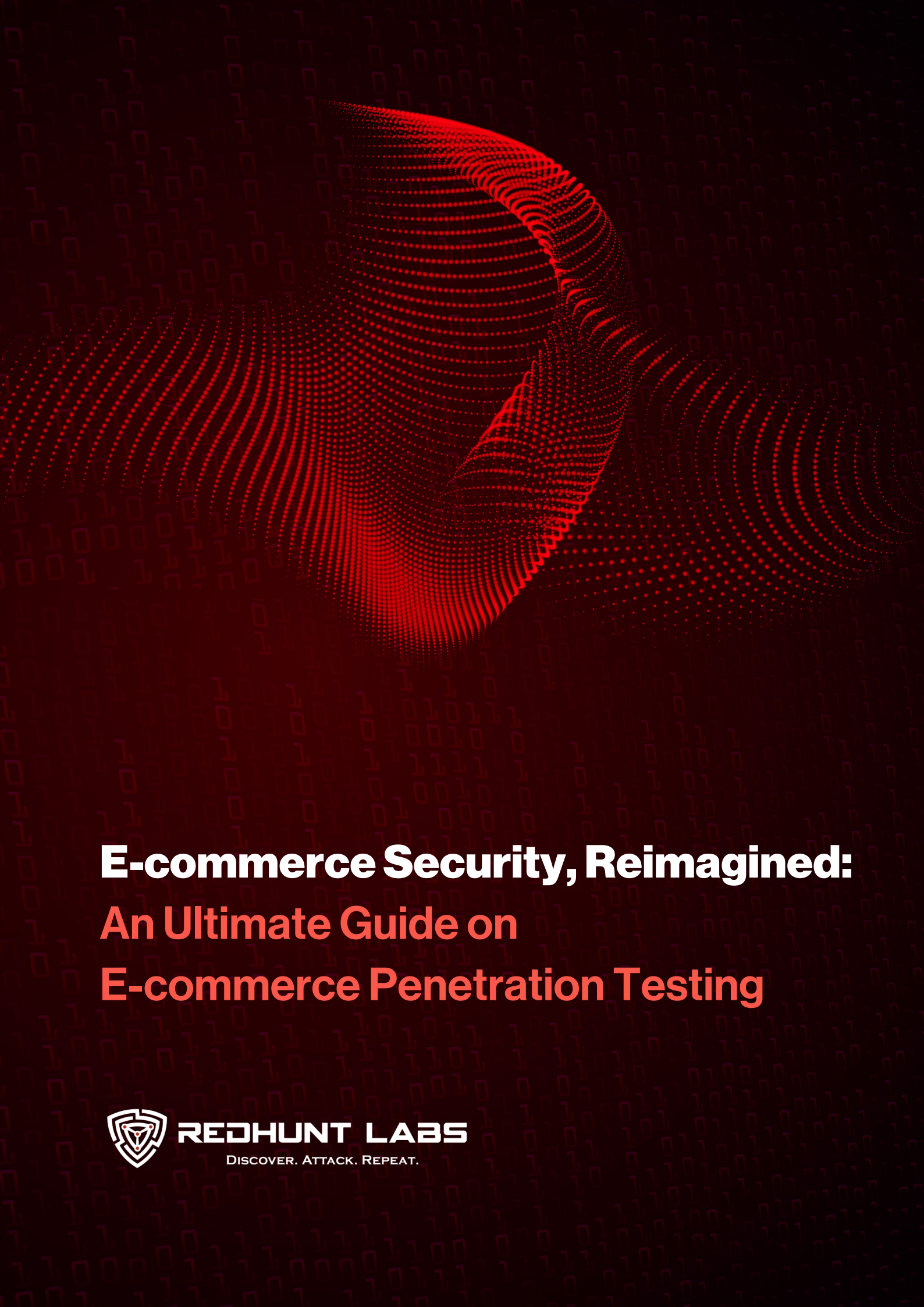




E-commerce Security, Reimagined: **An Ultimate Guide on** **E-commerce Penetration Testing**



REDHUNT LABS

DISCOVER. ATTACK. REPEAT.

TABLE OF CONTENTS



Introduction	3
Importance of E-commerce Security	4
Overview of Penetration Testing	5
Understanding E-commerce Architecture	6
Preparing for Penetration Testing	7
E-commerce Applications:	
Specific Vulnerability Test Cases	9
Generic Vulnerability Test Case	21
Conclusion	31

Introduction

E-commerce applications have revolutionized the way we shop, offering unparalleled convenience and accessibility to consumers worldwide.

With this convenience comes the responsibility to ensure the security of sensitive information, such as personal data and financial transactions. E-commerce applications are inherently susceptible to cyber-attacks due to their extensive surface area, which makes them prime targets for malicious attackers. These applications **encompass web applications, databases, payment gateways, purchase and return functionalities**, and more. Ensuring the security of every aspect is paramount to protecting against a wide range of threats.

Comprehensive penetration testing and robust remediation measures are essential to minimise the risks of potential financial fraud. Although there are many types of web vulnerabilities like **XSS, SQLI, LFI** etc., throughout this comprehensive guide, we will delve into numerous scenarios that illustrate how specific features in an e-commerce application could be vulnerable. These vulnerabilities may lead to **severe consequences**, including the exposure of personally **identifiable information (PII)**, **account takeover**, and various forms of financial and other fraud.

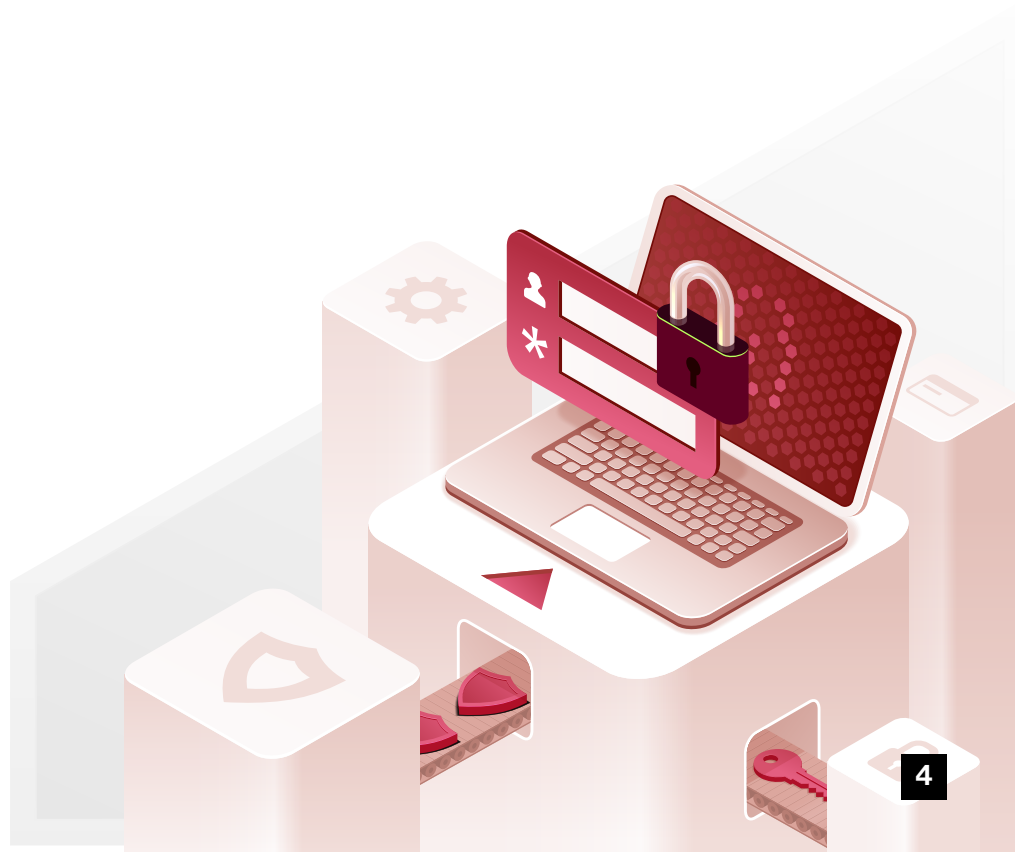
This entire guide is researched and conceptualised by **Utkarsh Agarwal - Senior Security Consultant, RedHunt Labs.**



Importance of E-commerce Security

“

The stakes are high in e-commerce security, with potential consequences ranging from financial losses and reputational damage to legal liabilities and regulatory fines. **Protecting customer data and maintaining trust are paramount for the success** and sustainability of e-commerce businesses. A single security breach can have far-reaching implications, leading to loss of customer confidence, brand erosion, and competitive disadvantage in the market.



Overview of Penetration Testing

Penetration testing, commonly referred to as pen testing, is a proactive approach to **identifying and mitigating security vulnerabilities in e-commerce applications**. It involves simulating real-world attacks to assess the strength of existing security controls and uncover potential weaknesses that could be exploited by attackers.



Penetration testing encompasses various techniques, including **vulnerability scanning, manual inspection, and exploitation of identified vulnerabilities**.



Understanding E-commerce Architecture

Before delving into penetration testing, it's crucial to understand the underlying architecture of e-commerce applications. These platforms typically consist of several interconnected components, including:



Web Servers and Applications: Host e-commerce websites and facilitate user interactions, product browsing, and online transactions.



Databases: Store sensitive information such as customer data, product catalogs, and transaction records.



Payment Gateways: Process financial transactions securely, ensuring compliance with industry standards such as the Payment Card Industry Data Security Standard (PCI DSS).



APIs (Application Programming Interfaces): Enable integration with third-party services, such as shipping providers, inventory management systems, and customer relationship management (CRM) platforms.

The data flow in e-commerce transactions involves multiple stages, starting from user interaction with the **website or mobile app, through product selection, checkout, payment processing, and order fulfillment.**

Understanding this flow is essential for identifying potential points of vulnerability and prioritizing security assessments and testing efforts.



Preparing for Penetration Testing

Setting Objectives and Scope

Before conducting penetration testing, it's essential to define clear objectives and scope to ensure a focused and effective assessment. Objectives may include:



Identifying and prioritizing critical vulnerabilities



Assessing the effectiveness of existing security controls



Evaluating compliance with regulatory requirements



Testing incident response and recovery procedures

The scope of penetration testing should encompass all relevant components of the e-commerce infrastructure, including web applications, databases, payment gateways, APIs, and any third-party integrations. It's essential to establish boundaries and limitations to avoid disruption of production systems and unauthorized access to sensitive data.



Legal and Ethical Considerations

Penetration testing must be conducted in compliance with legal and ethical guidelines to avoid legal liabilities and reputational damage. It's crucial to obtain explicit permission from the organization's stakeholders and adhere to relevant laws and regulations governing cybersecurity testing and data privacy.



Resource Allocation and Planning

Successful penetration testing requires adequate resources, including skilled personnel, tools, and infrastructure. It's essential to assemble a multidisciplinary team with expertise in cybersecurity, web development, database administration, and network security. Additionally, investing in comprehensive testing tools and technologies can enhance the efficiency and effectiveness of penetration testing efforts.

E-commerce Applications: Specific Vulnerability Test Cases

Race Conditions

In E-Commerce applications, there are lots of features on which race conditions might be found. A race condition occurs when multiple threads simultaneously access the same shared code/resource without locking or synchronisation, hence that makes the application behave unexpectedly.

In simple words, when multiple threads are sent in the form of requests, the application threading system fails to handle those requests simultaneously and it could perform unexpected behaviours. For example, in an application, if a user could create only 2 groups, it is possible to create more than 2 groups by performing Race Conditions.

• Common Areas to Look

Use of Coupons - Performing race conditions on coupon codes could allow you to reuse the coupon again and get a heavy discount. Suppose a coupon gives a discount of 30\$, but by performing the race condition, the attacker can get a discount of more than 30\$.

Adding Product to cart - It can be possible to get more quantity of the product at less price as compared to the original price. For example, if a product has a price of 30\$, then it is possible to get 2 quantities with a price less than 60\$.

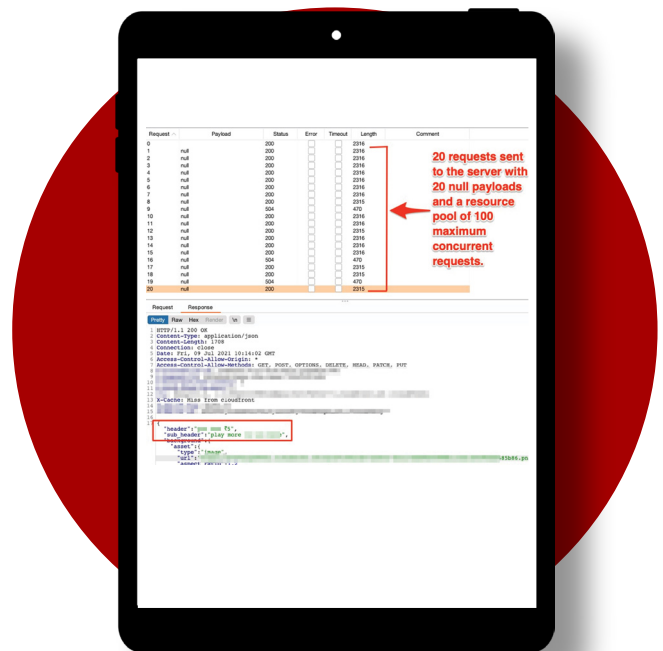
Refund/Cashback Processing - If multiple refund requests are made for the same product, it might be possible to get multiple refunds.



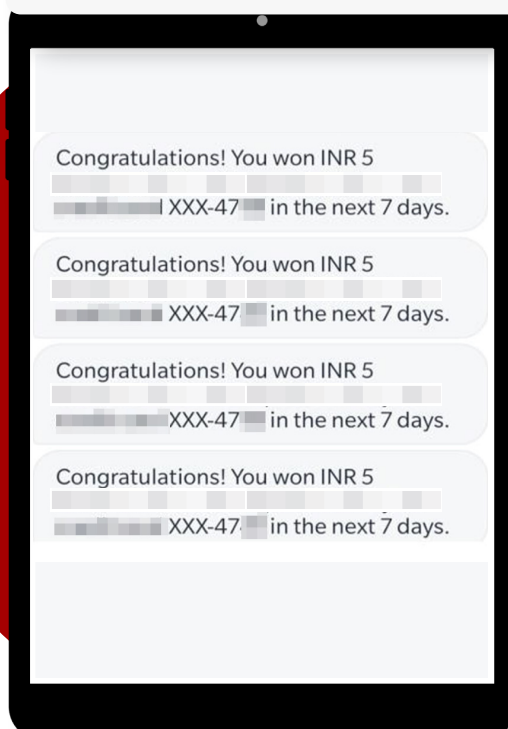
• Scenario/ Real Scenario

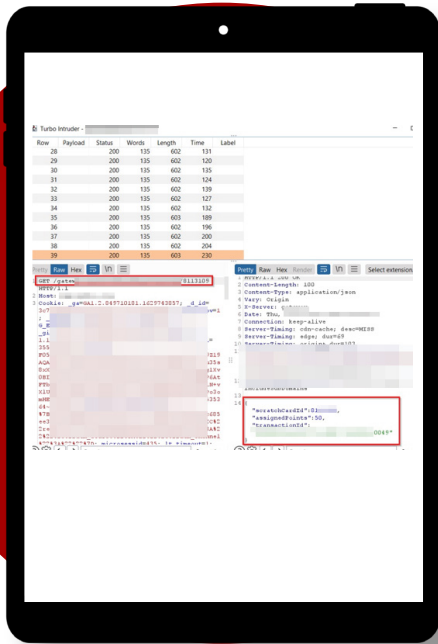
RedHunt Labs performed comprehensive penetration testing for applications and found lots of race conditions on large e-commerce applications. Some of the POCs with explanations are below.

In one of our client's pentest, there was a functionality which offered 5 INR cashback for a single transaction, so we captured the refund request and performed the race condition attack using the intruder feature of the Burp Suite tool.



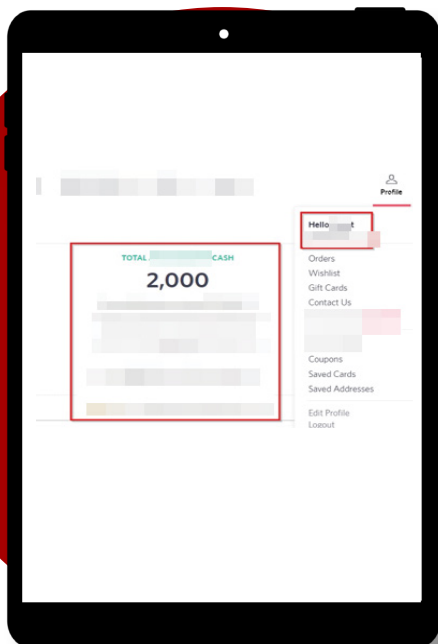
As a result, multiple cashbacks were processed and credited.





In another client's pentest, there was a functionality where the user could refer to someone and both receive a gift voucher of 50 points, and the user can redeem the voucher and the amounts get credited to their account.

Upon capturing the redemption request, and performing the race condition, we were able to redeem multiple points to our account.



• Remediation

Race Conditions occur due to a lack of locking threads and synchronisation. When dealing with threads, applications should look at one thread at a time instead of dealing with multiple threads at the same time.

Business Logic Issues

Business logic issues are flaws that allow an application to perform unintended behaviour. These flaws enable attackers to manipulate functionality to achieve malicious goals. These vulnerabilities are very unique and require a lot of think-out-of-the-box techniques. For example access the payment success URL directly without paying anything, hence bypassing the payment functionality.

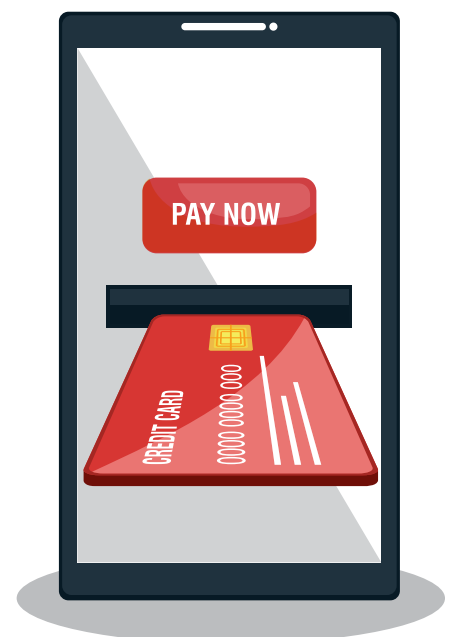
The business logic issues vary depending on functionality, but still, there are some common areas where you can try to perform business logic issues.

- **Common Functionalities/ Areas**

Payment Processing - As stated above, calling the success URL directly, hence bypassing the payment functionality.

Currency Change - Purchasing items in one currency (i.e. US dollars) but paying in another currency (i.e. Indian rupees).

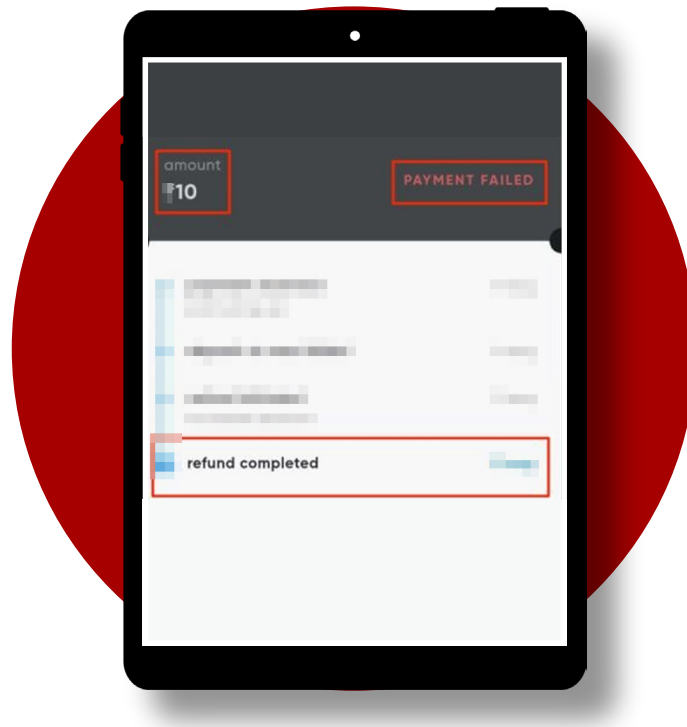
Dummy/test credit card validation - Verifying the valid credit card but changing the card to a dummy card at the time of purchase, making the purchase possible.



- **Scenario/ Real Scenario**

In penetration tests, RedHunt Labs found business logic issues for India's top e-commerce applications. Some of the POCs with explanations are below:

During a client's pentest, we successfully generated free reward points for our account by employing dummy credit cards. By adding these dummy cards and attempting a \$10 payment, the transaction failed due to their invalidity. Consequently, we received a \$10 refund, allowing us to generate unlimited reward points as desired.



- **Remediation**

Remediations are applied based on the vulnerability, as there are no common ways to secure the apps from business logic issues.

Misconfigured Payment Gateway

Most E-commerce applications use payment functionality where users can buy the product and pay for the same. The payment gateway provides payment functionality to the users, so they can easily process payments. This functionality requires security as it includes payments that contain cards, payment details, etc. Hence, a misconfigured payment gateway could lead to huge financial losses and fraud.

• Common Functionalities/ Areas

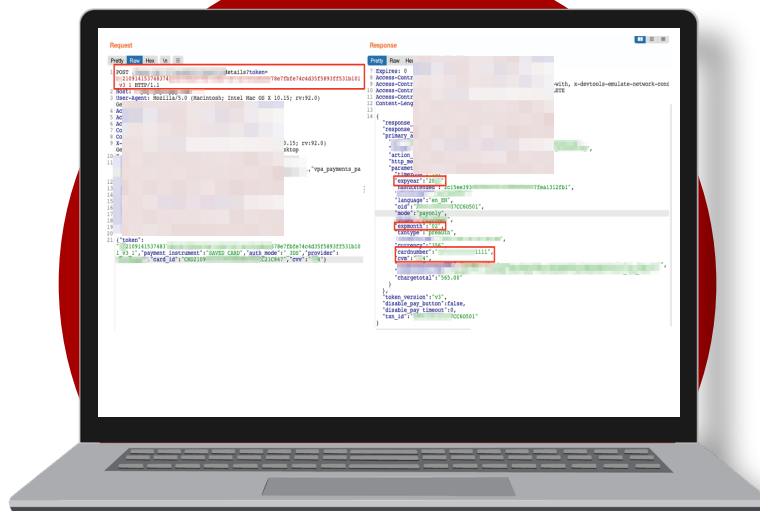
Payment Card Information Storage - Card information is stored as clear text in the database or leaked in the response, as it affects PCI DSS compliance.

Price Manipulation - Manipulating price at checkout time. Covered separately in detail below.

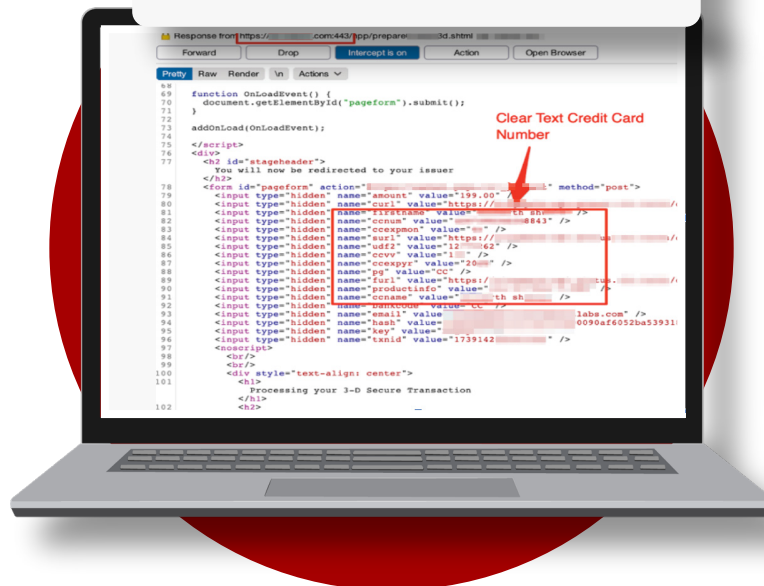
Quantity Manipulation - Increasing the quantity at the time of checkout.

• Scenario/ Real Scenario

In our client's pentest, we found that the card information was leaking clearly in response. This affects PCI DSS compliance, which clearly states that card information should never be disclosed anywhere in the



Another instance where RedHunt Labs found clear text credit card information being leaked



• Remediation

The payment gateway should be configured in such a way that it only takes product ID from the request, and all validations should be performed by the backend, or ensure that payment requests have integrity checks and card information is masked or removed in the response. The application should follow PCI DSS compliance as well.

Price Manipulation

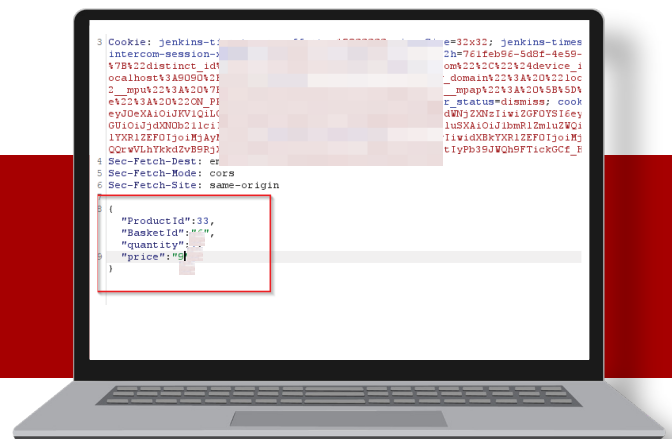
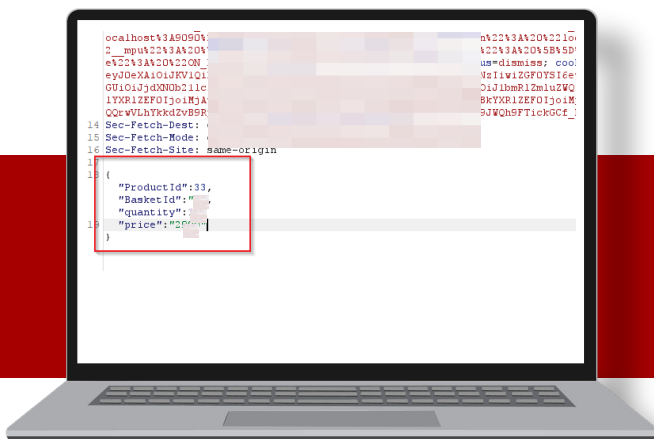
Price Manipulation is a common technique used by many attackers to purchase the product at a minimum price from the actual price. In this technique, attackers focus on finding the price in the request sent from the client to the server, during the process, attackers try to manipulate the price to a minimum price for ex if a product price is \$30, they can manipulate the price to \$1, hence purchasing at a minimum price which makes a huge amount of financial loss for the company.

• Common Functionalities/ Areas

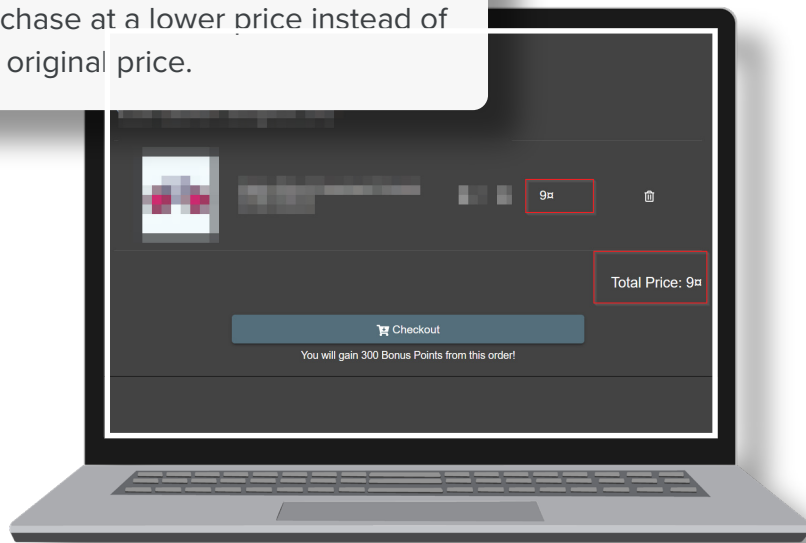
- Add product to cart
- Checkout

• Scenario/ Real Scenario

This is the example taken from a vulnerable app. In this scenario, the price amount was sent inside the request, so we tried to manipulate the price to lower the price. As we can see, the product price was \$2999 but we changed it to \$9.



As a result, we were able to purchase at a lower price instead of the original price.



“ Negative price - In this scenario, the attacker can inject a negative amount of price, and upon checking out, the user will get a refund instead of paying anything.

“ Usually, the application passes the product prices in a hashed format for integrity checks. If the hashing mechanism is poorly implemented, then an attacker might be able to modify the product price to a lower amount and regenerate the valid hash in the payment request, hence bypassing the hash mechanism.

• Remediation

The price should not be user-controlled, either web applications should always take prices from the database for the particular product or implement integrity checks to prevent tampering with parameters that affect the price.

Shopping cart

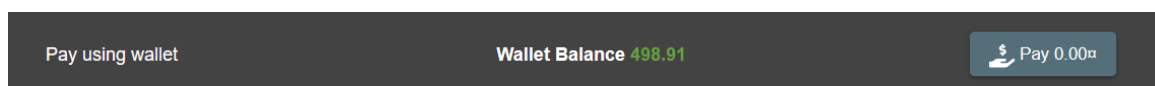
This is not a vulnerability but a feature where multiple vulnerabilities can be tested such as Authorization issues, race conditions, and negative quantity and exploiting these vulnerabilities could lead to huge financial issues. We will see some scenarios where we can try out some things and how we can exploit them.

• Common Functionalities/ Areas

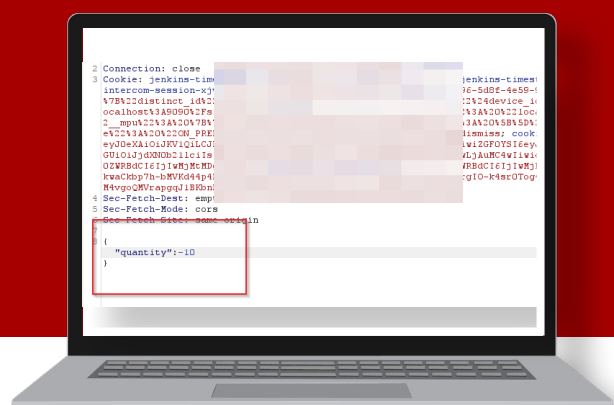
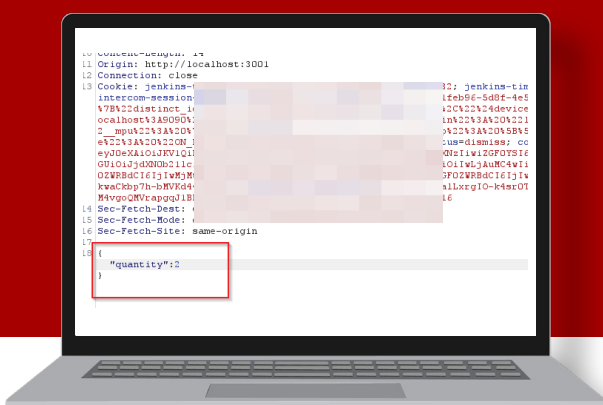
- Add product to cart

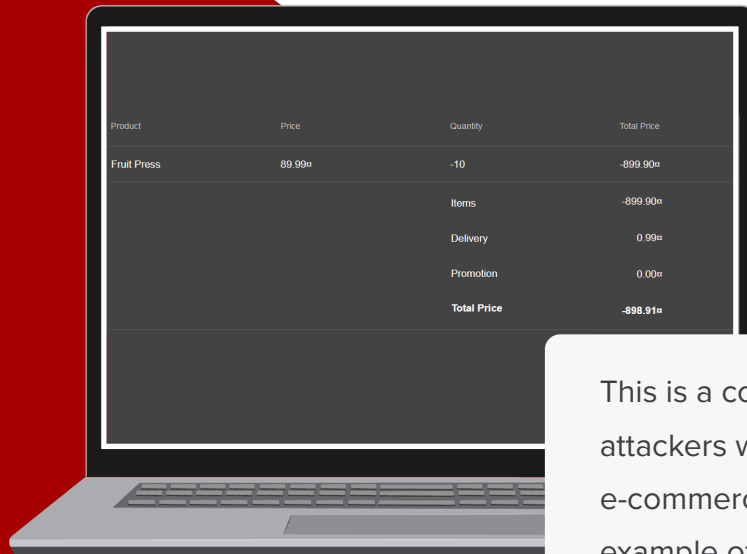
• Scenario/ Real Scenario

This is the example from the vulnerable OWASP juice shop application, where we can see there was a wallet balance of

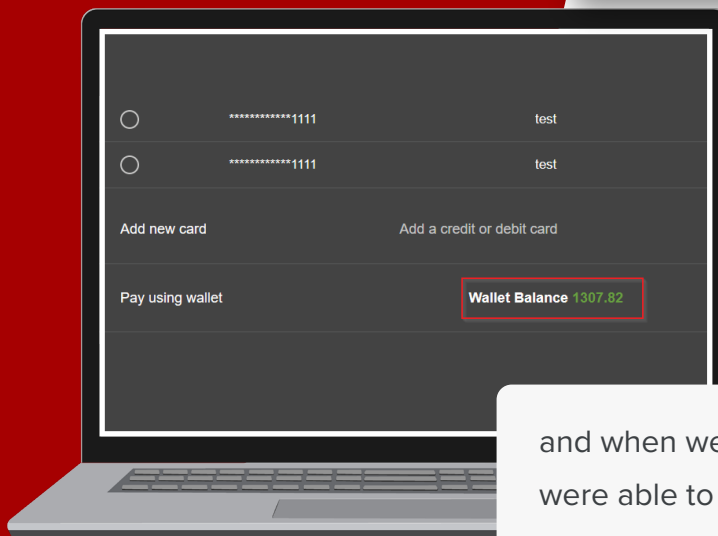


So we tried to add more quantity, but we modified the original quantity 2 to -10 which makes the -899 price.





This is a common bug that all attackers would give a try in any e-commerce application. This is an example of a vulnerable application and does not mean it will not be identified in real applications. Such issues are identified in real applications too.



and when we completed the payment we were able to increase the wallet balance by 899, hence total amount in the wallet increased by 1307

• Remediation

Input validation must be in place. If it takes an integer, it should only take a positive integer.



Bonus - Advanced Exploitation Techniques for Payment Gateway Integration

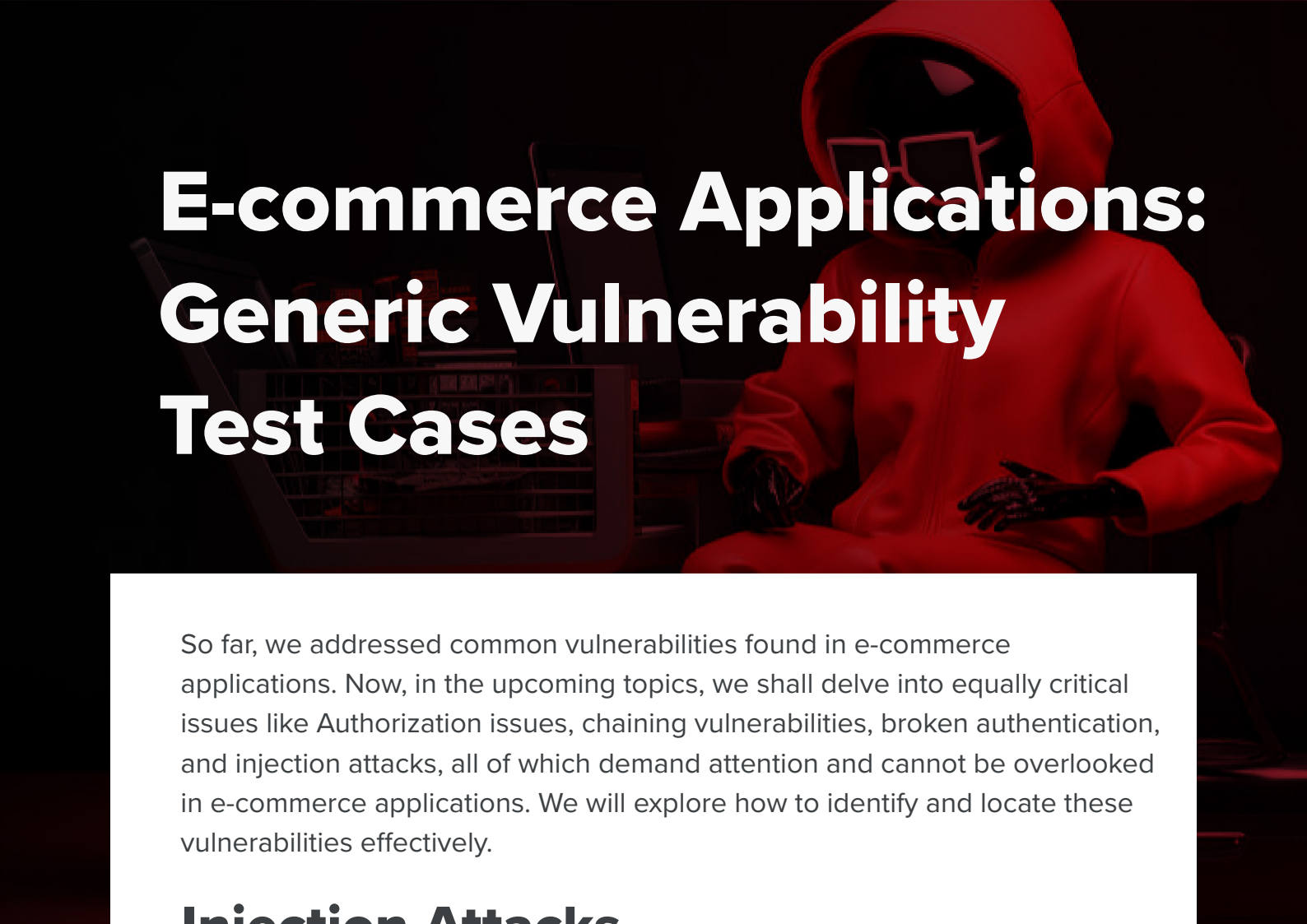
- **Bypassing Integrity**

Payment gateway supports integrity checks like hashing or encryption between payment gateways and e-commerce applications to prevent price manipulation by tempering certain parameters such as product price, product ids, quantity, currency etc. Such mechanisms could be missing or implemented poorly by e-commerce applications which allow an attacker to manipulate product prices and purchase products by paying the bare minimum amount.

- **Leakage Scenarios of Secrets Used by Payment Gateway Integration**

If salt used for hashing or encryption key used for encryption are somehow leaked to an internal or external attacker through sources such as hard-coded in client-side code (i.e. JS files), public GitHub repositories, environment variables exposed via vulnerabilities like SSRF or config files exposed publicly due to server misconfigurations. An attacker might be able to leverage leaked salt or encryption key to establish valid integrity checks between both parties (payment gateway and e-commerce app).





E-commerce Applications: Generic Vulnerability Test Cases

So far, we addressed common vulnerabilities found in e-commerce applications. Now, in the upcoming topics, we shall delve into equally critical issues like Authorization issues, chaining vulnerabilities, broken authentication, and injection attacks, all of which demand attention and cannot be overlooked in e-commerce applications. We will explore how to identify and locate these vulnerabilities effectively.

Injection Attacks

Injection attacks are attacks where malicious payload is injected that exfiltrates application server data to the attacker or which can be used to get access to the application server. Injection attacks are not bound to SQL Injection only, they include LDAP Injection, OS Command Injection, NoSQL injections, and Template injections. But for the sake of this part, we will focus only on SQL Injections, otherwise, it will be very lengthy. As we stated earlier, e-commerce applications include databases to store sensitive information such as credit card details, passwords, and emails. Hence these apps are prime targets for attackers because by using injection attacks such as SQL injection, attackers can easily exfiltrate the data from those databases.

- **Common Functionalities/ Areas**

There are no common areas to look for this vulnerability but instead, there are parameters where we can inject single or double quotes and check if there is any SQL-related error or 500 Internal Server Error. These two are promising indicators that there could be potential SQL Injection.



• Scenario/ Real Scenario

RedHunt Labs performed comprehensive penetration testing for applications and found lots of SQL Injections on large e-commerce applications. Some of the POCs with explanations are below.

Figure 1 shows that all e-commerce applications need databases where they can store information such as usernames, passwords, credit cards, product related information. Now, it's very common to see the input fields directly used inside SQL Statements where it allows the malicious user to exploit SQL Injection. Below, is an example of an e-commerce application where time-based SQL Injection was found.



• Remediation

The best mitigation for this issue is Parameterized Queries however if it is not possible then it is recommended to use Input validation at each step. For example, if a parameter is supposed to take integer values from user input, then the code must validate it with int function, then if any user injects any string, it will not process further inside the code.

Broken Authentication

Authentication at each step is very crucial otherwise it will lead the malicious actor to either exploit the flaw to achieve financial gain or to compromise another user account.

• Common Functionalities/ Areas

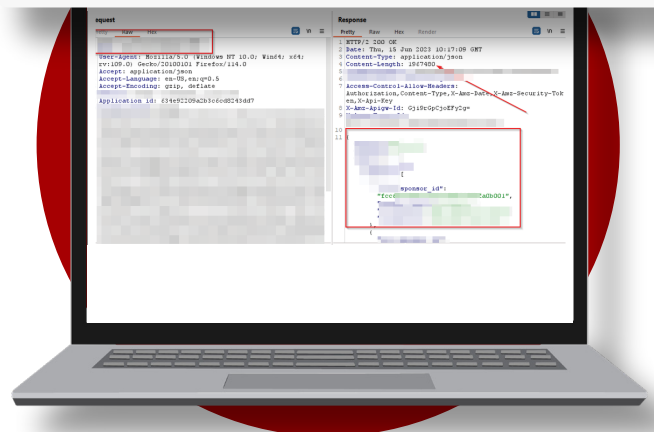
- Login/Registration/Forgot Password Pages
- 2FAs
- APIs

• Scenario/ Real Scenario

In this PoC below, you can see how RedHunt Labs was able to access an API endpoint without using any Authentication Token or Session ID which revealed information related to the Aadhar Card of users.



On another e-commerce application, we retrieved a very sensitive sponsor_id of each account leading to taking over any account.



• Remediation

To remediate the issue, one needs to implement authentication and validate on the server side. In this above-mentioned case, the authentication should be implemented on each and every API.



Authorization Issues

Authorization issues are those issues through which you can access or perform sensitive actions on the resource that you are not supposed to. For example, you could access other user details or you could change other user profile details. The vulnerability exploitation is the same in E-Commerce applications as well, however, there are more things you could check when finding authorization vulnerabilities such as, accessing other users' transaction receipts, accessing the credit or debit card details, etc.

• Common Areas to Look

There are numerous functionalities where we can look for auth issues. Some common areas where we can specifically check can be found below:

- Profile Details
- Order History
- Favorites
- Address
- Order Management
- Order Cancellation
- Cart
- Many more

Each functionality mentioned above must contain one primary key with which you could change it with another user account primary key. For example,

Transaction ID - Order history might contain transaction ID, change it with another user account transaction ID to get other user account transaction details

Receipt ID - To get other user receipts details

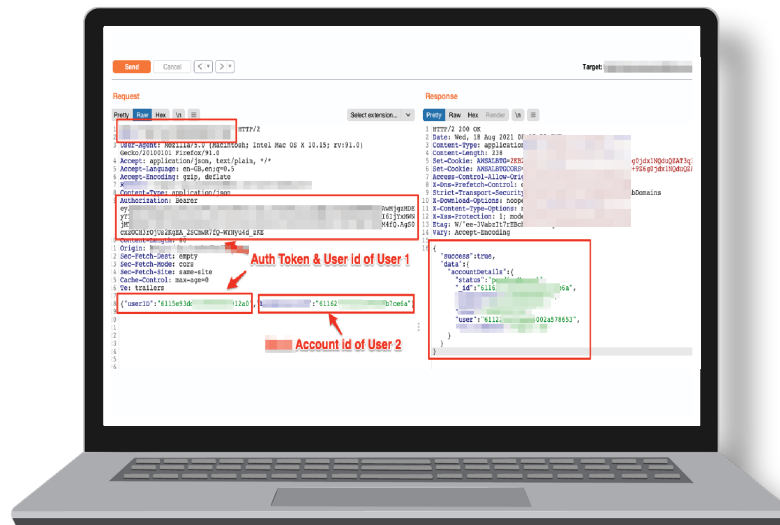
Profile ID - To change any user account details by using profile/user ID

Cart ID - To edit or delete the cart by using Cart ID



• Scenario/ Real Scenario

RedHunt Labs found a lot of Auth issues, below is one of the examples where we were able to retrieve bank account details saved in the application by providing the bank account id of another user.



• Remediation

To remediate the issue, the application must validate the session ID with the primary key ID. It should validate the user with the account before processing any requests.



Cross-Site Scripting

Cross-site scripting or XSS vulnerabilities are very common in many applications, these vulnerabilities occur when any user input is reflected in the page response as HTML. By doing so, the malicious actor could inject a script to modify the content of an HTML page or could try to steal other users' cookies or session IDs.

There are 3 types of XSS

- **Reflected XSS**

- **Stored XSS**

- **Dom-Based XSS**

The impact for all these is the same, the only difference is the mass exploitation of the vulnerability. For example, Reflected and DOM-based XSS are limited to one user at a time, but Stored XSS could be executed on multiple users at the same time.

XSS POCs are there but just after this line, POC is missing. So this line can be removed, and the same line anyways repeated under Scenario/ Real Scenario.

- **Common Areas to Look**

To exploit this vulnerability, you need to find the parameter which is reflected as it is in the HTML page.

- **Scenario/ Real Scenario**

RedHunt Labs found numerous XSS vulnerabilities in e-commerce applications, some examples can be found in the below screenshots.





• Remediation

It is recommended to use output encoding, functions such as HTML Entities along with Input validation.

Cross-site Request Forgery

CSRF are crucial in e-commerce applications as there might be instances where the attacker could not only change user account login details but also it could give the possibility of changing the user location details. For example, the attacker could change the user's default address to the attacker's address, what happens is that the user will end up placing an order to the attacker's address as the delivery address.

• Common Areas to Look

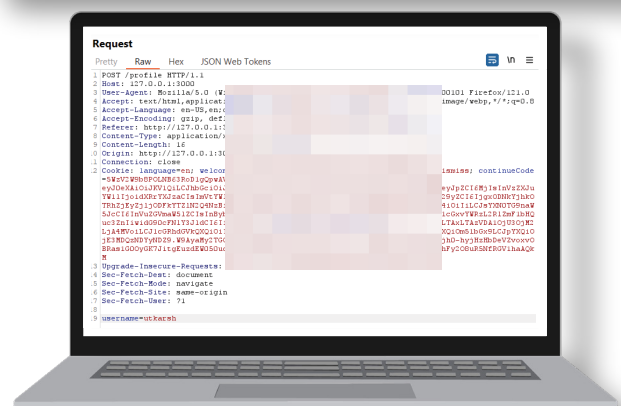
- Address Form
- Cart Changes
- Profile details

• Scenario/ Real Scenario

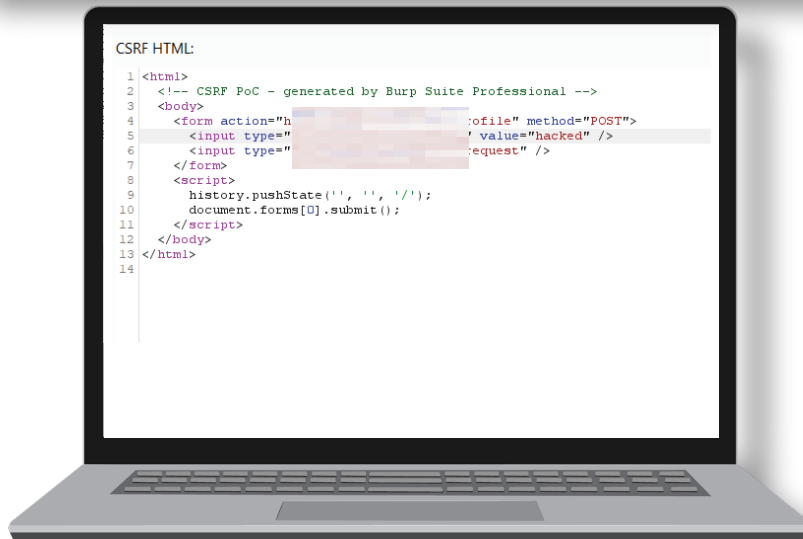
This was the basic exploitation of CSRF taken from OWASP juice shop where we changed the username of the user account from 'utkarsh' to 'hacked'.



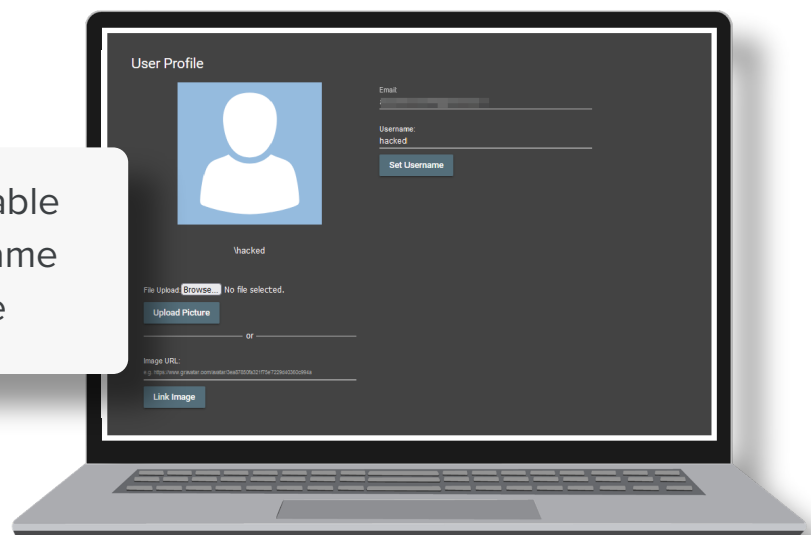
The actual request for changing the username is shown in the figure below.



This is the CSRF POC built by Burp Suite where we modified the value of the username parameter to 'hacked',



As a result, we were able to change the user name as shown in the figure



• Remediation

It is best to use the anti-csrf token inside the HTTP Post Body or as a header. The token must be validated at the server end.

Conclusion

“

In conclusion, penetration testing is a critical component of e-commerce security, enabling organizations to identify and mitigate vulnerabilities, protect sensitive data, and maintain customer trust. By adopting a proactive approach to security testing and implementing remediation strategies, organizations can strengthen the security posture of their e-commerce applications and defend against evolving threats and sophisticated attack techniques. Continuous monitoring, assessment, and improvement are essential for maintaining resilience and readiness in the face of emerging cyber threats and ensuring the long-term success and sustainability of e-commerce businesses.



E-commerce Security, Reimagined: An Ultimate Guide on E-commerce Penetration Testing

About RedHunt Labs

RedHunt Labs, a leading Attack Surface Management (ASM) company, accredited by CREST for Penetration Testing. We empower CISOs and security leaders by offering continuous monitoring solutions to track and assess their organization's true Attack Surface. By proactively addressing security risks and vulnerabilities, we ensure you discover your exposures and potential threats before attackers do.

For more information about RedHunt Labs and its CREST-accredited penetration testing services, please visit <https://redhuntlabs.com/> or write to us at contact@redhuntlabs.com

Get In Touch ↗

Schedule A Meeting ↗