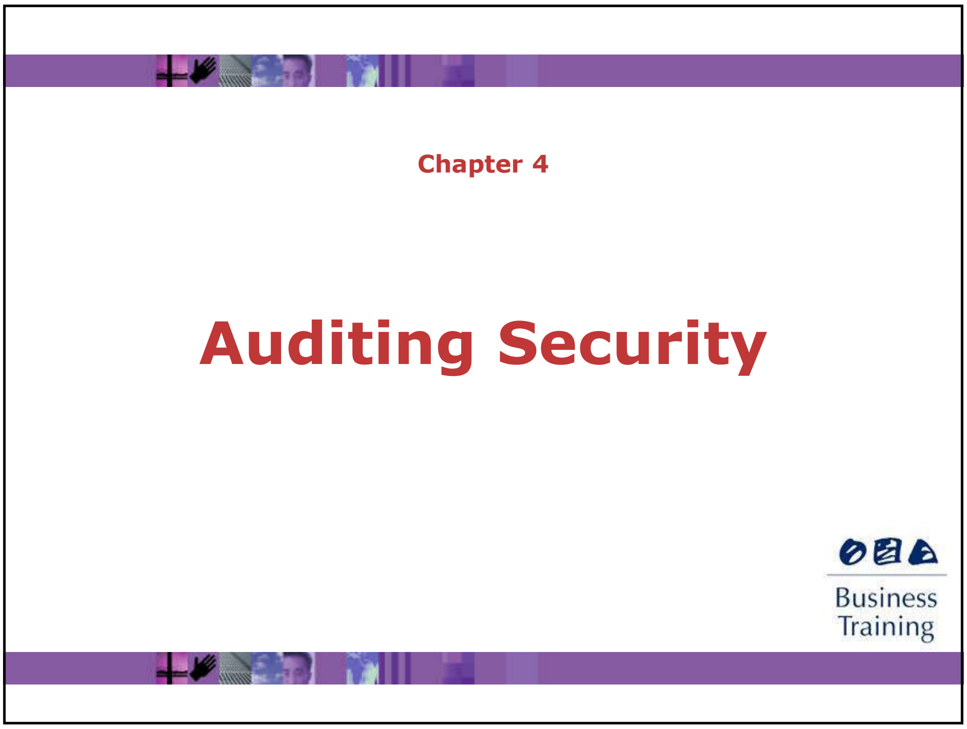
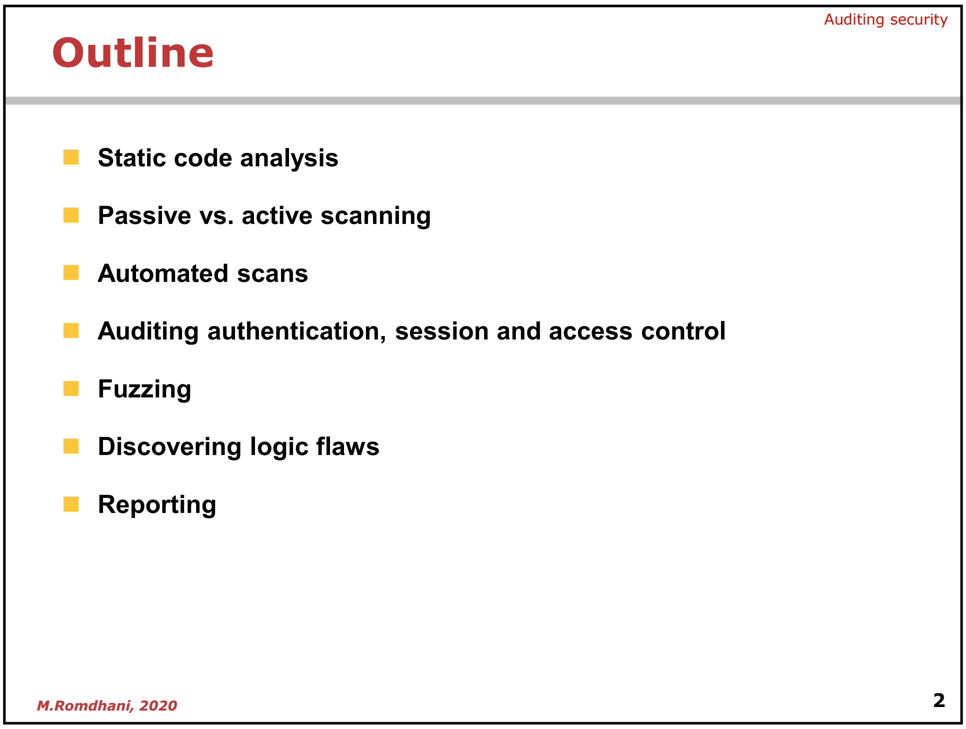


Chapter 4

Auditing Security

Business Training

1



Outline

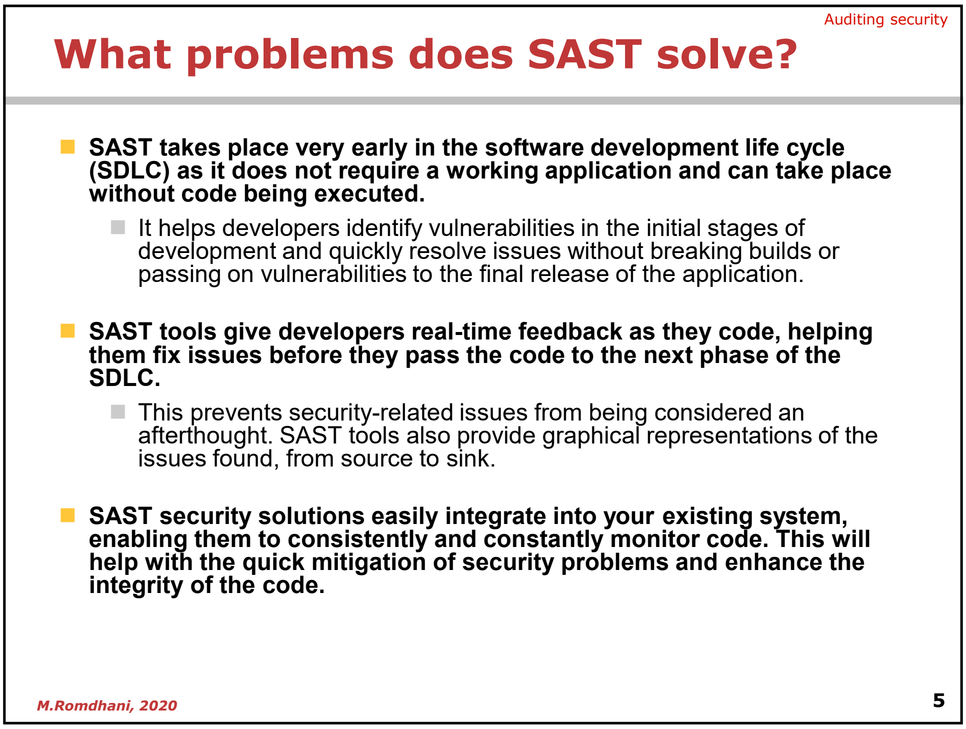
- Static code analysis
- Passive vs. active scanning
- Automated scans
- Auditing authentication, session and access control
- Fuzzing
- Discovering logic flaws
- Reporting

Static code analysis (SAST)

What is Static Code Analysis (aka SAST) ?

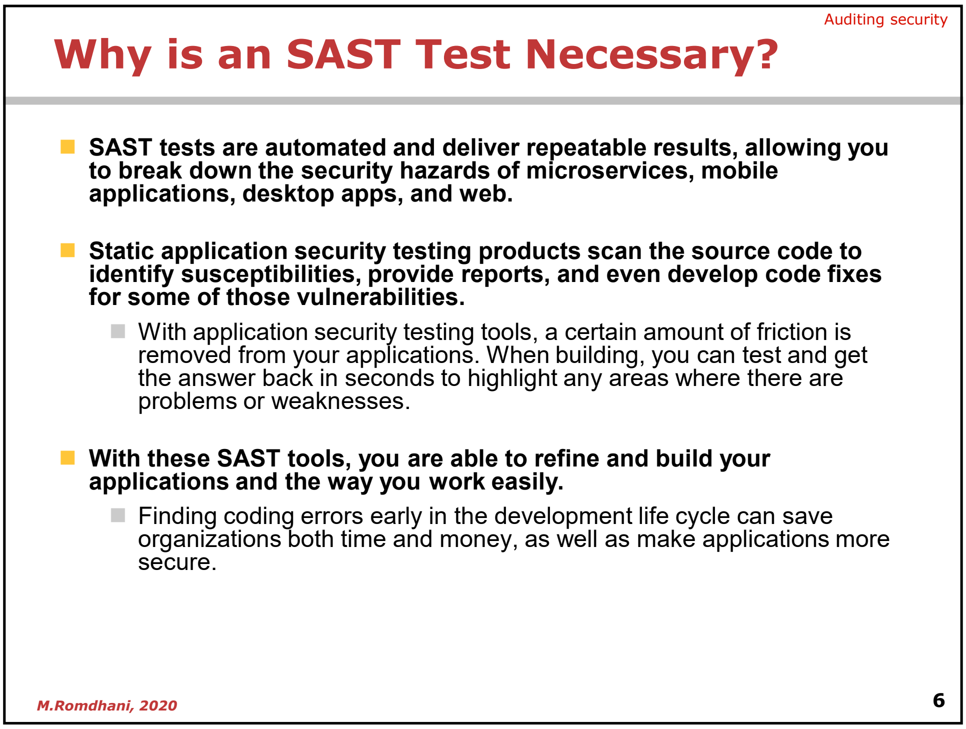
- **Static application security testing (SAST) is a type of security testing that relies on inspecting the source code of an application. In general, SAST involves looking at the ways the code is designed to pinpoint possible security flaws. Source:** Technopedia
  - SAST is a set of technologies designed to analyze application source code, byte code and binaries for coding and design conditions that are indicative of security vulnerabilities.

- **SAST solutions analyze an application from the “inside out” in a nonrunning state.**



What problems does SAST solve?

- **SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed.**
  - It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.
- **SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC.**
  - This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink.
- **SAST security solutions easily integrate into your existing system, enabling them to consistently and constantly monitor code. This will help with the quick mitigation of security problems and enhance the integrity of the code.**



Why is an SAST Test Necessary?

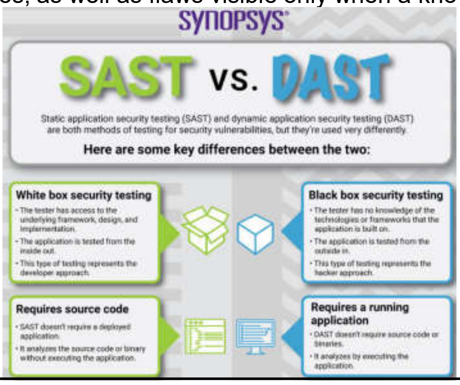
- **SAST tests are automated and deliver repeatable results, allowing you to break down the security hazards of microservices, mobile applications, desktop apps, and web.**
- **Static application security testing products scan the source code to identify susceptibilities, provide reports, and even develop code fixes for some of those vulnerabilities.**
  - With application security testing tools, a certain amount of friction is removed from your applications. When building, you can test and get the answer back in seconds to highlight any areas where there are problems or weaknesses.
- **With these SAST tools, you are able to refine and build your applications and the way you work easily.**
  - Finding coding errors early in the development life cycle can save organizations both time and money, as well as make applications more secure.

SAST Tools

- **There are a number of SAST tools—both commercial and open source available to organizations. Here are five of the most popular in each category**
  - Open Source
    - SpotBugs (aka FindBugs/FindSecBugs)
    - JsHint
    - LGTM.com
    - CodeWarrior
    - reshift
  - Commercial
    - Fortify Static Code Analyzer
    - Veracode
    - Sysnopsis Coverity Scan
    - Checkmarx CxSAST
    - IBM AppScan

SAST vs DAST

- **DAST or Dynamic Application Security Testing, also known as “black box” testing, can find security vulnerabilities and weaknesses in a running application, typically web apps.**
  - It does that by employing fault injection techniques on an app, such as feeding malicious data to the software, to identify common security vulnerabilities, such as SQL injection and cross-site scripting.
  - DAST can also cast a spotlight in runtime problems that can’t be identified by static analysis for example, authentication and server configuration issues, as well as flaws visible only when a known user logs in.



Static application security testing (SAST) and dynamic application security testing (DAST) are both methods of testing for security vulnerabilities, but they're used very differently.

Here are some key differences between the two:

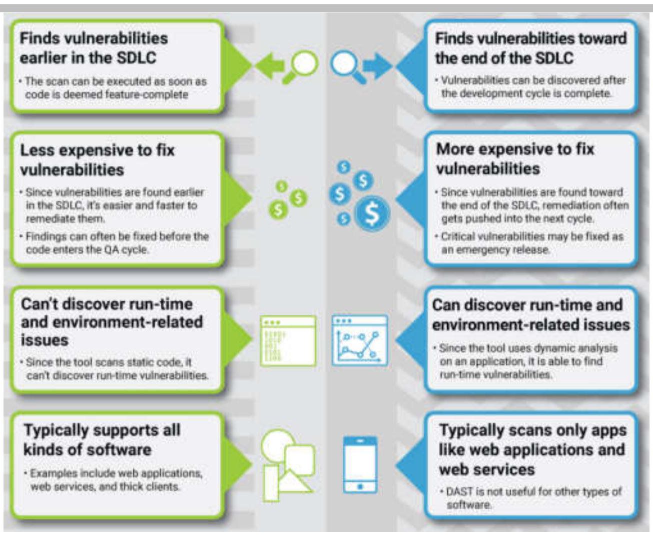
White box security testing
- The tester has access to the underlying framework, design, and implementation.
- The application is tested from the inside out.
- This type of testing represents the developer approach.

Black box security testing
- The tester has no knowledge of the technologies or frameworks that the application is built on.
- The application is tested from the outside in.
- This type of testing represents the hacker approach.

Requires source code
- SAST doesn't require a deployed application.
- It analyzes the source code or binary without executing the application.

Requires a running application
- DAST doesn't require source code or binaries.
- It analyzes by executing the application.

SAST vs DAST



Finds vulnerabilities earlier in the SDLC
- The scan can be executed as soon as code is deemed feature-complete

Finds vulnerabilities toward the end of the SDLC
- Vulnerabilities can be discovered after the development cycle is complete.

Less expensive to fix vulnerabilities
- Since vulnerabilities are found earlier in the SDLC, it's easier and faster to remediate them.
- Findings can often be fixed before the code enters the QA cycle.

More expensive to fix vulnerabilities
- Since vulnerabilities are found toward the end of the SDLC, remediation often gets pushed into the next cycle.
- Critical vulnerabilities may be fixed as an emergency release.

Can't discover run-time and environment-related issues
- Since the tool scans static code, it can't discover run-time vulnerabilities.

Can discover run-time and environment-related issues
- Since the tool uses dynamic analysis on an application, it is able to find run-time vulnerabilities.

Typically supports all kinds of software
- Examples include web applications, web services, and thick clients.

Typically scans only apps like web applications and web services
- DAST is not useful for other types of software.

- **SAST and DAST techniques complement each other**
- **They need to be carried of for comprehensive testing**

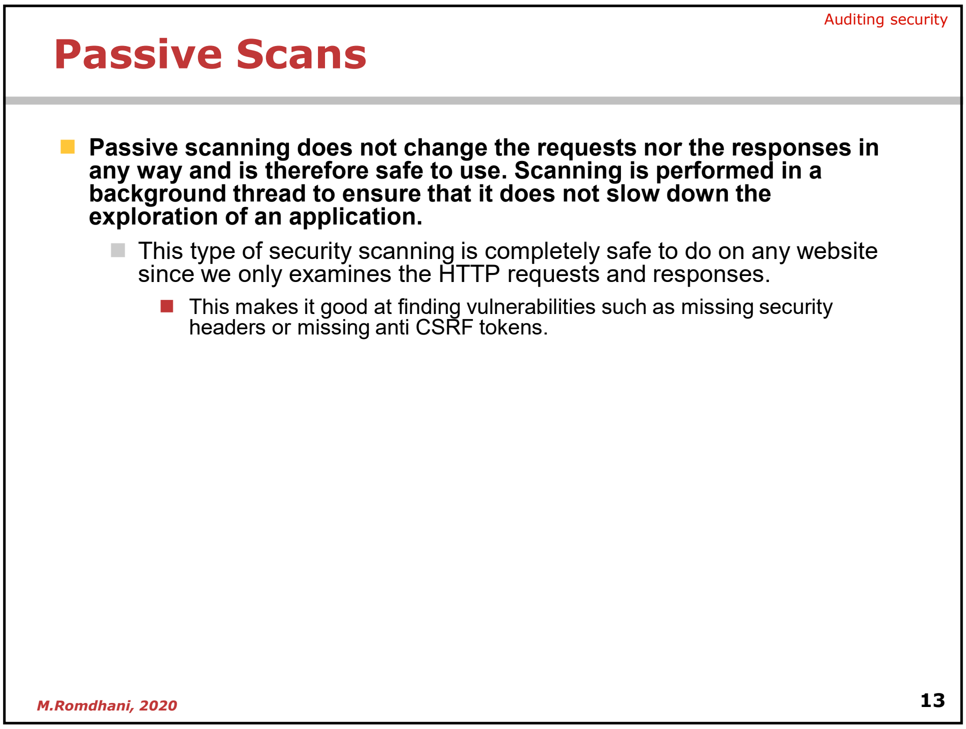
DAST Tools

- **There are a number of DAST tools—both commercial and open source available to organizations. Here are five of the most popular in each category. OWASP maintains a page of known DAST Tools here [https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools]**
  - Open Source
    - OWASP ZAP
    - Arachni
    - Grabber
    - Wapiti
    - Vega
  - Commercial
    - Acunetix Vulnerability Scanner
    - Burp Suite
    - Netsparker
    - CheckMarx
    - Micro Focus Fortify On Demand
    - IBM AppScan

Passive vs. active scanning

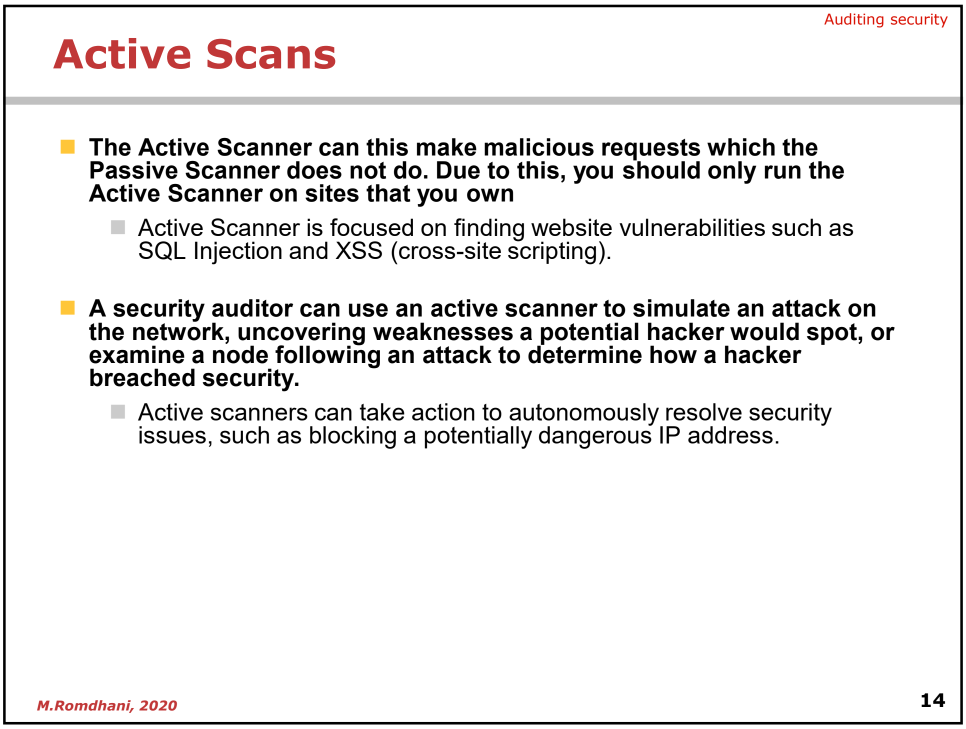
Vulnerability Scanning Basics

- **Vulnerability scanners return data concerning potential security risks that allow IT personnel to view the network the way a potential hacker might, clearly seeing the potential avenues for denial of service attacks or gaining information through packet sniffing.**
- **Vulnerability scanners often prioritize the weaknesses they discover, assigning different values to represent the potential damage a hacker could cause within a network by exploiting a specific weakness.**
- **Vulnerability scanners are the tools used to perform the vulnerability scanning. Vulnerability scanners have a database of vulnerabilities based on which it performs the check on the remote host.**
  - The vulnerability database contains all the information required (service, port, packet type, a potential path to exploit, etc.) to check the security issue.
  - They can scan the network and websites against thousands of vulnerabilities, provide the list of issues based on the risk and suggest the remediation as well.



Passive Scans

- **Passive scanning does not change the requests nor the responses in any way and is therefore safe to use. Scanning is performed in a background thread to ensure that it does not slow down the exploration of an application.**
  - This type of security scanning is completely safe to do on any website since we only examines the HTTP requests and responses.
    - This makes it good at finding vulnerabilities such as missing security headers or missing anti CSRF tokens.



Active Scans

- **The Active Scanner can this make malicious requests which the Passive Scanner does not do. Due to this, you should only run the Active Scanner on sites that you own**
  - Active Scanner is focused on finding website vulnerabilities such as SQL Injection and XSS (cross-site scripting).
- **A security auditor can use an active scanner to simulate an attack on the network, uncovering weaknesses a potential hacker would spot, or examine a node following an attack to determine how a hacker breached security.**
  - Active scanners can take action to autonomously resolve security issues, such as blocking a potentially dangerous IP address.

Automated scans

Automated VS Manual Web Security Scanning

- **Traditional security testing methods like manual web app penetration testing are fragmented and time-consuming, they tend to cause unacceptable delays in the development process.**
  - Web Security testing requires, above creativity and intuition, cycles consisting of hundreds of repetitive tests.
- **Automated code testing tools can help by accelerating testing and taking the responsibility for testing off the developer's plate.**
- **Automated Web security testing offers excellent coverage of the web application by performing thousands of tests in a few hours:**
  - In a web application with hundreds of possible attack vectors, where the automated web security scanner never skips an input or neglects a field.
  - While the faults discovered by the scanner are fixed (and, in turn, verified by the web security scanner), the testers invest their time in researching and testing logical vulnerabilities, where their intelligence and skill are truly

Automated Security Benefits

- **Automated Security testing brings these benefits:**
  - **Greater accuracy**. Static Analysis Security Testing (SAST) And Dynamic Analysis Security Testing (DAST) automated code testing technology identify flaws and vulnerabilities that traditional source code scanners often miss.
  - **Faster testing**. Static scans are completed within few hours. This eliminates the need to stop development to accommodate testing.
  - **Integrated tools**. Automated scans are integrated into the software development lifecycle with APIs and plug-ins, so developers never have to interrupt coding to open a separate testing system.

Keeping Up with Changes in Web Applications

- **As developers face increasing pressure to deliver software more quickly, automated code testing tools can help to effectively and painlessly inject security into the software development lifecycle (SDLC).**
  - While preparing for the second version of the web based product, application security scanner itself knows which tests to perform and how
  - On repeated scan delta reporting ensures that findings only need to be judged when they first appear in the scan results or when their output changes.

Auditing authentication, session and access control

19

Auditing Authentication and Session management

- **Authentication and authorization problems are prevalent security vulnerabilities. In fact, they consistently rank second highest in the OWASP Top 10.**
- **General Guidelines on Testing Authentication**
  [OWASP Web Testing Guide**]**
  - There's no one-size-fits-all approach to authentication. When reviewing the authentication architecture of an app, you should first consider whether the authentication method(s) used are appropriate in the given context. Authentication can be based on one or more of the following:
    - Something the user knows (password, PIN, pattern, etc.)
    - Something the user has (SIM card, one-time password generator, or hardware token)
    - A biometric property of the user (fingerprint, retina, voice)

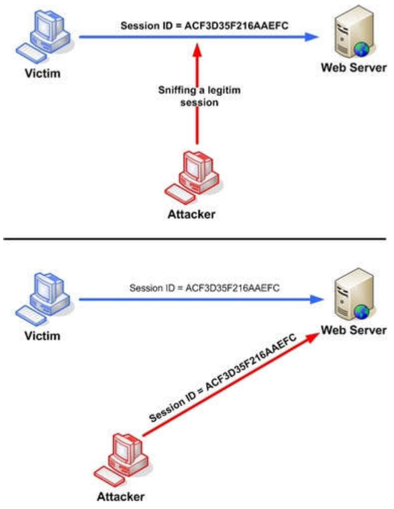
20

Auditing Authentication and Session management

- **How strong is initial user authentication ?**
  - What is your authentication scheme?
  - Are you incorporating two-factor authentication?
  - How safely do you store user credentials?
  - Some form of credentials have to go with every request (initial auth, then session ID)
  - Should use SSL for everything requiring authentication
- **Session management flaws**
  - SESSION ID used to track state since HTTP doesn’t
    - and it is just as good as credentials to an attacker
  - SESSION ID is frequently exposed on the network, in browser, in logs, …
- **Beware the side-doors**
  - Change my password, remember my password, forgot my password, secret question, logout, email address, etc…
- **Typical impact**
  - User accounts compromised or user sessions hijacked

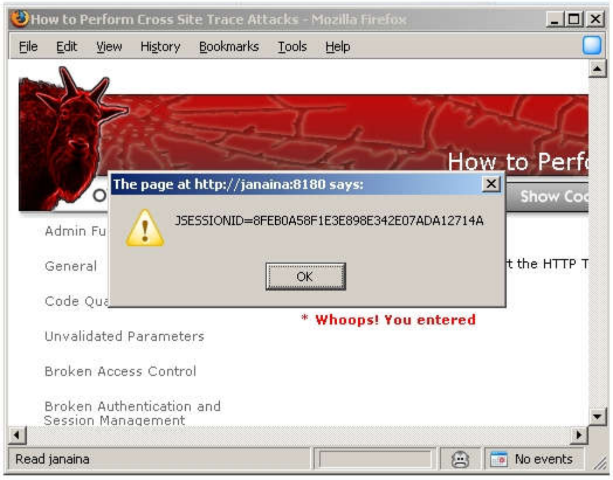
Broken Authentication Illustrated

- **Example 1:**
  Session Sniffing [https://www.owasp.org/index.php/Session_hijacking_attack]


Session ID = ACF3D35F216AAEFC
Victim
Web Server
Sniffing a legitim session
Attacker
Session ID = ACF3D35F216AAEFC
Victim
Web Server
Session ID = ACF3D35F216AAEFC
Attacker

Broken Authentication Illustrated

- **Example 2:**
  Cross-site script attack[https://www.owasp.org/index.php/Session_hijacking_attack]



How to Perform Cross Site Trace Attacks - Mozilla Firefox

The page at http://janaina:8180 says:

JSESSIONID=8FEB0A58F1E3E898E342E07ADA12714A

OK

Admin Fu

General

Code Qua

Unvalidated Parameters

Broken Access Control

Broken Authentication and Session Management

t the HTTP T

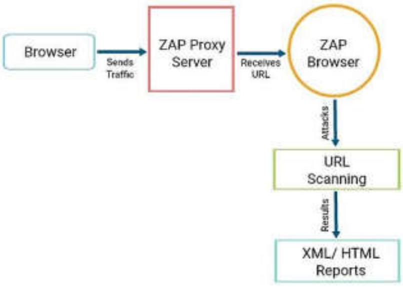
* Whoops! You entered

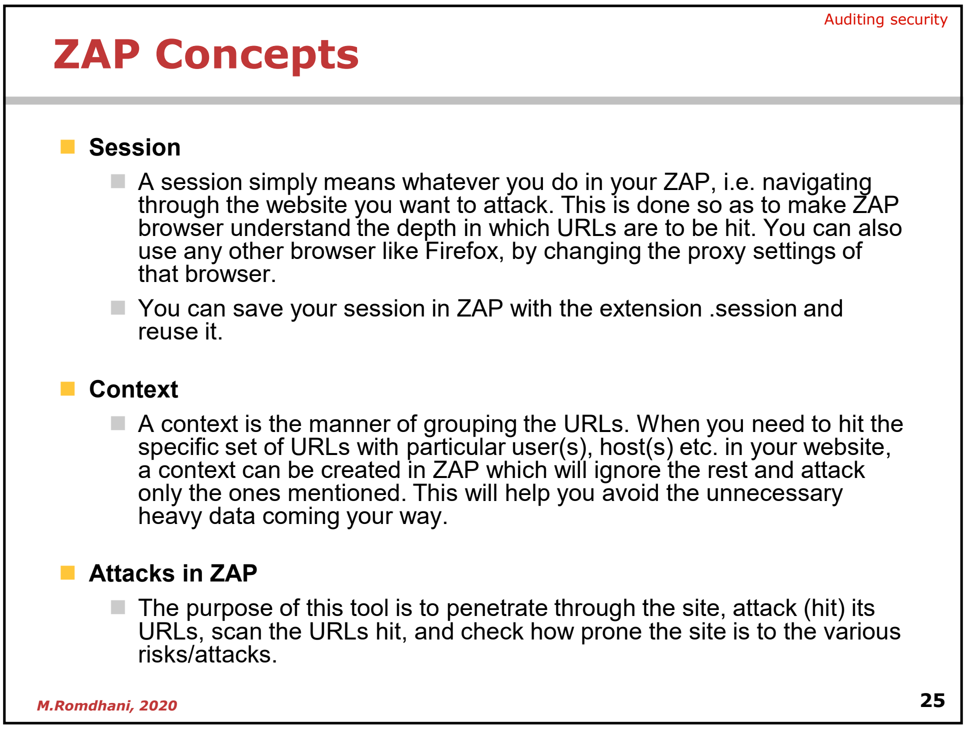
Read janaina

No events

Detecting Broken Authentication with OWASP Zap Proxy

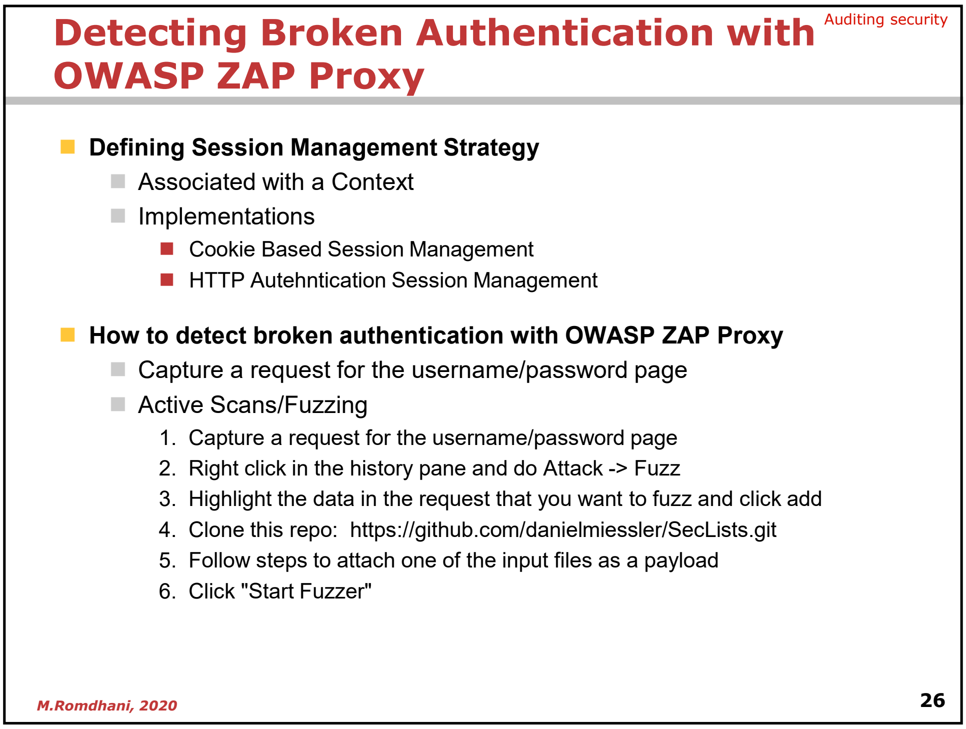
- **What is Zap Proxy ?**
  - An easy to use webapp pentest tool
  - Completely free and open source
  - Ideal for devs, esp. for automated security tests
  - Becoming a framework for advanced testing
- **How does ZAP wok ?**





ZAP Concepts

- **Session**
  - A session simply means whatever you do in your ZAP, i.e. navigating through the website you want to attack. This is done so as to make ZAP browser understand the depth in which URLs are to be hit. You can also use any other browser like Firefox, by changing the proxy settings of that browser.
  - You can save your session in ZAP with the extension .session and reuse it.
- **Context**
  - A context is the manner of grouping the URLs. When you need to hit the specific set of URLs with particular user(s), host(s) etc. in your website, a context can be created in ZAP which will ignore the rest and attack only the ones mentioned. This will help you avoid the unnecessary heavy data coming your way.
- **Attacks in ZAP**
  - The purpose of this tool is to penetrate through the site, attack (hit) its URLs, scan the URLs hit, and check how prone the site is to the various risks/attacks.



Detecting Broken Authentication with OWASP ZAP Proxy

- **Defining Session Management Strategy**
  - Associated with a Context
  - Implementations
    - Cookie Based Session Management
    - HTTP Autehntication Session Management
- **How to detect broken authentication with OWASP ZAP Proxy**
  - Capture a request for the username/password page
  - Active Scans/Fuzzing
    1. Capture a request for the username/password page
    2. Right click in the history pane and do Attack -> Fuzz
    3. Highlight the data in the request that you want to fuzz and click add
    4. Clone this repo: https://github.com/danielmiessler/SecLists.git
    5. Follow steps to attach one of the input files as a payload
    6. Click "Start Fuzzer"

Fuzzing

What is Fuzzing ?

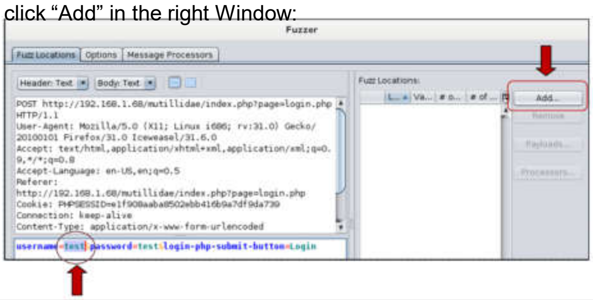
- **Fuzz testing or Fuzzing is a Black Box software testing technique, which basically consists in finding implementation bugs using malformed/semi-malformed data injection in an automated fashion. [OWASP]**
- **A fuzzer would try combinations of attacks on:**
  - numbers (signed/unsigned integers/float...)
  - chars (urls, command-line inputs)
  - metadata : user-input text (id3 tag)
  - pure binary sequences
- **A common approach to fuzzing is to define lists of "known-to-be-dangerous values" (fuzz vectors) for each type, and to inject them or recombinations.**
  - for integers: zero, possibly negative or very big numbers
  - for chars: escaped, interpretable characters / instructions (ex: For SQL Requests, quotes / commands...)
  - for binary: random ones

Why Fuzz ?

- **The purpose of fuzzing relies on the assumption that there are bugs within every program, which are waiting to be discovered. Therefore, a systematical approach should find them sooner or later.**
- **Fuzzing can add another point of view to classical software testing techniques (hand code review, debugging) because of it's non-human approach. It doesn't replace them, but is a reasonable complement, thanks to the limited work needed to put the procedure in place.**
- **Fuzzers advantages**
  - The great advantage of fuzz testing is that the test design is extremely simple, and free of preconceptions about system behavior (from Wikipedia http://en.wikipedia.org/wiki/Fuzz_testing).
- **Fuzzers limitations**
  - Fuzzers usually tend to find simple bugs; plus, the more a fuzzer is protocol-aware, the less weird errors it will find. This is why the exhaustive / random approach is still popular among the fuzzing community.
  - Another problem is that when you do some black-box-testing, you usually attack a closed system, which increases difficulty to evaluate the dangerosity/impact of the found vulnerability (no debugging possibilities).

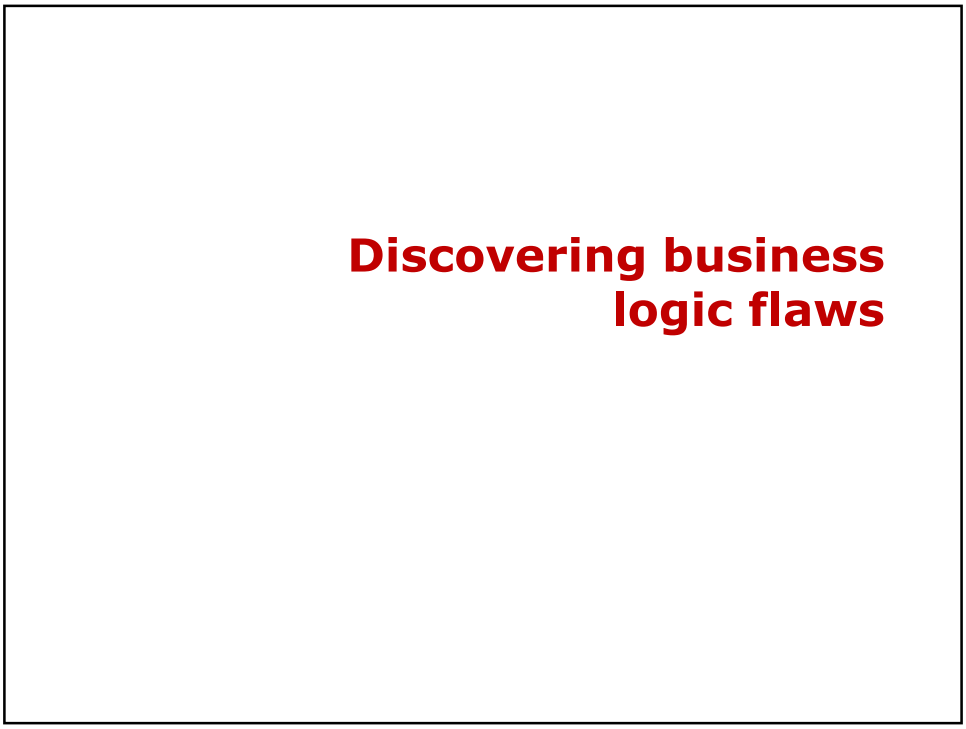
Fuzzing with ZAP

- **Fuzzing**
  - Start ZAP and then set your browser internet proxy settings to localhost:8080.
  - Then surf to the target webpage and login. Surf to a few other pages if you like, entering data as you go, the more site interaction the better.
  - In the sites Window, select the login webpage, right click on it and select, “Attack” and then “Fuzz…” This will open the fuzzer screen. It lists the header text in the top left box, the target query with selectable text in the bottom left box and the fuzz location/tool window on the right.
    - In the bottom left window, highlight the name you used to login as a keyword. I used the username, “test”.
    - Now click “Add” in the right Window:



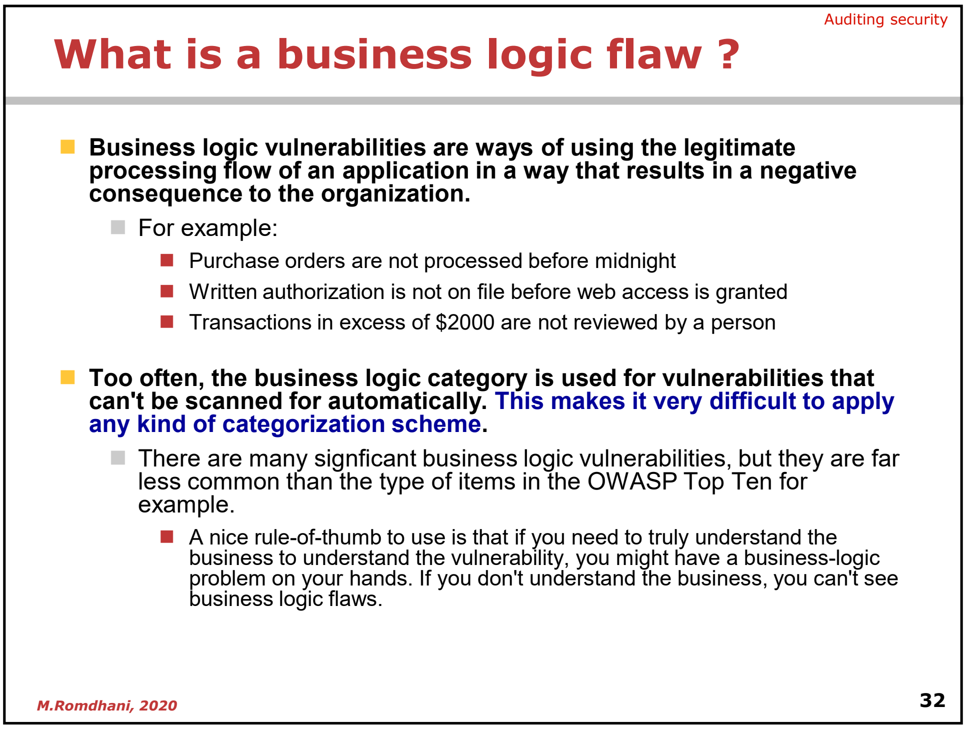
```
POST http://192.168.1.68/mutillidae/index.php?page=login.php
HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:31.0) Gecko/
20100101 Firefox/31.0 Iceweasel/31.6.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.
9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer:
http://192.168.1.68/mutillidae/index.php?page=login.php
Cookie: PHPSESSID=e1f908aaba8502ebb416b9a7df9da739
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
```

```
username=test&password=test&login-php-submit-button=Login
```



Discovering business logic flaws

31



What is a business logic flaw ?

- **Business logic vulnerabilities are ways of using the legitimate processing flow of an application in a way that results in a negative consequence to the organization.**
  - For example:
    - Purchase orders are not processed before midnight
    - Written authorization is not on file before web access is granted
    - Transactions in excess of $2000 are not reviewed by a person
- **Too often, the business logic category is used for vulnerabilities that can't be scanned for automatically. This makes it very difficult to apply any kind of categorization scheme.**
  - There are many signficant business logic vulnerabilities, but they are far less common than the type of items in the OWASP Top Ten for example.
    - A nice rule-of-thumb to use is that if you need to truly understand the business to understand the vulnerability, you might have a business-logic problem on your hands. If you don't understand the business, you can't see business logic flaws.

32

What is a business logic flaw ?

What a business logic flaw is not	What is a business logic flaw
It is not a code based flaw	It is a logical based flaw
It is not a malicious behaviour of the application	It is the legitimate workflow used un a malicious way
It is not about what code is written	It is about how the code is written
It is not what an application does with the input	It is about how the input triggers a sequence of actions pertaining to a prescribed workkflow

How to discover business logic flaws ?

- **Things to consider when testing for business logic flaws:**
  - Whether or not workflows and processes can be automated and then attacked using vulnerability scanners and scripts.
  - Testing both with and without user authentication. (You can often uncover publicly exploitable weaknesses that might otherwise go unnoticed to a logged-in user.)
  - Audit or exception logging and how anomalies are being monitored and addressed.

Vulnerable application logic

- **There are some areas you discover as being weak and exploitable:**
  - Initial logins and how the application and workflow are presented to the user may allow for unauthorized access if the user simply hits "Esc" or clicks the back button in the Web browser.
  - Order and data entries that can be manipulated.
  - Search queries that return interesting information that leads to different
  - Role or privilege escalation that can be exploited by merely knowing where to go within the application.
  - Session hand-offs to separate (often third-party) applications that disclose how the authentication and session management work.
  - File upload areas that facilitate the spreading of malware on unprotected servers.
- **These areas involve core application security principles such as authentication, access control, session management and input validation. It's really all related, but uncovering many business logic flaws takes a special way of thinking and a special eye that can look at the bigger picture.**

An example of business logic flaws

- **An e-commerce merchant, YYY.com sells electronic merchandize to consumers worldwide. The typical checkout process during fulfillment includes the following steps in sequence**
  1. User picks one or more items and adds to basket
  2. User then heads to order page to initiate purchase
  3. User pushes purchase or checkout button
  4. Merchant YYY.com sends order and customer information to it's partner payments processor (for authorization and capture)
  5. Payments processor returns transaction-id back to Merchant YYY.com
  6. Merchant YYY.com displays confirmation details on fulfillment page to consumer
- **An attacker carefully tracks the request/response through each of these stages prepares to induce a currency attack on this merchant.**
  - At step (3), the attacker manipulates a currency related parameter in the POST request within the HTTP header and changes the currency type from `**EU Pounds**` to `**US Dollars**`. As a result the attacker was able to exploit this logic flaw by paying less for his/her order.

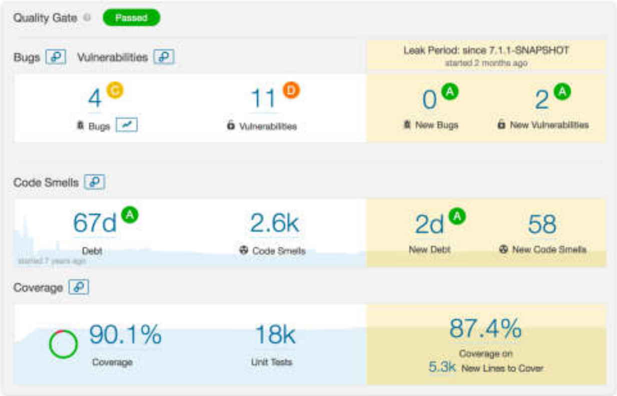
Reporting

Security Reports

- **Security Reports quickly give you the big picture on your application's security, with breakdowns of just where you stand in regard to each of the OWASP Top 10, and SANS Top 25 categories.**
- **The Security Reports are fed by the analyzers, which rely on the rules activated in your quality profiles to raise security issues.**
  - If there are no rules corresponding to a given OWASP category activated in your Quality Profile, you will get no issues linked to that specific category and the rating displayed will be A. That won't mean you are safe for that category, but that you need to activate more rules (assuming some exist).

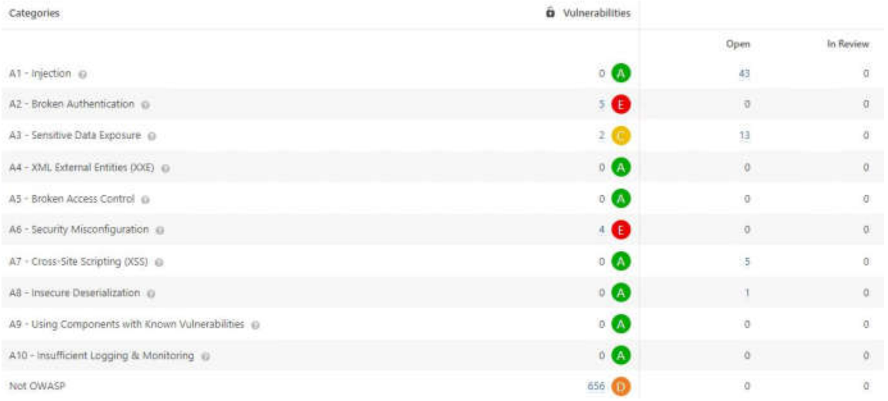
SonarQube Security Reports

- **Dashbord Reporting(Web interface)**
  - It is pssible to generate reports in PDF format with the most relevant information from SonarQube web interface. The report aims to be a deliverable as part of project documentation



SonarQube Security Reports

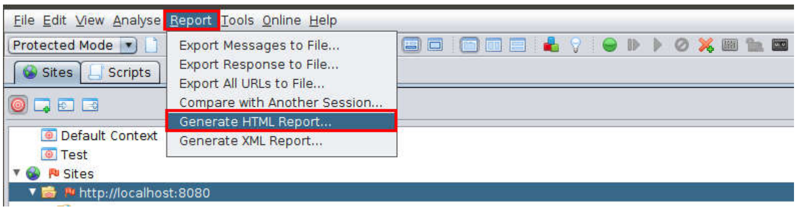
- **SonarQube Reports can be organized as OWASP Top 10 or SANS Top 25**



Categories	Vulnerabilities		
		Open	In Review
A1 - Injection	0 A	43	0
A2 - Broken Authentication	5 E	0	0
A3 - Sensitive Data Exposure	2 C	13	0
A4 - XML External Entities (XXE)	0 A	0	0
A5 - Broken Access Control	0 A	0	0
A6 - Security Misconfiguration	4 E	0	0
A7 - Cross-Site Scripting (XSS)	0 A	5	0
A8 - Insecure Deserialization	0 A	1	0
A9 - Using Components with Known Vulnerabilities	0 A	0	0
A10 - Insufficient Logging & Monitoring	0 A	0	0
Not OWASP	656 D	0	0

ZAP Proxy Reports

- **Once the Active Scan is complete, you can generate the reports for exporting the results of the scan. Go to Report -> Generate HTML Report from the menu.**
  - Then it will prompt where to save the report. Once you provide a file path, it will export the ZAP scan report. By examining the report, you will be able to identify possible security threats and get them fixed.
- **ZAP Can generate reports in many formats PDF, DOC using additional extensions [https://github.com/zaproxy/zap-extensions/wiki/HelpAddonsExportreportExportreport]**


File Edit View Analyse Report Tools Online Help
Protected Mode
Sites
Scripts
Export Messages to File...
Export Response to File...
Export All URLs to File...
Compare with Another Session...
Generate HTML Report...
Generate XML Report...
Default Context
Test
Sites
http://localhost:8080