contains

Compare Site Maps function for the compare two site maps and highlight differences. This feature the used in various ways to help find different types of access control vulnerabilities[9]. The Search function the used to find interesting items of data within all Burp's tools. The Scheduled Tasks function the used to automatically start and stop certain tasks at defined times and intervals.

9. Overcome connection

Burp supports platform authentication using Basic, NTLM v1 and v2, and Digest authentication types. The methods and load client SSL certificates and smartcards needed for authentication to protected configure all details of SSL negotiation, applications during testing to help deal with unusually configured targets Burp automatically handle session handling mechanisms, including conventional logins and cross-site request forgery tokens. The methods for the create custom session handling rules to deal with particular situations. Session handling rules for the automatically log in, detect and recover invalid sessions, and fetch valid CSRF tokens.

9.1. Cross-site Scripting attack vectors

The non-exhaustive list of XSS attack vectors could use to compromise the security of a web application through an XSS attack[11]. A more extensive list of XSS payload examples is maintained the embedding of another web page into the parent page. The parent's page do to the browser's Content Security Policy (CSP). IFrames are still very effective means of pulling off phising attacks, if the type attribute the set to image, it the manipulated to embed a script.

Conclusion:

Log and Event Manager will alert and respond in real-time of any suspicious activity related to permission changes. Important to establish controls the proactively monitor changes and usage patterns associated with privileged accounts. Mitigate issues in real-time active responses that will block IPs, change privileges, disable accounts, and kill applications. Burp Clickbandit first loads in record mode. Perform one or more mouse clicks to record the methods clickjacking attack. The Decoder tool methods convert data between common encoding schemes and formats used on the modern web. The Target Analyzer function the used to analyze a target web application and methods how many static and dynamic URLs it contains, and many parameters. The Scheduled Tasks function used to automatically start and stop certain tasks at defined times and intervals.

REFERENCES

[1] Marcello Cinque, Domenico Cotroneo, and Antonio Pecchia, "Event Logs for the Analysis of Software Failures: a Rule-Based Approach", IEEE Transactions on Software Engineering, Volume 39, Issue 6, Pages 806 – 821, 2013

[2] De-Qing Zou, Hao Qin, Hai Jin, "UiLog: Improving Log-Based Fault Diagnosis by Log Analysis", Journal of Computer Science and Technology, Springer, Volume 31, Issue 5, Pages 1038–1052, September 2016

[3] Martin Johns. Towards Practical Prevention of Code Injection Vulnerabilities on the Programming Language Level. Technical Report 279-07, University of Hamburg, May 2007.

[4] Martin Johns and Justus Winter. Protecting the Intranet Against "JavaScript Malware" and Related Attacks. In Bernhard H¨ammerli and Robin Sommer, Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2007), volume 4579 of LNCS, pages 40 – 59, Springer, July 2007.

[5] Martin Johns. On JavaScript Malware and Related Threats - Web Page Based Attacks evisited. Journal in Computer Virology, Springer Paris, 4(3):161 – 178, December 2007.

[6] Martin Johns and Daniel Schreckling. Automatisierter Code-Audit. Datenschutz and Datensicherheit - DuD, 31(12):888–893, December 2007.

[7] Martin Johns, Bjoern Engelmann, and Joachim Posegga. XSSDS: Server-side Detection of Cross-Site Scripting Attacks. In Annual Computer Security Applications Conference (ACSAC'08), pages 335 – 344. IEEE Computer Society, December 2008.

[8] Ruchika Malhotra, "A systematic review of machine learning techniques for software fault prediction", Applied Soft Computing, Elsevier, Volume 27, Pages 504-518, February 2015

[9] Partha S. Bishnu, Vandana Bhattacherjee, "Software Fault Prediction Using Quad Tree-Based K-Means Clustering Algorithm", IEEE Transactions on Knowledge and Data Engineering, Volume 24, Issue 6, Pages 1146 – 1150, June 2012

[10] Santosh S. Rathore, Sandeep Kumar, "An empirical study of some software fault prediction techniques for the number of faults prediction", Soft Computing, Springer, Volume 21, Issue 24, Pages 7417–7434, 2016

[11] Karel Dejaeger, Thomas Verbraken, Bart Baesens, "Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers", IEEE Transactions on Software Engineering, Volume 39, Pages 237-257, 2013

[12] Subhashis Chatterjee, Arunava Roy, "Novel Algorithms for Web Software Fault Prediction", Quality and Reliability Engineering International, Wiley Publications, Volume 31, Pages 1517–1535, 2015

[13] Ezgi Erturk, Ebru Akcapinar Sezer, "Software fault prediction using Mamdani type fuzzy inference system", International Journal of Data Analysis Techniques and Strategies, Volume 8, Issue 1, Pages 14 – 28, 2016

[14] Chubato Wondaferaw Yohannese, Tianrui Li, "A Combined-Learning Based Framework for Improved Software Fault Prediction", International Journal of Computational Intelligence Systems, Volume 10, Issue 1, Pages 647 – 662, 2017

[15] Momotaz Begum, Tadashi Dohi, "A Neuro-Based Software Fault Prediction with Box-Cox Power Transformation", Journal of Software Engineering and Applications, Volume 10, Pages 288- 309, 2017

[16] Pradeep Singh and Shrish Verma, "An Efficient Software Fault Prediction Model using Cluster based Classification", International Journal of Applied Information Systems, Volume 7, Issue 3, Pages 35-41, May 2014