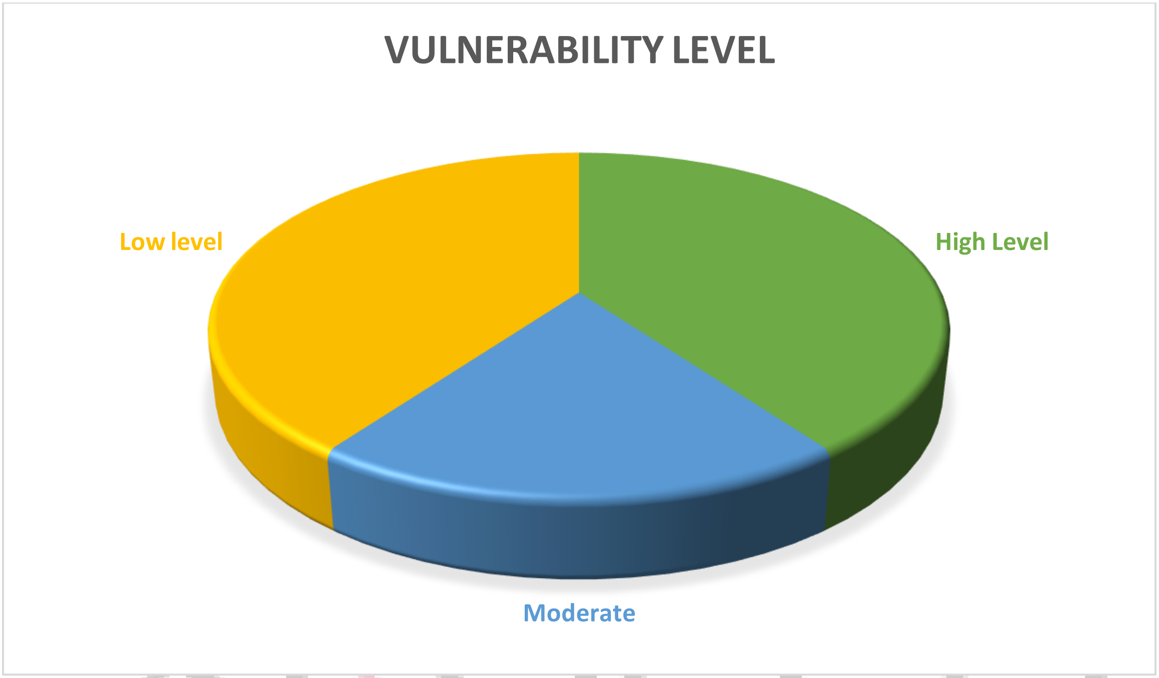
Vulnerability Assessment Report

Link: http://15.207.221.18:1002/dow88u170t/app



Vulnerability List

- HTML Injection
- Reflected XSS
- Insecure File Upload
- Insecure Password (no limit)
- Insecure GET POST PUT Request Type
- Unsecure Domain
- Insecure Username
- Denial of Service Attack
- Violation of Secure Web Design Principal

HTML Injection

Description

What is HTML Injection?

The essence of this type of injection attack is injecting HTML code through the vulnerable parts of the website. The Malicious user sends HTML code through any vulnerable field with a purpose to change the website's design or any information, that is displayed to the user.

In the result, the user may see the data, that was sent by the malicious user. Therefore, in general, HTML Injection is just the injection of markup language code to the document of the page.

Data, that is being sent during this type of injection attack may be very different. It can be a few HTML tags, that will just display the sent information. Also, it can be the whole fake form or page.

When this attack occurs, the browser usually interprets malicious user data as legit and displays it.

Changing a website's appearance is not the only risk, that this type of attack brings. It is quite similar to the XSS attack, where the malicious user steals other person's identities. Therefore stealing another person's identity may also happen during this injection attack.

Types of HTML Injection

This attack does not seem to be very difficult to understand or to perform, as HTML is considered as a quite simple language.

However, there are different ways to perform this type of attack.

We can also distinguish different types of this injection.

Firstly, different types may be sorted by the risks, that they bring.

As mentioned, this injection attack can be performed with two different purposes:

- To change the displayed website's appearance.

- To steal another person's identity.

Also, this injection attack can be performed through different parts of the website i.e data input fields and the website's link.

However, the main types are:

- Stored HTML Injection
- Reflected HTML Injection

#1) Stored HTML Injection:

The main difference between those two injection types is that stored injection attack occurs when malicious HTML code is saved in the web server and is being executed every time when the user calls an appropriate functionality.

However, in the reflected injection attack case, malicious HTML code is not being permanently stored on the webserver. Reflected Injection occurs when the website immediately responds to the malicious input.

#2) Reflected HTML Injection:

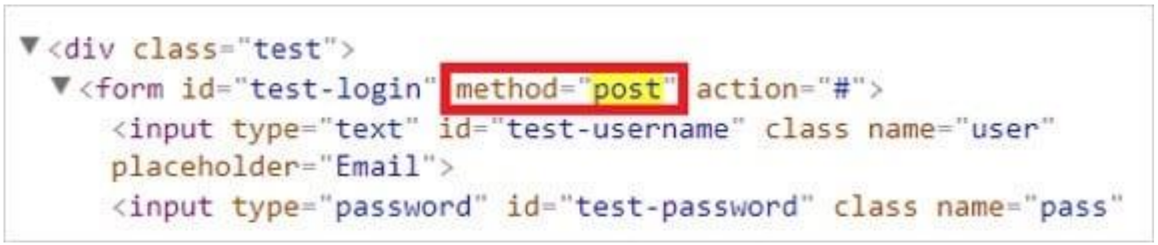
This can be again divided into more types:

- Reflected GET
- Reflected POST
- Reflected URL

Reflected Injection attack can be performed differently according to the HTTP methods i.e, GET and POST. I would remind, that with POST method data is being sent and with GET method data is being requested.

To know, which method is used for appropriate website's elements, we can check the source of the page.

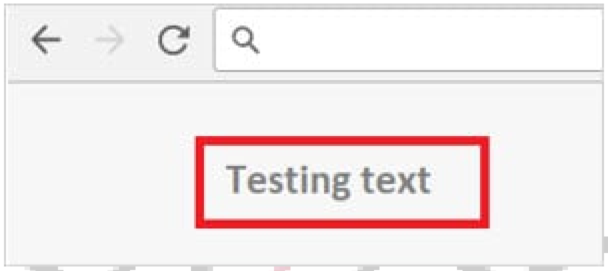
<u>For Example</u>, a tester can check the source code for the login form and find what method is being used for it. Then appropriate HTML Injection method can be selected accordingly.



```
▼<div class="test">
  ▼<form id="test-login" method="post" action="#">
     <input type="text" id="test-username" class name="user"
     placeholder="Email">
     <input type="password" id="test-password" class name="pass"
```

Reflected GET Injection occurs, when our input is being displayed (reflected) on the website. Suppose, we have a simple page with a search form, which is vulnerable to this attack. Then if we would type any HTML code, it will appear on our website and at the same time, it will be injected into the HTML document.

<u>For Example, we enter simple text with HTML tags:</u>

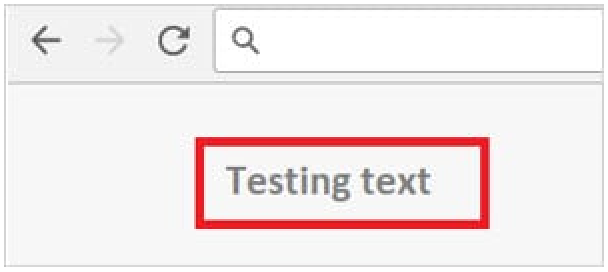


Reflected POST HTML Injection is a little bit more difficult. It occurs when a malicious HTML code is being sent instead of correct POST method parameters.

<u>For Example</u>, we have a login form, which is vulnerable to HTML attack. Data typed in the login form is being sent with POST method. Then, if we would type any HTML code instead of the correct parameters, then it will be sent with POST method and displayed on the website.

To perform Reflected POST HTML attack, it is recommended to use a special browser's plugin, that will fake the sent data. One of it is Mozilla Firefox plugin “Tamper Data”. The plugin takes over the sent data and allows the user to change it. Then changed data is being sent and displayed on the website.

For Example, if we use such a plugin then we would send the same HTML code *<h1>Testing test</h1>*, and it will also display the same as the previous example.



Reflected URL happens, when HTML code is being sent through the website URL, displayed in the website and at the same time injected to the website's HTML document

Impact of HTML Injection:

It can allow an attacker to modify the page.

To steal another person's identity.

The attacker discovers injection vulnerability and decides to use an HTML injection attack.

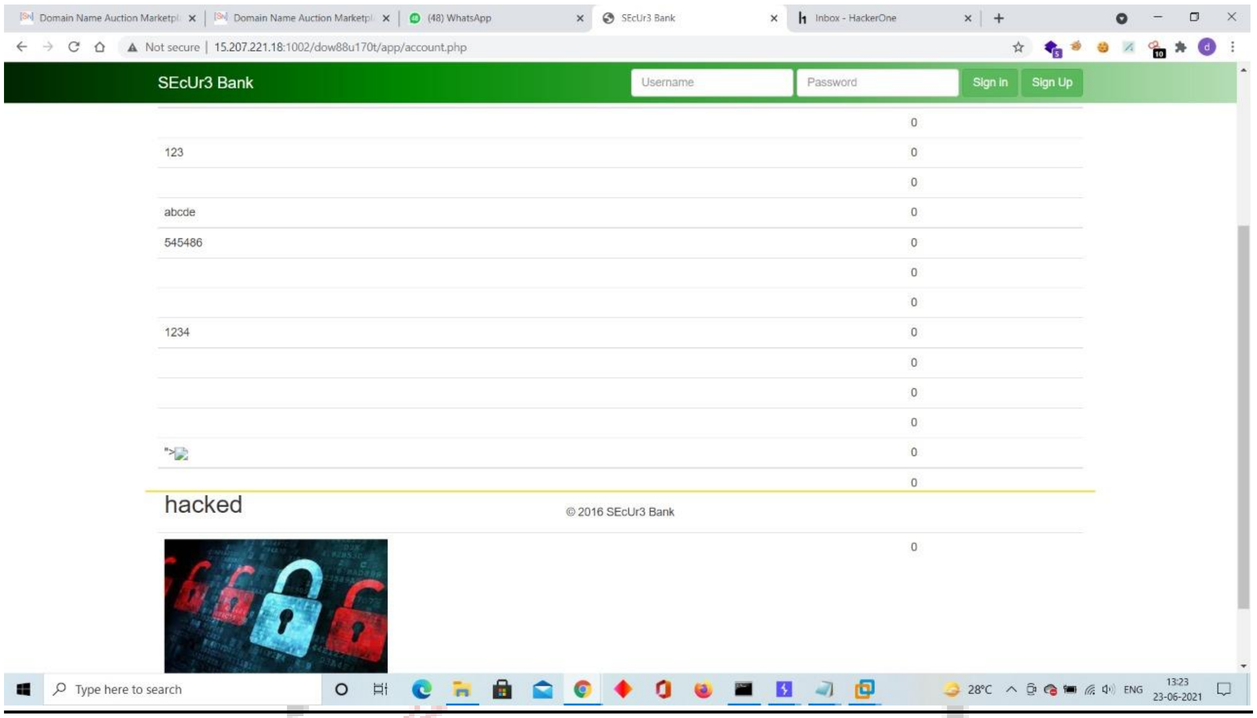
attacker's server.Attacker crafts malicious links, including his injected HTML content, and sends it to a user via email.

The user visits the page due to the page being located within a trusted domain.

The attacker's injected HTML is rendered and presented to the user asking for a username and password.

The user enters a username and password, which are both sent to the

Proof of Vulnerability



	0
123	0
	0
abcde	0
545486	0
	0
	0
1234	0
	0
	0
	0
">	0
	0
hacked	
	0

© 2016 SEcUr3 Bank

We Can Inject any link, Payload in the HTML Code

Prevention form HTML Injection

There are no doubts, that the main reason for this attack is the developer's inattention and lack of knowledge. This type of injection attack occurs when the input and output are not properly validated. Therefore the main rule to prevent HTML attack is appropriate data validation.

Every input should be checked if it contains any script code or any HTML code. Usually it is being checked, if the code contains any special script or HTML brackets – <script></script>, <html></html>.

There are many functions for checking if the code contains any special brackets. Selection of checking function depends on the programming language, that you are using.

It should be remembered that good security testing is also a part of prevention. I would like to pay attention, that as HTML Injection attack is very rare, there is less literature to learn about it and less scanner to select for automatic testing. However, this part of security testing really should not be missed, as you never know when it may happen.

Also, both the developer and tester should have good knowledge of how this attack is being performed. Good understanding of this attack process may help to prevent it.

Reference Link

https://www.softwaretestinghelp.com/html-injection-tutorial/#What_is_HTML_Injection

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/03-Testing_for_HTML_Injection#:~:text=HTML%20injection%20is%20a%20type,into%20a%20vulnerable%20web%20page.&text=An%20injection%20allows%20the%20attacker,HTML%20page%20to%20a%20victim.

Reflected XSS

Description

Reflected cross-site scripting (or XSS) arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

Suppose a website has a search function which receives the user-supplied search term in a URL parameter:

https://insecure-website.com/search?term=gift

The application echoes the supplied search term in the response to this URL:

<p>You searched for: gift</p>

Assuming the application doesn't perform any other processing of the data, an attacker can construct an attack like this:

https://insecure-website.com/search?term=<script>/*+Bad+stuff+here...+*/</script>

This URL results in the following response:

<p>You searched for: <script>/* Bad stuff here... */</script></p>

If another user of the application requests the attacker's URL, then the script supplied by the attacker will execute in the victim user's browser, in the context of their session with the application.

Impact of reflected XSS attacks

If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. Amongst other things, the attacker can:

Perform any action within the application that the user can perform.

View any information that the user is able to view.

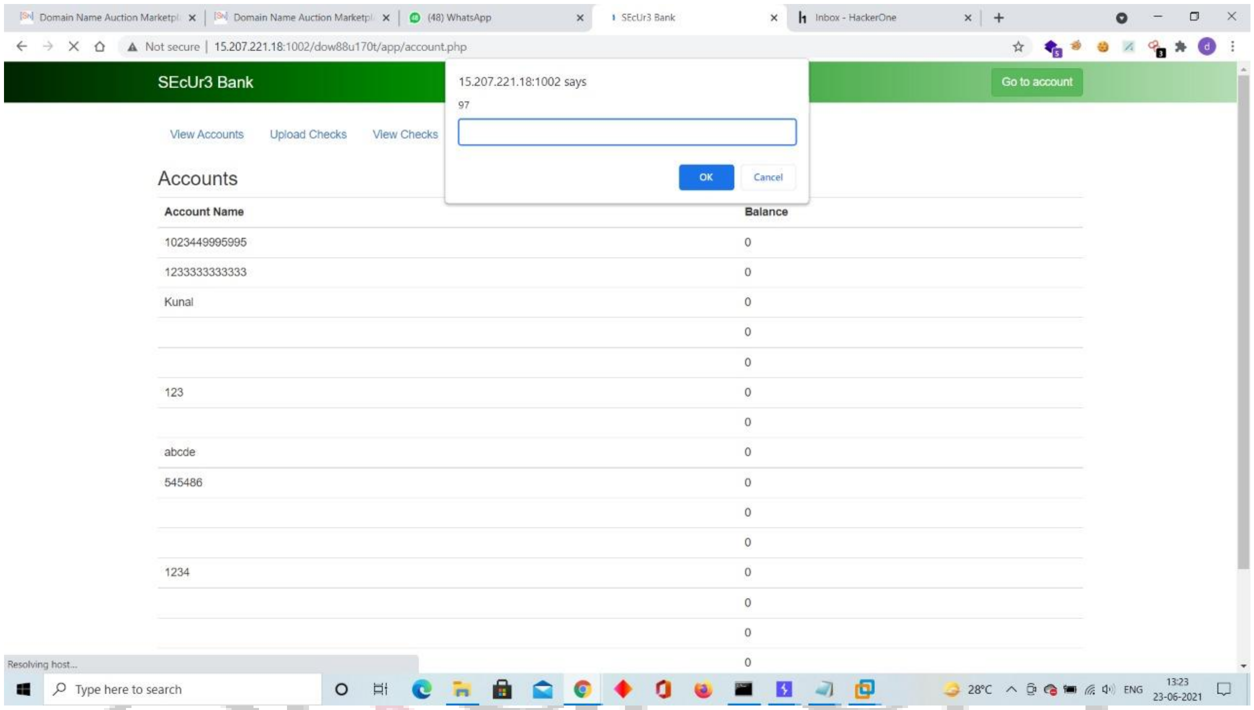
Modify any information that the user is able to modify.

Initiate interactions with other application users, including malicious attacks, that will appear to originate from the initial victim user.

There are various means by which an attacker might induce a victim user to make a request that they control, to deliver a reflected XSS attack. These include placing links on a website controlled by the attacker, or on another website that allows content to be generated, or by sending a link in an email, tweet or other message. The attack could be targeted directly against a known user, or could an indiscriminate attack against any users of the application:

The need for an external delivery mechanism for the attack means that the impact of reflected XSS is generally less severe than stored XSS, where a self-contained attack can be delivered within the vulnerable application itself.

Proof of Vulnerability



15.207.221.18:1002 says

97

Accounts

Account Name	Balance
1023449995995	0
1233333333333	0
Kunal	0
	0
	0
123	0
	0
abcde	0
545486	0
	0
	0
1234	0
	0
	0
	0

How to prevent XSS attacks

Preventing cross-site scripting is trivial in some cases but can be much harder depending on the complexity of the application and the ways it handles user-controllable data.

In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:

- **Filter input on arrival.** At the point where user input is received, filter as strictly as possible based on what is expected or valid input.
- **Encode data on output.** At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted

as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.

- **Use appropriate response headers.** To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.
- **Content Security Policy.** As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.

Reference Link

https://portswigger.net/web-security/cross-site-scripting

https://sushant747.gitbooks.io/total-oscp-guide/content/cross-site-scripting.html

Insecure File Upload Allowance

Description

File upload vulnerability is a common security issue found in web applications. Whenever the web server accepts a file without validating it or keeping any restriction, it is considered as an unrestricted file upload.

In many web servers, the vulnerability depends entirely on its purpose, allowing a remote attacker to upload a file with malicious content. This might end up in the execution of unrestricted code in the server. File upload vulnerability can be exploited in many ways, including the usage of specially crafted multipart form-data POST requests with particular filename or mime type.

The consequences include whole system acquisition, an overloaded file system or database, diverting attacks to backend systems, and simple defamation.

insufileuplod

Types Of File Upload Vulnerability

Local File Upload Vulnerability

Local file upload vulnerability allows an application user to upload or drop a malicious file directly into the website.

Given below is an example program that allows an attacker to upload a malicious file to an application:

```
   <?php
    $target_dir = "uploads/";    //specifies the directory where the file is going to be placed
    $target_file = $target_dir.basename($_FILES["fileToUpload"]["name"]);
    //specifies the path of the file to be uploaded
    $imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION)); //holds the file extension of the file
```

Given below is the code that shows that this vulnerability can be avoided by verifying the user license:

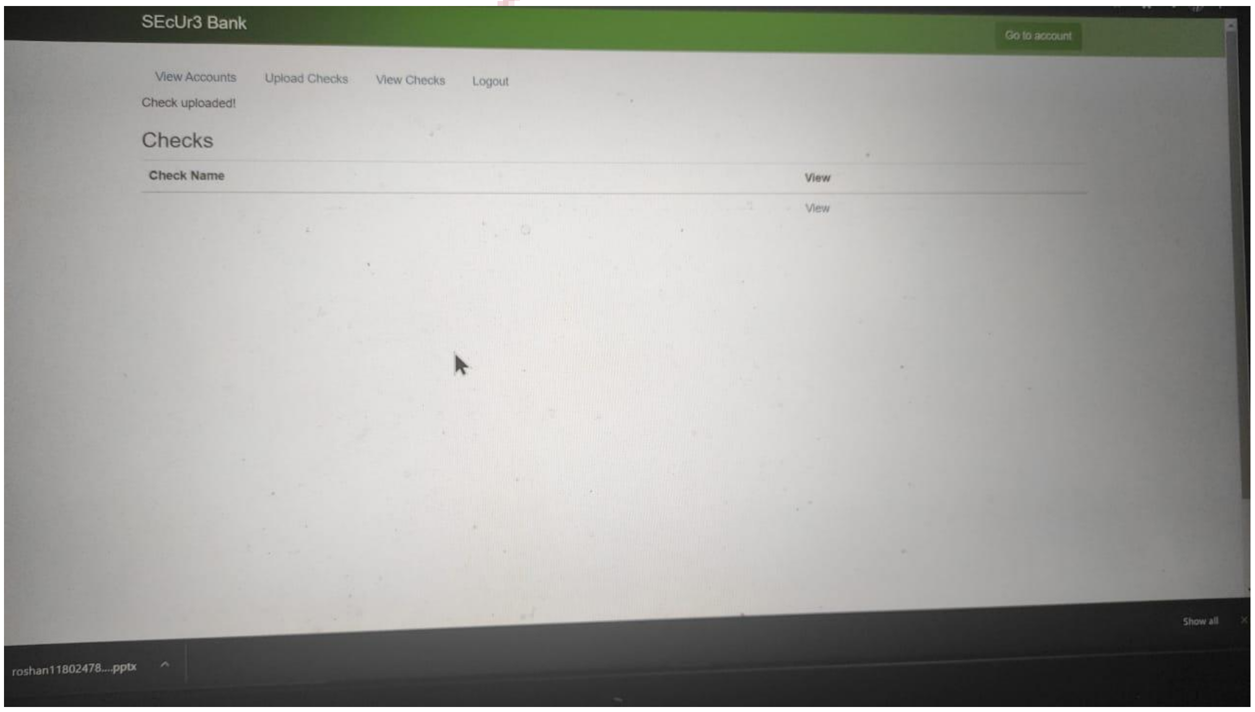
```
    if (!current_user_can('upload_files'))  //verify current user has the permission to upload a file
    die(__('You do not have permission to upload files.'));
    // Process file upload
```

This protection can be bypassed with some extensions (eg: double extensions) that are executable on the server but not listed. (Example files: “file.php5”, “file.shtml”,”wecome.html.fr”,“file.asa”, or “file.cer”)

Impact of Insecure File Upload Allowance

- akeover of the victim’s entire system through a server-side attack.
- Files are injected through the malicious paths. So, existing critical files can be overwritten as the .htaccess file can be embedded to run specific scripts.
- Inject phishing pages to discredit the web application.
- File uploads may expose critical internal information in error messages such as server-internal paths.

Proof of Vulnerability



Check uploaded!

Checks

Check Name	View
	View

I was able to upload ppt.

How To Prevent File Upload Vulnerability

A file upload vulnerability can be prevented by following the below mitigation techniques:

- Allow only certain file extension.
- Set maximum file size and name length.
- Allow only authorized users.
- Make sure the fetched file from the web is an expected one.
- Keep your website updated.
- Name the files randomly or use a hash instead of user input.
- Block uploads from bots and scripts using captcha.
- Never display the path of the uploaded file.

Reference link

https://beaglesecurity.com/blog/vulnerability/insecure-file-upload.html

https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

Insecure Password

Description

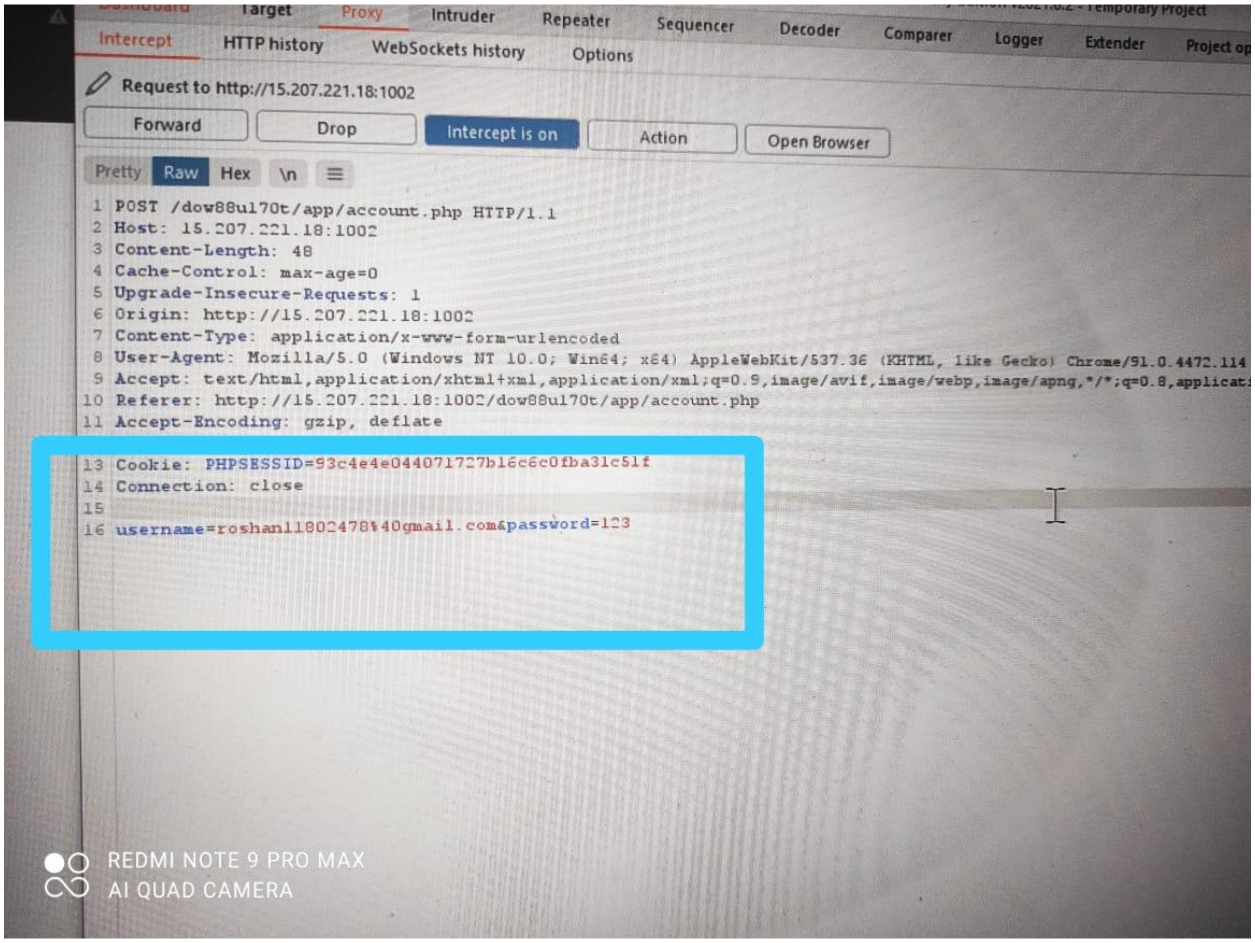
The product does not require that users should have strong passwords, which makes it easier for attackers to compromise user accounts.

Authentication mechanisms often rely on a memorized secret (also known as a password) to provide an assertion of identity for a user of a system. It is therefore important that this password be of sufficient complexity and impractical for an adversary to guess. The specific requirements around how complex a password needs to be depended on the type of system being protected. Selecting the correct password requirements and enforcing them through implementation are critical to the overall success of the authentication mechanism.

Impact of Insecure Password

Weak passwords can be guessable or attacker can bruteforce if the length of the password is very small, so try to use random strings with special characters. Though that can be hard to remember as a security point of view it’s quite secure

Proof of Vulnerability



Request to http://15.207.221.18:1002

```
POST /dow88ul70t/app/account.php HTTP/1.1
Host: 15.207.221.18:1002
Content-Length: 48
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://15.207.221.18:1002
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
Referer: http://15.207.221.18:1002/dow88ul70t/app/account.php
Accept-Encoding: gzip, deflate

Cookie: PHPSESSID=93c4e4e044071727b16c6c0fba31c51f
Connection: close

username=roshan11802478%40gmail.com&password=123
```

Check my account password.

Prevention

Password policies are a front line of defense. They are typically a set of rules intended to improve security by motivating or compelling users to create and maintain dependable, safe passwords. Password policies govern password lifecycle events, such as authentication, periodic resets, and expiration. Although some password policies are advisory and outline best practices for users, most sites require users to adhere to the policy using programmatic rules. User frustration can arise if users are required to spend time attempting to create passwords that meet unfamiliar criteria. Having a password policy can help mitigate user frustration by providing guidelines and certainty. The following are examples of password policies:

- Requiring longer passwords. Longer passwords and passphrases have been shown to substantially improve security. However, it's still essential to avoid longer passwords that have been previously compromised or regularly appear in cracking dictionaries.
- Do not use personal details. This password policy encourages users to create passwords with no link to the user's personal information. As explained earlier, most users build passwords using personal details, such as hobbies, nicknames, names of pets or family members, etc. If a hacker has access to personal details about a particular user (such as through social media), they will try password combinations using this information. At a minimum, passwords should be checked screened to make sure they don't include basic information like the user's name or login information.
- Use different passwords for different accounts. Password policies should require users to differentiate security from convenience and disallow users from using the same password for all of their accounts. Password sharing between users – even those who might work in the same department or who may use the same equipment – should have distinct passwords.
- Adopt passphrases as a standard. Some password policies require users to create a passphrase as opposed to a password. While passphrases serve the same purpose, they are usually harder to crack due to their length. An effective passphrase should include numbers and symbols as well as letters. Users may remember passphrases more easily than passwords.
- Discourage sharing. Password policies should specify that passwords are meant to be personal and should not be shared between users. Use the second factor. Another password policy is the adoption of two-factor authentication (2FA.) 2FA requires a user to present two pieces of evidence before they can log in, which is typically a password and a temporary code delivered to a cellphone, an email or other method.

(2) **Password screening**
One of the best ways to prevent dictionary attacks is to screen them against known lists of dictionary passwords and compromised passwords. Compromised password screens collect compromised data from the internet and Dark Web sources and then determines if the password a user is trying to create has been compromised. Password screening tools work by checking a partial hash of username and password at login, password setup, and reset. It can also be beneficial to consumer sites and e-commerce companies to detect and protect them from fraudsters who use previously compromised credentials.

Reference

https://www.acunetix.com/vulnerabilities/web/weak-password/#:~:text=A%20weak%20password%20is%20short,common%20variations%20on%20these%20themes.

Insecure GET POST PUT Request Types

Method Definitions

The set of common methods for HTTP/1.1 is defined below. Although this set can be expanded, additional methods cannot be assumed to share the same semantics for separately extended clients and servers.

The Host request-header field MUST accompany all HTTP/1.1 requests.

Safe and Idempotent Methods

Safe Methods

Implementors should be aware that the software represents the user in their interactions over the Internet, and should be careful to allow the user to be aware of any actions they might take which may have an unexpected significance to themselves or others.

In particular, the convention has been established that the GET and HEAD methods SHOULD NOT have the significance of taking an action other than retrieval. These methods ought to be considered "safe". This allows user agents to represent other methods, such as POST, PUT and DELETE, in a special way, so that the user is made aware of the fact that a possibly unsafe action is being requested.

Naturally, it is not possible to ensure that the server does not generate side-effects as a result of performing a GET request; in fact, some dynamic resources consider that a feature. The important distinction here is that the user did not request the side-effects, so therefore cannot be held accountable for them.

Impact

It can allow an attacker to modify the page.

To steal another person’s identity.

The attacker discovers injection vulnerability and decides to use an HTML injection attack.

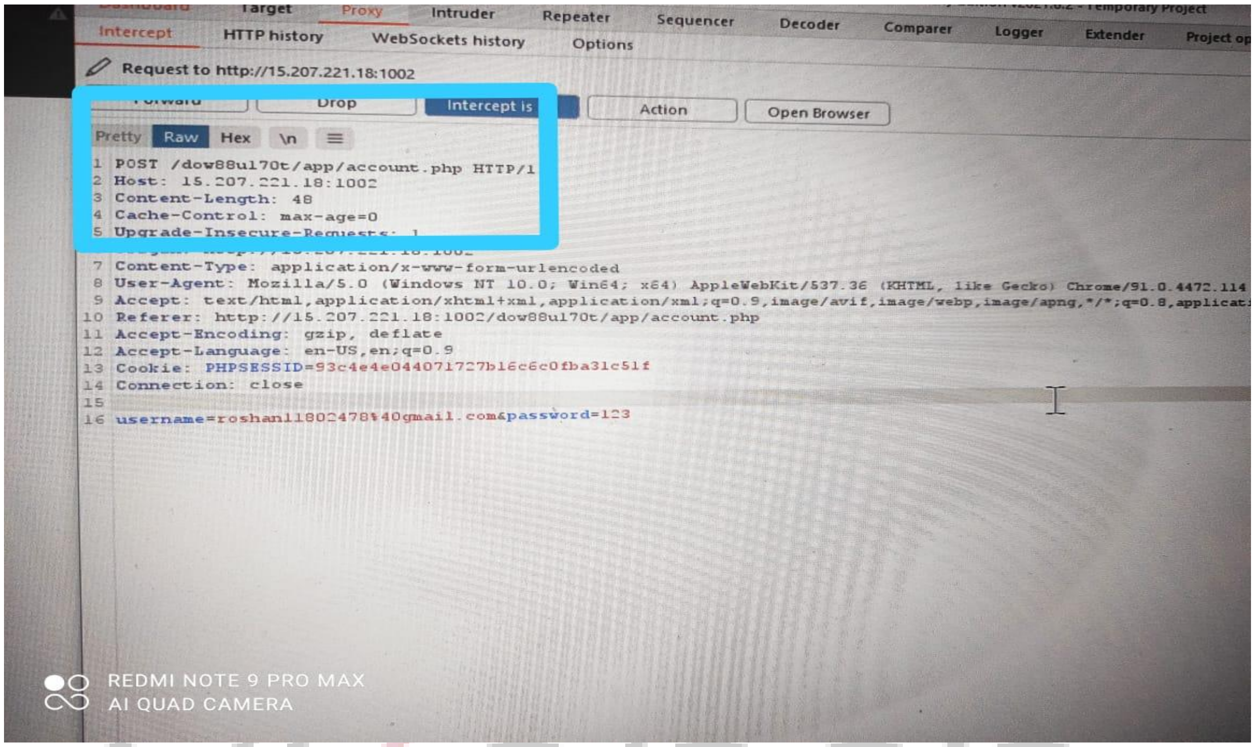
attacker’s server.Attacker crafts malicious links, including his injected HTML content, and sends it to a user via email.

The user visits the page due to the page being located within a trusted domain.

The attacker’s injected HTML is rendered and presented to the user asking for a username and password.

The user enters a username and password, which are both sent to the

Proof of Vulnerability



```
POST /dow88ul70t/app/account.php HTTP/1
Host: 15.207.221.18:1002
Content-Length: 48
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://15.207.221.18:1002
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
Referer: http://15.207.221.18:1002/dow88ul70t/app/account.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=93c4e4e044071727b16c6c0fba31c51f
Connection: close

username=roshan11802478%40gmail.com&password=123
```

We can see in the image how I can change GET to POST.

Prevention

Enable only HTTP methods on your web server which are necessary for your application to run. Use only GET and POST methods for all HTTP requests where possible.

If you need any insecure HTTP methods to be enabled on your server, make sure they are properly authorized and available only for specific resources. This way you'll prevent any malicious usage of those.

Reference

https://scanrepeat.com/web-security-knowledge-base/insecure-http-method

Username enumeration

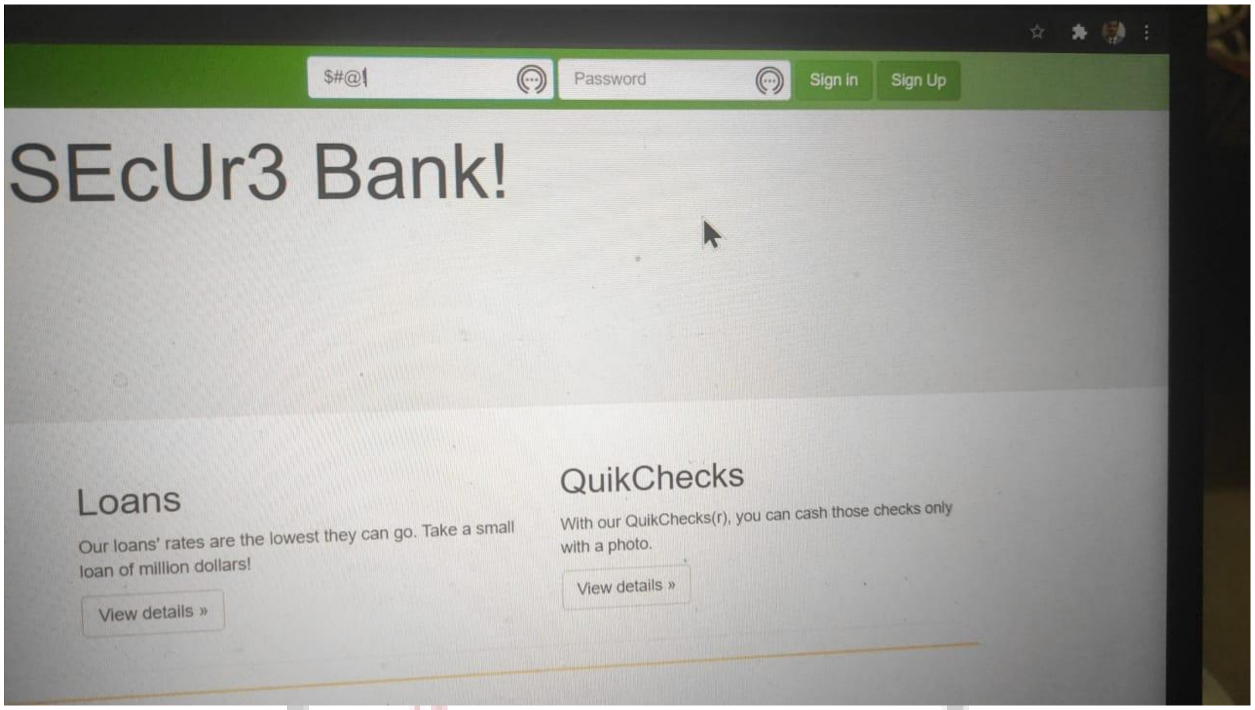
Username enumeration is when an attacker is able to observe changes in the website's behavior in order to identify whether a given username is valid.

Username enumeration typically occurs either on the login page, for example, when you enter a valid username but an incorrect password, or on registration forms when you enter a username that is already taken. This greatly reduces the time and effort required to brute-force a login because the attacker is able to quickly generate a shortlist of valid usernames.

While attempting to brute-force a login page, you should pay particular attention to any differences in:

- **Status codes**: During a brute-force attack, the returned HTTP status code is likely to be the same for the vast majority of guesses because most of them will be wrong. If a guess returns a different status code, this is a strong indication that the username was correct. It is best practice for websites to always return the same status code regardless of the outcome, but this practice is not always followed.
- **Error messages**: Sometimes the returned error message is different depending on whether both the username AND password are incorrect or only the password was incorrect. It is best practice for websites to use identical, generic messages in both cases, but small typing errors sometimes creep in. Just one character out of place makes the two messages distinct, even in cases where the character is not visible on the rendered page.
- **Response times**: If most of the requests were handled with a similar response time, any that deviate from this suggest that something different was happening behind the scenes. This is another indication that the guessed username might be correct. For example, a website might only check whether the password is correct if the username is valid. This extra step might cause a slight increase in the response time. This may be subtle, but an attacker can make this delay more obvious by entering an excessively long password that the website takes noticeably longer to handle.

Proof of vulnerability


$#@!

Password

Sign In

Sign Up

SEcUr3 Bank!

Loans

Our loans' rates are the lowest they can go. Take a small loan of million dollars!

View details »

QuikChecks

With our QuikChecks(r), you can cash those checks only with a photo.

View details »

Reference

https://portswigger.net/web-security/authentication/password-based

Unsecure Domain

Description

Occurs when an application contains content provided from a 3rd party resource that is delivered without any type of content scrub.

Environments Affected

- Web servers
- Application servers
- Client Machines

Risk Factors

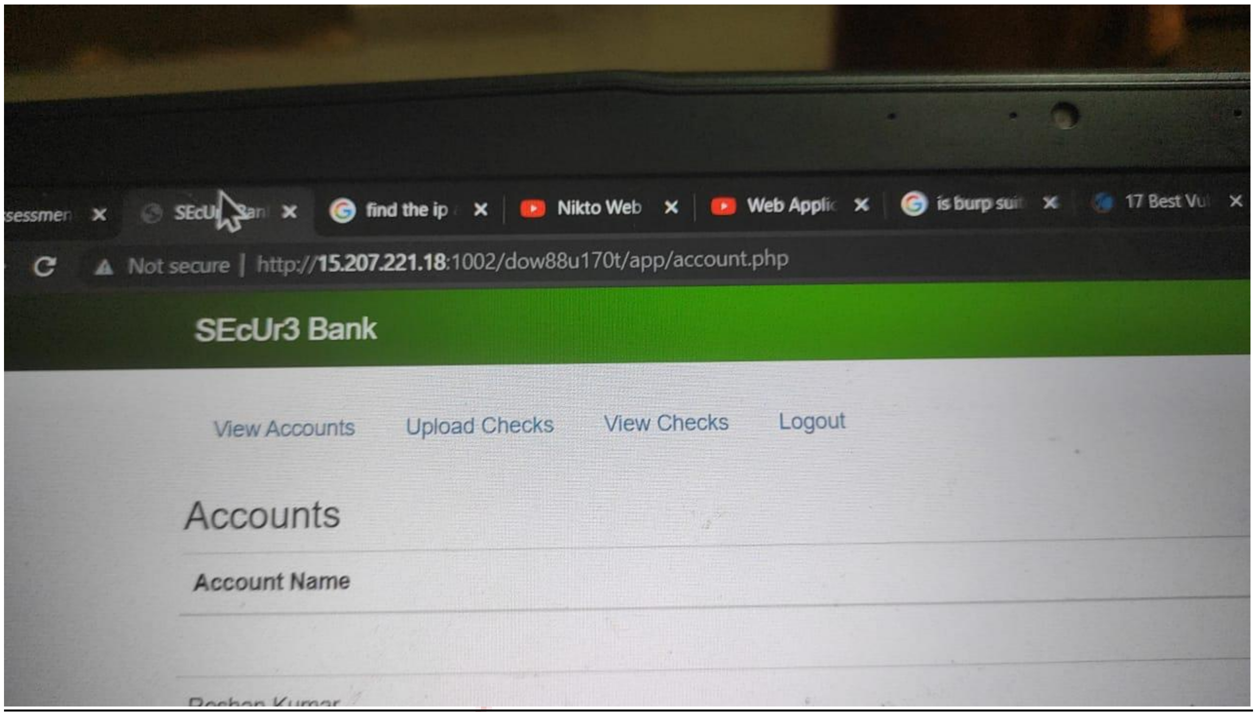
- Allowing hosted content from an untrusted server into a trusted application: affecting the server, server environment, and client machine.
- No confirmation of Third-Party Controls.

Examples

This following example is a common method to insert third party hosted content into a trusted an application. If the hosting site is vulnerable to attack, all content delivered to an application would be vulnerable malicious changes.

```
<iframe src="http://site.com/share/Action.swf" width="720" height="420"
       marginwidth="0" marginheight="0" scrolling="Auto" frameborder="0">
</iframe>
```

<u>Proof of Vulnerability</u>



We can see the is HTTP not HTTPS

Reference

https://www.guru99.com/difference-http-vs-https.html

DNS vulnerability

The Domain Name System (DNS) is the internet's version of the Yellow Pages. Back in the olden times, when you needed to find a business' address, you looked it up in the Yellow Pages. DNS is just like that, except you don't actually have to look anything up: your internet connected computer does that for you.

For two computers to communicate on an IP network, protocol dictates that they need an IP address. Think of an IP address like a street address – for one computer to "locate" another, they need to know the other computer's number. Since most humans are better at remembering names than numbers – 104.196.44.111, they needed a program for computers to translate names into IP addresses.

Hate computers professionally? Try Cards Against IT.

The program to translate names into numbers and vice versa is called, "DNS," or Domain Name System, and computers that run DNS are called, "DNS servers." Without DNS, we'd have to remember the IP address of any server we wanted to connect to – no fun.

How DNS Works

DNS is such an integral part of the internet that it's important to understand how it works.

Think of DNS like a phone book, but instead of mapping people's names to their street address, the phone book maps computer names to IP addresses. Each mapping is called a "DNS record."

The internet has a lot of computers, so it doesn't make sense to put all the records in one big book. Instead, DNS is organized into smaller books, or domains. Domains can be very large, so they are further organized into smaller books, called, "zones." No single DNS server stores all the books – that would be impractical.

Instead, there are lots of DNS servers that store all the DNS records for the internet. Any computer that wants to know a number or a name can ask their DNS server, and their DNS server knows how to ask – or query – other DNS servers when they need a record. When a DNS server queries other DNS servers, it's making an "upstream" query. Queries for a domain can go "upstream" until they lead back to domain's authority, or "authoritative name server."

An authoritative name server is where administrators manage server names and IP addresses for their domains. Whenever a DNS administrator wants to add, change or delete a server name or an IP address, they make a change on their authoritative DNS server (sometimes called a "master DNS server"). There are also "slave" DNS servers; these DNS servers hold copies of the DNS records for their zones and domains.

The Four DNS Servers that Load a Webpage

- **DNS recursor:** The DNS recursor is the server that responds to a DNS query and asks another DNS server for the address, or already has the IP address for the site saved.
- **Root name server**:A root name server is the name server for the root zone. It responds to direct requests and can return a list of authoritative name servers for the corresponding top-level domain.
- **TLD name server:** The top-level domain server (TLD) is one of the high-level DNS servers on the internet. When you search for TLD server for the '.com' will respond first, then DNS will search for 'varonis.'
- **Authoritative name server:** The authoritative name server is the final stop for a DNS query. The authoritative name server has the DNS record for the request.

Types of DNS Service

There are two distinct types of DNS services on the internet. Each of these services handles DNS queries differently depending on their function.

- **Recursive DNS resolver:** A recursive DNS resolver is the DNS server that responds to the DNS query and looks for the authoritative name server or a cached DNS result for the requested name.
- **Authoritative DNS server:** An authoritative DNS server stores the DNS request. So if you ask an authoritative DNS server for one of its IP addresses, it doesn't have to ask anyone else. The authoritative name server is the final authority on those names and IP addresses.

Public DNS and Private DNS

DNS was created so people could connect to services on the internet. For a server to be accessible on the public internet, it needs a public DNS record, and its IP address needs to be reachable on the internet – that means it's not blocked by a firewall. Public DNS servers are accessible to anyone that can connect to them and don't require authentication.

Interestingly, not all DNS records are public. Today, in addition to allowing employees to use DNS to find things on the internet, organizations use DNS so their employees can find private, internal servers. When an organization wants to keep server names and IP addresses private, or not directly reachable from the internet, they don't list them in public DNS servers. Instead, organizations list them in private, or internal DNS servers – internal DNS servers store names and IP addresses for internal file servers, mail servers, domain controllers, database servers, application servers, etc. – all the important stuff.

Something to remember – like external DNS servers, internal DNS servers don't require authentication. That's because DNS was created long ago, when security wasn't such a big concern. Most of the time, anyone on the inside of the firewall – by infiltration or connected through a VPN – can query internal DNS servers. The only thing that prevents someone "outside" from accessing and querying internal DNS servers is that they can't connect to them directly.

- **Public DNS:** For a server to be accessible on the public internet, it needs a public DNS record, and its IP address needs to be reachable on the internet.
- **Private DNS**: Computers that live behind a firewall or on an internal network use a private DNS record so that local computers can identify them by name. Outside users on the internet will not have direct access to those computers.

7 Steps in a DNS Lookup

Let's look at exactly how a DNS request works.

1. **A DNS request starts when you try to access a computer on the internet**. For example, you type in your browser address bar.
2. **The first stop for the DNS request is the local DNS cache.** As you access different computers, those IP addresses get stored in a local repository. If you visited before, you have the IP address in your cache.
3. **If you don't have the IP address in your local DNS cache, DNS will check with a recursive DNS server.** Your IT team or Internet Service

Provider (ISP) usually provides a recursive DNS server for this purpose.

4. **The recursive DNS server has its own cache, and if it has the IP address, it will return it to you.** If not, it will go ask another DNS server.
5. **The next stop is the TLD name servers, in this case, the TLD name server for the .com addresses.** These servers don't have the IP address we need, but it can send the DNS request in the right direction.
6. **What the TLD name servers do have is the location of the authoritative name server for the requested site.** The authoritative name server responds with the IP address for and the recursive DNS server stores it in the local DNS cache and returns the address to your computer.
7. **Your local DNS service gets the IP address and connects to to download all the glorious content.** DNS then records the IP address in local cache with a time-to-live (TTL) value. The TTL is the amount of time the local DNS record is valid, and after that time, DNS will go through the process again when you request Varonis.com the next time.

What are Types of DNS Queries?

DNS queries are the computer code that tells the DNS servers what kind of query it is and what information it wants back. There are three basic DNS queries in a standard DNS lookup.

- **Recursive query:** In a recursive query the computer requests an IP address or the confirmation that the DNS server doesn't know that IP address.
- **Iterative query:** An iterative query the requester asks a DNS server for the best answer it has. If the DNS server doesn't have the IP address, it will return the authoritative name server or TLD name server. The requester will continue this iterative process until it finds an answer or times out.

- **Non-recursive query:** A DNS resolver will use this query to find an IP address that it doesn't have in its cache. These are limited to a single request to limit network bandwidth usage.

Proof of vulnerability

Not Allowed to perform

Reference

https://www.varonis.com/blog/what-is-dns/

Violation of Secure Web Design Principal

Description

The product violates well-established principles for secure design.

Extended Description

This can introduce resultant weaknesses or make it easier for developers to introduce related weaknesses during implementation. Because code is centered around design, it can be resource-intensive to fix design problems.

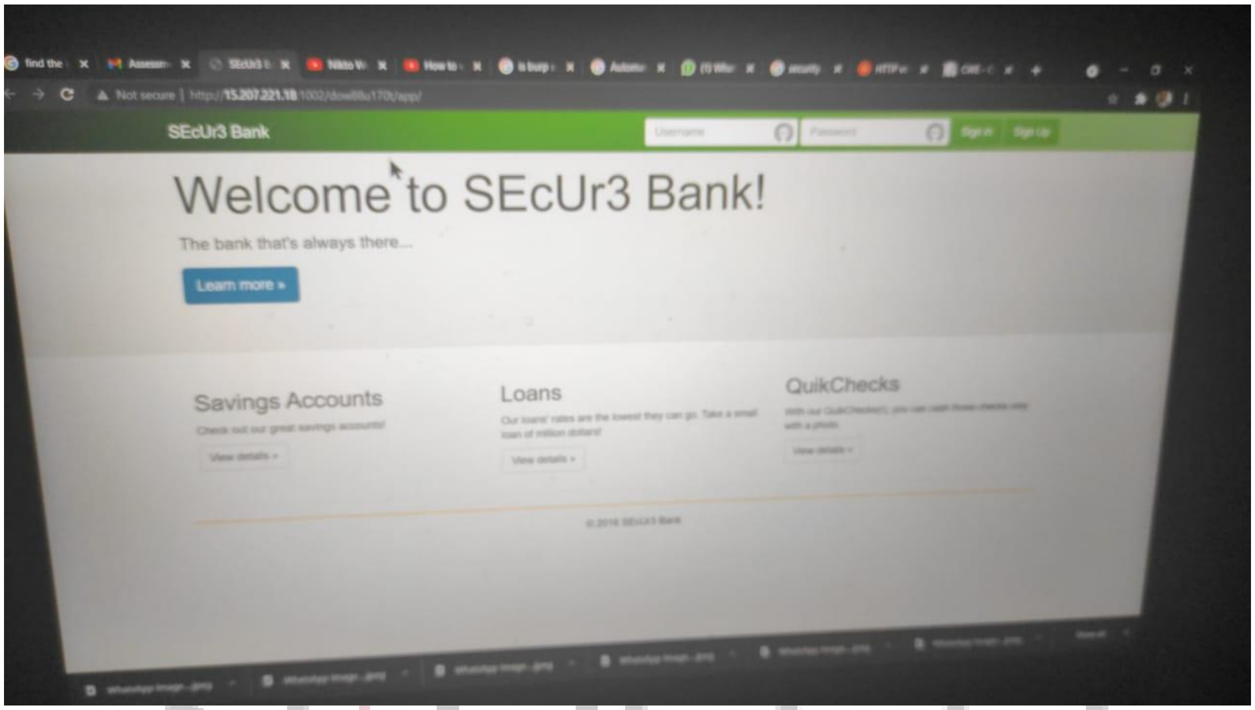
Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOf and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that the user may want to explore.

Impact

The table below specifies different individual consequences associated with the weakness. The Scope identifies the application security area that is violated, while the Impact describes the negative technical impact that arises if an adversary succeeds in exploiting this weakness. The Likelihood provides information about how likely the specific consequence is expected to be seen relative to the other consequences in the list. For example, there may be high likelihood that a weakness will be exploited to achieve a certain impact, but a low likelihood that it will be exploited to achieve a different impact.

Proof of Vulnerability



Reference

https://cwe.mitre.org/data/definitions/657.html

Reported By
Roshan Kumar

Contact Details
Ph: 9100794126
roshan11802478@gmail.com