



PENTESTING REPORT ON

www.ginandjuice.shop

LockBit Team



About Our Target

Created in 2022 by the man Distiller's World
has called "the evil genius of gin"

Gin & Juice Shop is open 24/7 to satisfy all of your
web vulnerability scanner evaluation needs.



GIN & JUICE SHOP

PRODUCTS

BLOG

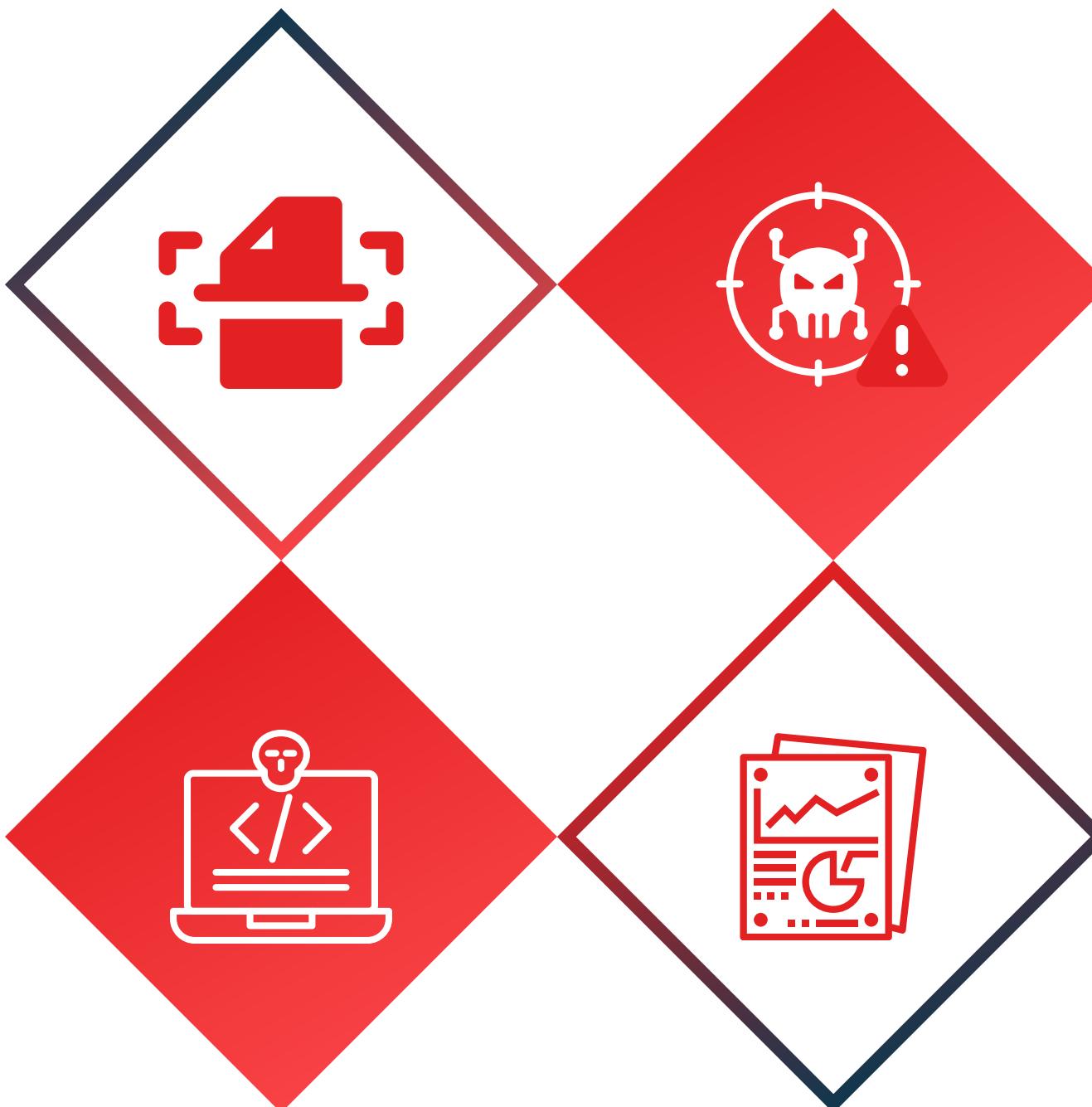
OUR STORY



Pentesting Time Line

Reconnaissance

We have made active and passive reconnaissance as a first important thing before starting to scan the target for identify the vulnerabilities



Exploitation

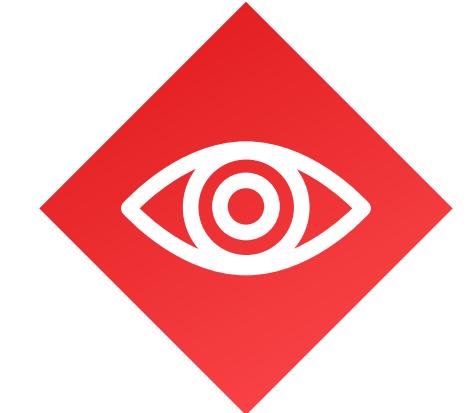
We have been attempting to capitalize on the vulnerabilities discovered during the previous phase.

Vulnerability Assessment

We have made a careful analysis of the target system to identify potential points of exploitation. That have been done by some automated tools and other manual methodologies

Reporting

After finding and exploit the finding vulnerabilities. It's time for the final stage which is reporting every thing.

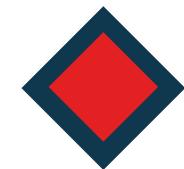


Site Preview



Accessed Pages

- + <https://ginandjuice.shop/>
- + <https://ginandjuice.shop/catalog>
- + <https://ginandjuice.shop/blog>
- + <https://ginandjuice.shop/about>
- + <https://ginandjuice.shop/login>
- + <https://ginandjuice.shop/catalog/cart>
- + <https://ginandjuice.shop/blog/post?postId=1-6>
- + <https://ginandjuice.shop/catalog/product?productId=1-18>



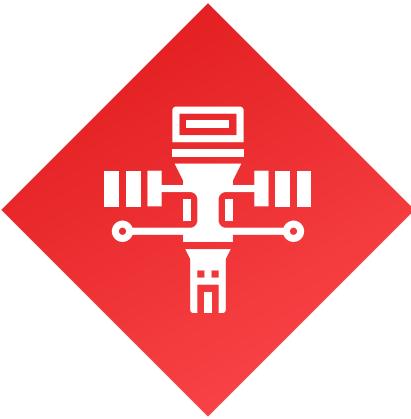
Pages that takes inputs

- <https://ginandjuice.shop/blog>
 - <https://ginandjuice.shop/login>
 - <https://ginandjuice.shop/catalog>
- Noted** that every page takes an email to subscribe and gives a coupon
9In&JUICE5HOP



Other Notes

- You can view the stock of each item in every location (london, paris, milan)
- Login Page provides you a username and password :
 - username: carlos
 - password: hunter2



Passive Reconnaissance

- ◆ **Hosting company** : Amazon - EU West (Ireland) datacenter
 - ◆ **Location** : Ireland
 - ◆ **Domain Owner** : Godaddy
 - ◆ **IPv4 address** : 34.246.169.176 – 34.246.169.140
 - ◆ **DNS admin** : awsdns-hostmaster@amazon.com
 - ◆ **Reverse DNS** : ec2-34-246-169-176.eu-west-1.compute.amazonaws.com
 - ◆ **Open Ports** : 80 (HTTP), 443 (TCP)
- ◆ **CNAME**
 - ns-1000.awsdns-61.ne
 - ns-110.awsdns-13.com
 - ns-1496.awsdns-59.org
 - ns-1543.awsdns-00.co.uk

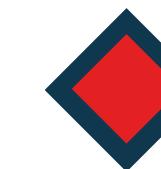


Active Reconnaissance



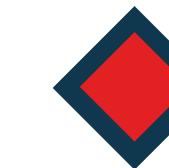
Dirb Results

```
+ https://ginandjuice.shop/subscribe (CODE:405|SIZE:20)
+ https://ginandjuice.shop/my-account (CODE:302|SIZE:0)
+ https://ginandjuice.shop/logout (CODE:302|SIZE:0)
+ https://ginandjuice.shop/Login (CODE:200|SIZE:7442)
+ https://ginandjuice.shop/login (CODE:200|SIZE:7451)
+ https://ginandjuice.shop/favicon.ico (CODE:200|SIZE:15406)
+ https://ginandjuice.shop/catalog (CODE:200|SIZE:16798)
+ https://ginandjuice.shop/Blog (CODE:200|SIZE:10905)
+ https://ginandjuice.shop/blog (CODE:200|SIZE:10923)
+ https://ginandjuice.shop/analytics (CODE:200|SIZE:0)
+ https://ginandjuice.shop/ADMIN (CODE:403|SIZE:15)
+ https://ginandjuice.shop/Admin (CODE:403|SIZE:15)
+ https://ginandjuice.shop/admin (CODE:403|SIZE:15)
```



Nikto Results

```
- Nikto v2.5.0/
+ Target Host: ginandjuice.shop
+ Target Port: 80
+ GET /: The anti-clickjacking X-Frame-Options header is not present.
+ GET /: The X-Content-Type-Options header is not set.
```



Nmap Scan

```
# nmap -Pn -p- -sS -sV -sC -O -oN namp.test -A --script vuln ginandjuice.shop
```

Nmap scan report for ginandjuice.shop (34.249.203.140)
Host is up (0.066s latency).

Other addresses for ginandjuice.shop (not scanned): 34.246.169.176
rDNS record for 34.249.203.140: ec2-34-249-203-140.eu-west-
1.compute.amazonaws.com

Not shown: 65533 filtered tcp ports (no-response)

| PORT | STATE | SERVICE | VERSION |
|--------|-------|---------|------------|
| 80/tcp | open | http | awselb/2.0 |

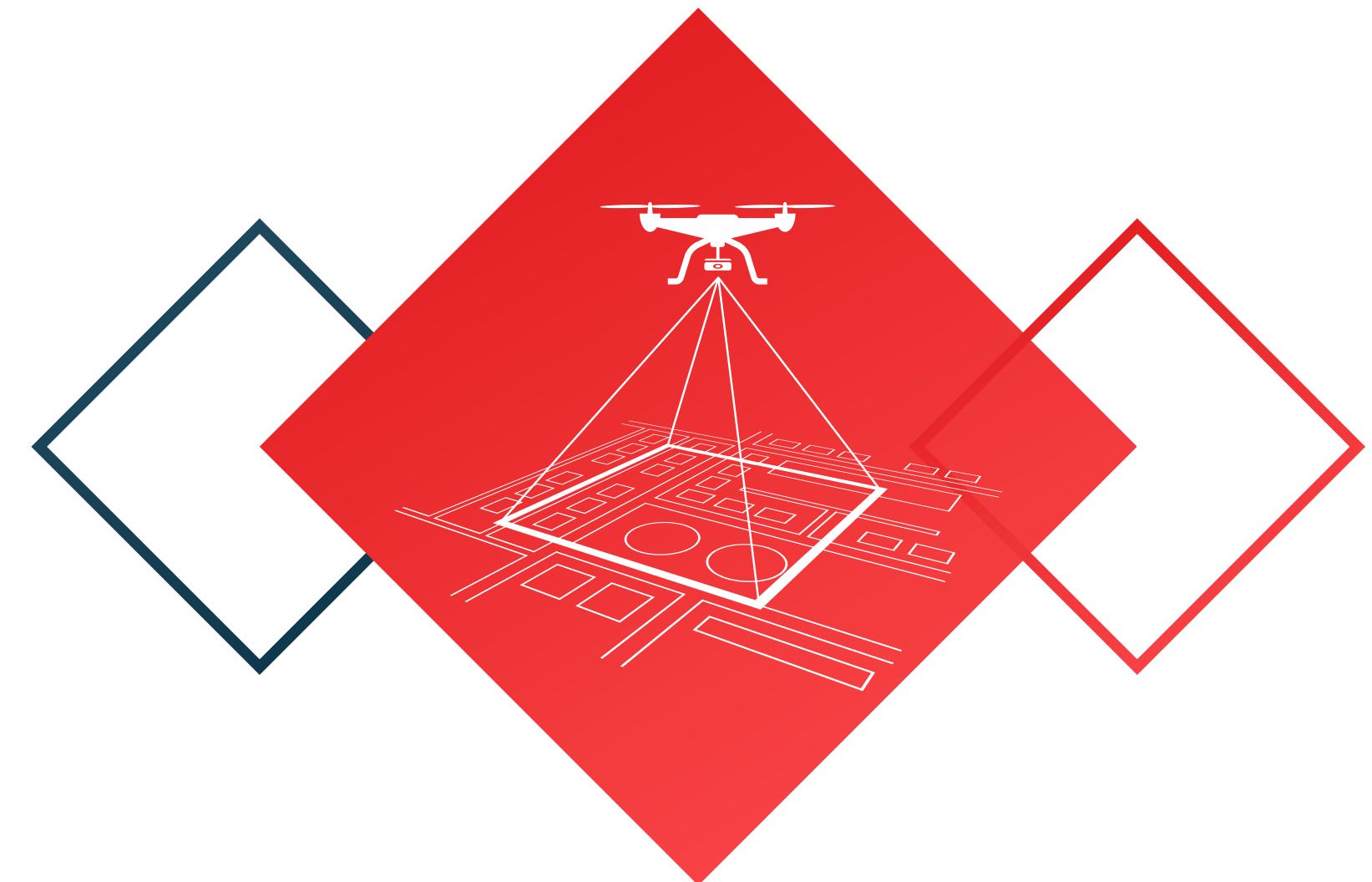
vulnerabilities Discovered

◆ Tools Used

- **Nmap** (for port and service scanning)
- **Nikto** (for web vulnerability scanning)
- **Burp Suite**
- **Arjun** (for parameters discovering)
- **Sqlmap** (for SQL injunction scanning)
- **Others as applicable** (custom scripts, etc.)

◆ Overview of Vulnerabilities

- **JavaScript Secrets Exposure**
- **SQL Injection**
- **Base64-encoded data in parameter**
- **DOM Based XSS (Document.Write())**





JavaScript Secrets Exposure

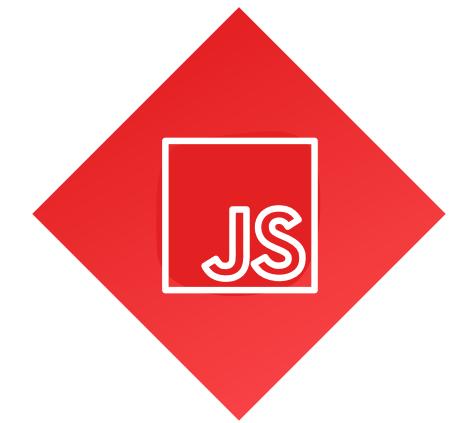
How We Found It

Running this command used to scan multiple JavaScript files listed in the file js.txt using SecretFinder, a tool designed to detect sensitive data, such as API keys, tokens, and passwords, within JavaScript files.

Findings

1. URL: <https://ginandjuice.shop/resources/footer/js/scamme.js>
 - password="gld.ngTmited" in plain text
2. URL: <https://ginandjuice.shop/resources/js/react-dom.development.js>
 - password="true" in plain text
3. Multiple Twilio account keys were exposed within the JavaScript files:
 - twilio_account_sid, twilio_app_sid, twilio_api_sid

```
(fergani@fergani)~]$ cat js.txt | while read url; do python3 SecretFinder/SecretFinder.py -i $url -o cli; done
+ ] URL: https://ginandjuice.shop/resources/footer/js
+ ] URL: https://ginandjuice.shop/resources/footer/js/scamme.js
+ ] URL: https://ginandjuice.shop/resources/js
+ ] URL: https://ginandjuice.shop/resources/js/angular_1-7-7.js
possible_Creds → password"=g||d.ngTrim&&
+ ] URL: https://ginandjuice.shop/resources/js/react-dom.development.js
amazon_aws_url2 → s3-animations/
amazon_aws_url2 → s3-transitions-20090320/
authorization_basic → basicStateReducer
authorization_api → API because
authorization_api → API hooks
twilio_account_sid → accumulateOrCreateContinuousQueued
twilio_account_sid → accumulateEnterLeaveTwoPhaseListen
twilio_account_sid → accumulateEnterLeaveListenersForEv
twilio_account_sid → actInternalMemoizedUnmaskedChildCo
twilio_account_sid → actInternalMemoizedMaskedChildCont
twilio_account_sid → actInternalMemoizedMergedChildCont
twilio_account_sid → acyErrorBoundariesThatAlreadyFaile
twilio_app_sid → apturePhaseSelectiveHydrationWitho
twilio_app_sid → apshotBeforeUpdateWithoutDidUpdate
possible_Creds → password: true,
+ ] URL: https://ginandjuice.shop/resources/js/react_development.js
```



JavaScript Secrets Exposure



Security Implications

Exposed secrets such as API keys and passwords can be used by attackers to:

- **Gain Unauthorized Access** : Attackers can exploit exposed credentials to access internal services
- **Data Theft or Modification** : Sensitive customer data or website functionality can be altered or stolen
- **Reputation Damage** : Exposing sensitive data can cause harm to the company's reputation, leading to loss of customer trust and potential legal repercussions.



How To Protect

- **Never Store Secrets in JavaScript Files**
 - Keep sensitive information like API keys or tokens out of front-end code.
- **Move Secrets to Server-Side**
 - Always store sensitive data on the server and interact with it via server-side logic.
- **API Gateway with Server-Side Authentication**
 - Use server-side authentication for APIs. API keys or tokens should never be exposed in the browser.



SQL Injection – Discovery



How We Found It

- Using **Arjun** to identify all hidden parameters in the URL of the pages.
- Using **sqlmap** on these parameters to discover any injunction we can exploit.
- Using **sqlmap** to raise the impact and reach the database which has sensitive information



Findings

1. An infected parameter in the catalog page (**category**)
2. Getting the databases names, columns names and access them
 - INFORMATION_SCHEMA, PUBLIC (Databases)
 - USERS, TRACKING, PRODUCTS .etc (Tables)
 - carlos (username), hunter2 (password)

```
inoskai@kaliOs: ~
File Actions Edit View Help
needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[17:38:38] [INFO] target URL appears to have 8 columns in query
[17:38:38] [WARNING] applying generic concatenation (CONCAT)
[17:38:39] [INFO] URI parameter '#1*' is 'Generic UNION query (NULL) - 1 to 10 columns' injectable
[17:38:39] [INFO] checking if the injection point on URI parameter '#1*' is a false positive
URI parameter '#1*' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
N
sqlmap identified the following injection point(s) with a total of 91 HTTP(s) requests:
Parameter: #1* (URI)
Type: UNION query
Title: Generic UNION query (NULL) - 8 columns
Payload: http://ginandjuice.shop/catalog?category=' UNION ALL SELECT NULL,NULL,CONCAT
(CONCAT('qqvxq','rflctEgPgNFPscvqmdeLtvWqskdyjsUSJpJtMzRS'), 'qjxqq'),NULL,NULL,NULL,NULL,
NULL-- muAH
[17:38:53] [INFO] testing MySQL
[17:38:54] [WARNING] the back-end DBMS is not MySQL
[17:38:54] [INFO] testing Oracle
[17:38:54] [INFO] confirming Oracle
```

The screenshot shows a web browser window with a terminal-like interface at the top displaying sqlmap command-line output. Below the terminal is a screenshot of a website titled 'GIN&JUICESHOP'. The website has a dark blue header with 'PRODUCTS', 'BLOG', 'OUR STORY', and other navigation links. A search bar says 'Search products'. Below the header is a 'PRODUCTS' section with a sub-section for 'Search results'. A search term 'qqvxqrflctEgPgNFPscvqmdeLtvWqskdyjsUSJpJtMzRSqjxqq' is entered into the search bar. Below the search bar are several small buttons for 'All', 'Accessories', 'Accompaniments', 'Books', 'Gin', and 'Juice'. At the bottom of the page, there is a link 'Continue shopping'.



SQL Injection



Security Implications

- **Data Exposure:** An attacker can retrieve sensitive data from the database, including user information, product details, and potentially confidential business data.
- **Database Manipulation:** Depending on the privileges of the database user, an attacker could alter, delete, or insert new records.
- **Application Compromise:** Further exploitation could lead to complete control over the application or the underlying server.



How To Protect

- **Input Validation:** Use parameterized queries (prepared statements) to avoid SQL injection.
- **Web Application Firewall:** Use a WAF to filter and block malicious traffic.
- **Error Handling:** Avoid exposing database errors to users.
- **Security Testing:** Regular vulnerability assessments.

Base64-Encoded Data

How We Found It

- By using burp suite we found that there is parameter encoded in Based64.

The screenshot shows a browser's developer tools Network tab. The Request section shows a GET request to /admin. The Response section shows a 403 Forbidden status with a JSON response body. The response body contains a session token and other headers. The 'Selected text' panel shows the decoded JSON response:

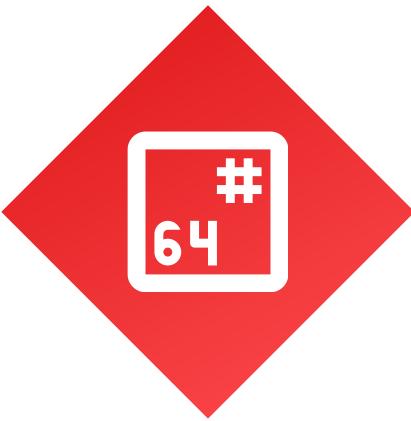
```

{
  "type": "class",
  "value": "nx86SfdCV4wIGXlz"
}

```

Security Implications

- Easily Decoded**
 - Base64 is not a secure method of encoding; it can be easily decoded with simple tools or even through the browser's developer console.
- Sensitive Data Exposure**
 - If confidential information (e.g., session tokens, authentication credentials) is encoded in Base64 and used in URLs or parameters, it can be exposed in server logs, browser history, and referer headers, making it easy for attackers to access.



Base64-Encoded Data – Mitigation



How To Protect

- **Avoid Using Base64 for Sensitive Data:**
 - Base64 encoding is not encryption—it is easily reversible. Avoid using Base64 encoding for sensitive data (like user IDs, session tokens, or any confidential information) in URLs or parameters.
- **Use Proper Encryption Instead of Base64:**
 - For any sensitive data that needs to be transmitted, use proper encryption techniques (e.g., AES or RSA). Ensure that data is encrypted securely on the server side before being sent to the client or transmitted in URLs or parameters.
- **Implement Server-Side Validation and Access Control:**
 - Always validate and authorize any incoming Base64-encoded data or parameters on the server side. Don't rely on the client to handle validation—check on the back end to ensure the integrity and legitimacy of the data before processing.

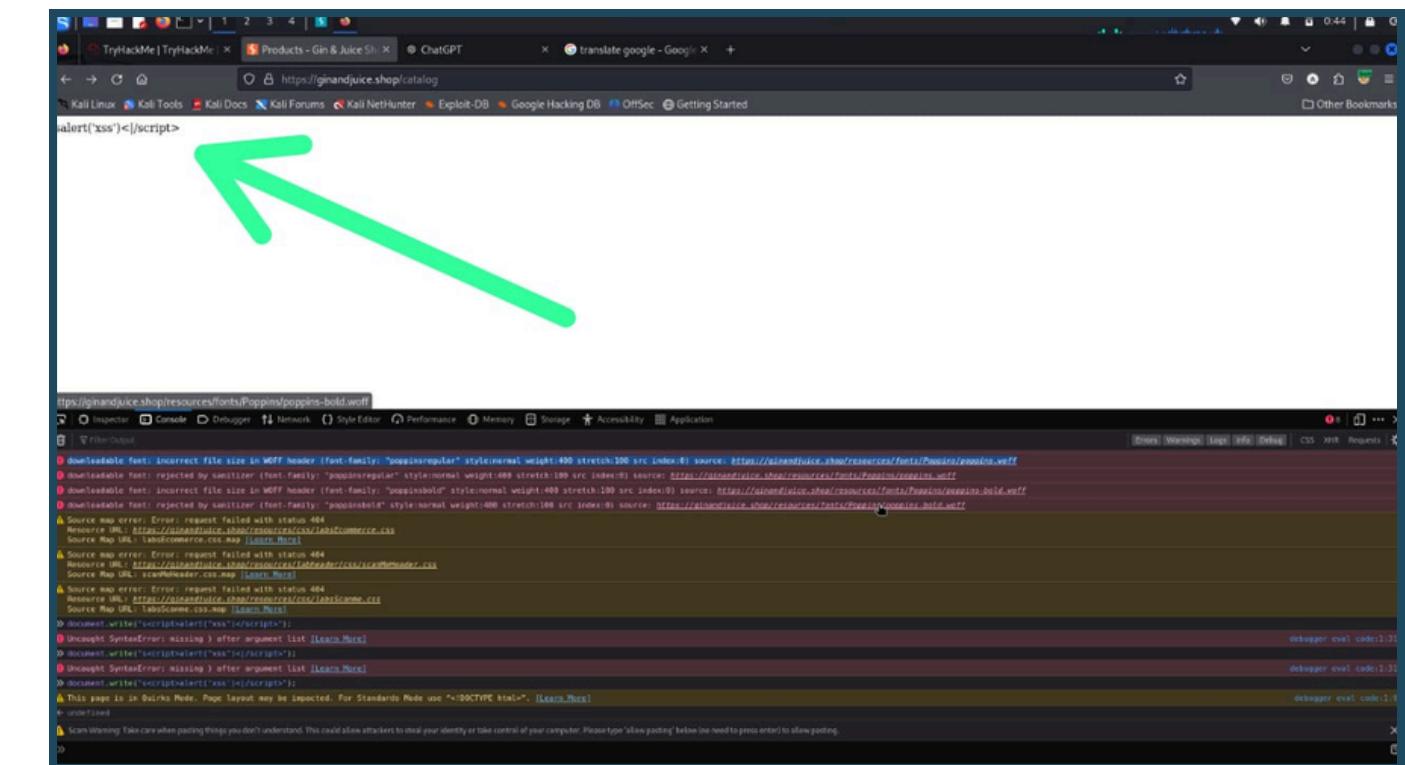


DOM Based XSS



How We Found It

- **Observation:** When searching for a term, the input value was reflected in the URL but was sanitized, preventing direct injection of a malicious payload via the search field.
- **Exploit Method:**
 - I switched to the browser's Developer Tools, specifically the **Console**.
 - I attempted to inject an XSS payload directly using the **Document Object Model (DOM)** instead of relying on traditional input fields.
- `document.write("<script>alert('XSS');</script>");`





DOM Based XSS

Security Implications

- **Unauthorized Script Execution:**
 - Attackers can inject malicious scripts into a webpage, which can execute in the user's browser.
- **Phishing and Social Engineering:**
 - An attacker could manipulate the webpage content using XSS to display fake login prompts or other deceptive messages, tricking users into giving away sensitive information.

How To Protect

- **Sanitize User Input:**
 - Ensure any data taken from the user (e.g., input fields, URLs) is sanitized and validated before being processed or rendered in the DOM.
- **Use Safe DOM Manipulation Methods:**
 - Avoid using methods like `document.write()`, `innerHTML`, or `eval()` to insert user input directly into the DOM.

Report Writing

Our penetration testing report provides a comprehensive summary of identified vulnerabilities, their impact, and actionable remediation steps, ensuring both technical teams and business stakeholders understand the security risks.

[Gin and Juice Shop Report Link](#)



Jr Penetration Tester Path

We have successfully make it just in 12 days and finished the path. Together after our collaboration and amazing efforts we made a lot and we will continue after our discussion to make more. This path covers the core technical skills that will allow us to succeed as a junior penetration tester.



CERTIFICATE OF COMPLETION

this is to acknowledge that

Philopater shenouda sedkiy shenouda

has **successfully** completed the

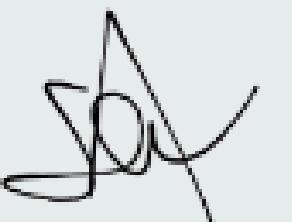
**Jr Penetration Tester
Learning Path**

24th October 2024

Time to complete: 29 hours 5 minutes



Ben Spring
CEO



Ashu Savani
CTO



**Try
Hack
Me**



Ahmed Elsaygh

ginandjuice.shop Pentest

Active Reconnaissance

- Interacting with the target to get information about it
- Using Burp suite to analyse the sent and received packets

Searching For Vulnerabilities in packets

- By using Burp suite

Report Writing

Jr. Pentesting Path

Introduction to Pentesting

Burp Suite

Metasploit

- All completed
- All Tasks Solved

Antony Essac

ginandjuice.shop Pentest

Active Reconnaissance

- Interacting with the target to get information about it
- Using Dirb tool to get the hierarchy of the target

Identify an XSS vulnerability

Raising the impact of the SQL injection vulnerability

- Make it further to get to the USERS table and got the user names and passwords

Jr. Pentesting Path

Introduction to Web Hacking





Belal Rabea

ginandjuice.shop Pentest

Active scanning

- Interacting with the target to get information about it
- Using Burp suite to analyse the sent and received packets

Identify SQL injection vulnerability

- By using SQL map to search for it in all pages

Identify JS Secrets Exposure vulnerability

- By using script securityFinder

Jr. Pentesting Path

Vulnerability Research

- Completed
- Complete the Vulnerability Capstone

A photograph of a young man with short dark hair, smiling at the camera. He is standing in a library aisle with tall bookshelves on either side. He is wearing a black t-shirt with a logo that includes a globe and the letters "ACPC".

Osama Yousef

ginandjuice.shop Pentest

Passive Reconnaissance

- Searching for public information about the target
- Review the website as a normal user to identify its functionality

Identify Base64-encoded data in parameter

Identify SQL injection in an infected parameter

Jr. Pentesting Path

Introduction to cybersecurity

- Completed

Privilege Escalation

- Not completed yet

A professional portrait of a young man with dark hair and glasses, wearing a dark suit, white shirt, and blue tie. He is smiling and has his hands clasped in front of him.

Philopater Shenoda

ginandjuice.shop Pentest

Passive Reconnaissance

- Searching for public information about the target
- Searching for subdomains

Active scanning

- nmap

Searching for XSS vulnerabilities

- Trying to find any working XSS payload manually.

Jr. Pentesting Path

Network Security

- Completed
- All tasks have been solved
- Complete Net Sec Challenge



THANK YOU

LockBit Team

