
NorthIT™
Web-App Penetration Testing – Overview/Training

Key Topics Overview

NorthIT

- Web-App Penetration Testing
  - What is it.
  - Why is it important
- OWASP
  - What OWASP does.
  - The OWASP top 10.
  - Why the top 10 isn't enough.
- High level overview of OWASP methodology testing.
  - What website testers look for.
  - What key information assists the attacker in exploiting a system.

What is web app penetration testing?

NorthIT™

- Web application penetration testing works by using manual or automated penetration tests to identify any vulnerability, security flaws or threats in a web application.
- The tests involve using/implementing any of the known malicious penetration attacks on the application.
- The de-facto standard for web application testing is the OWASP standard.

What is OWASP?

NorthIT

- OWASP stands for the Open Web Application Security Project.
- The primary goal of OWASP is to spread awareness around software security.
- Compiles methods and information repositories on web application security.
- OWASP sets a security standard for testing, which should be done be qualified professionals.
- OWASP curates a list of the top 10 vulnerabilities every other year. This list is based upon what we, the penetration testers. Tell them we are seeing.

The OWASP Top 10

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Is the top 10 enough?

NorthIT

- Short answer is no. Testers should complete a full OWASP standard penetration test.
  - What about number 11?
- Many vulnerability scanning companies/software vendors. Claim to test for the OWASP top 10.
  - This isn't enough. If you test for the top 10 vulnerabilities exclusively. The chances are you might be attacked via number 11.
- Everything must be tested within the application, not just the most prevalent issues. (OWASP Top 10)

The OWASP standard.

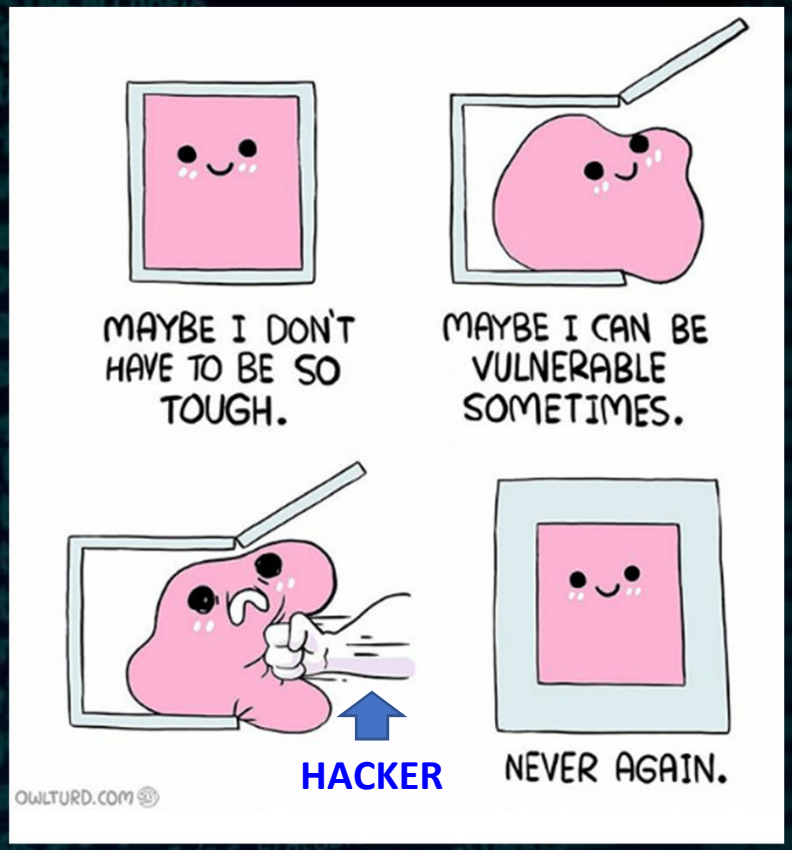
NorthIT

- Information Gathering
- Configuration and Deployment Management Testing
- Identity Management Testing
- Authentication Testing
- Authorization Testing
- Session Management Testing
- Input Validation Testing
- Error Handling
- Cryptography
- Business Logic Testing
- Client Side Testing

Important Note

NorthIT

- It takes only a single vulnerability to undermine the security of the entire infrastructure.



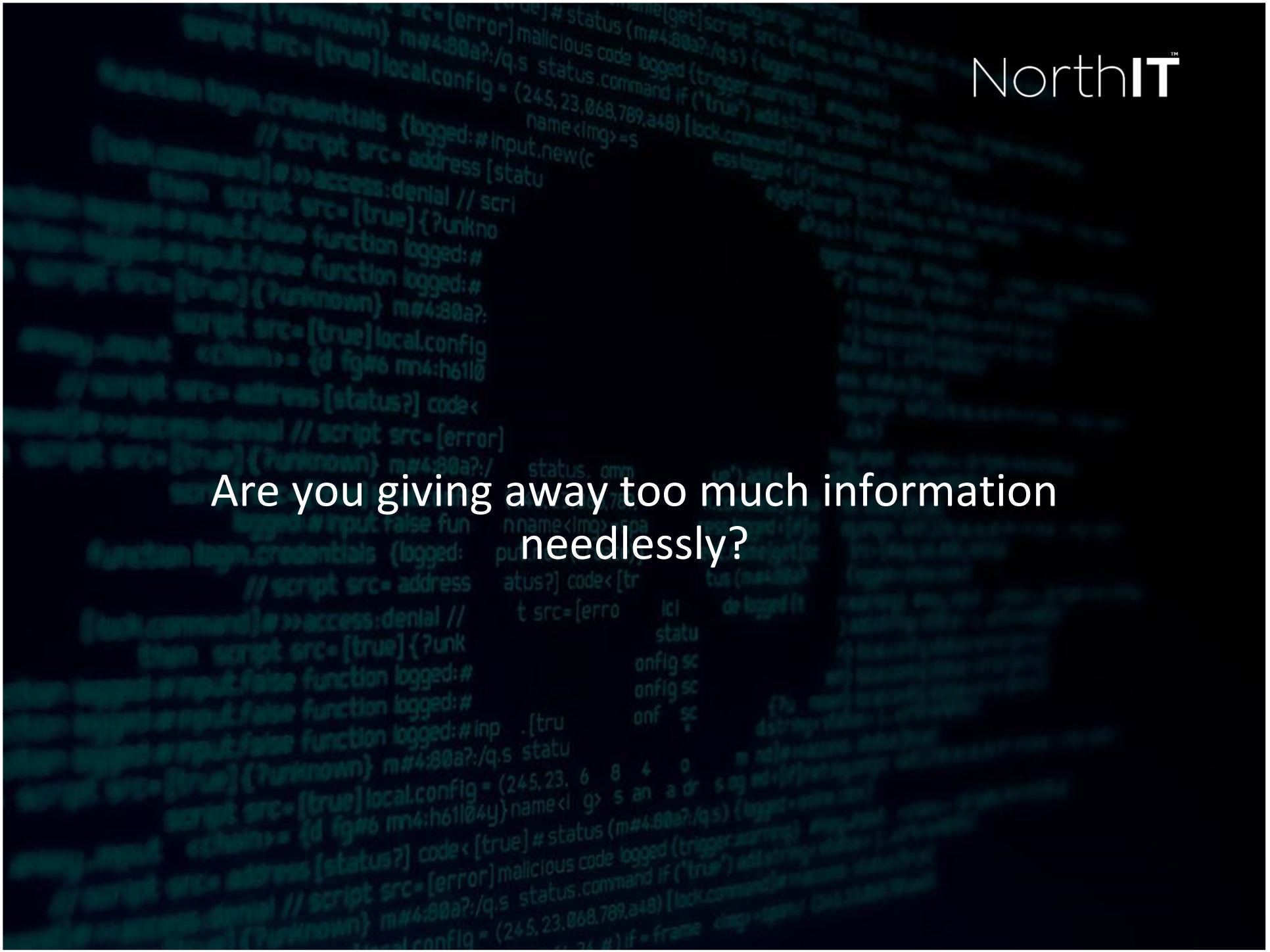
NorthIT™

INFORMATION GATHERING

Information Gathering

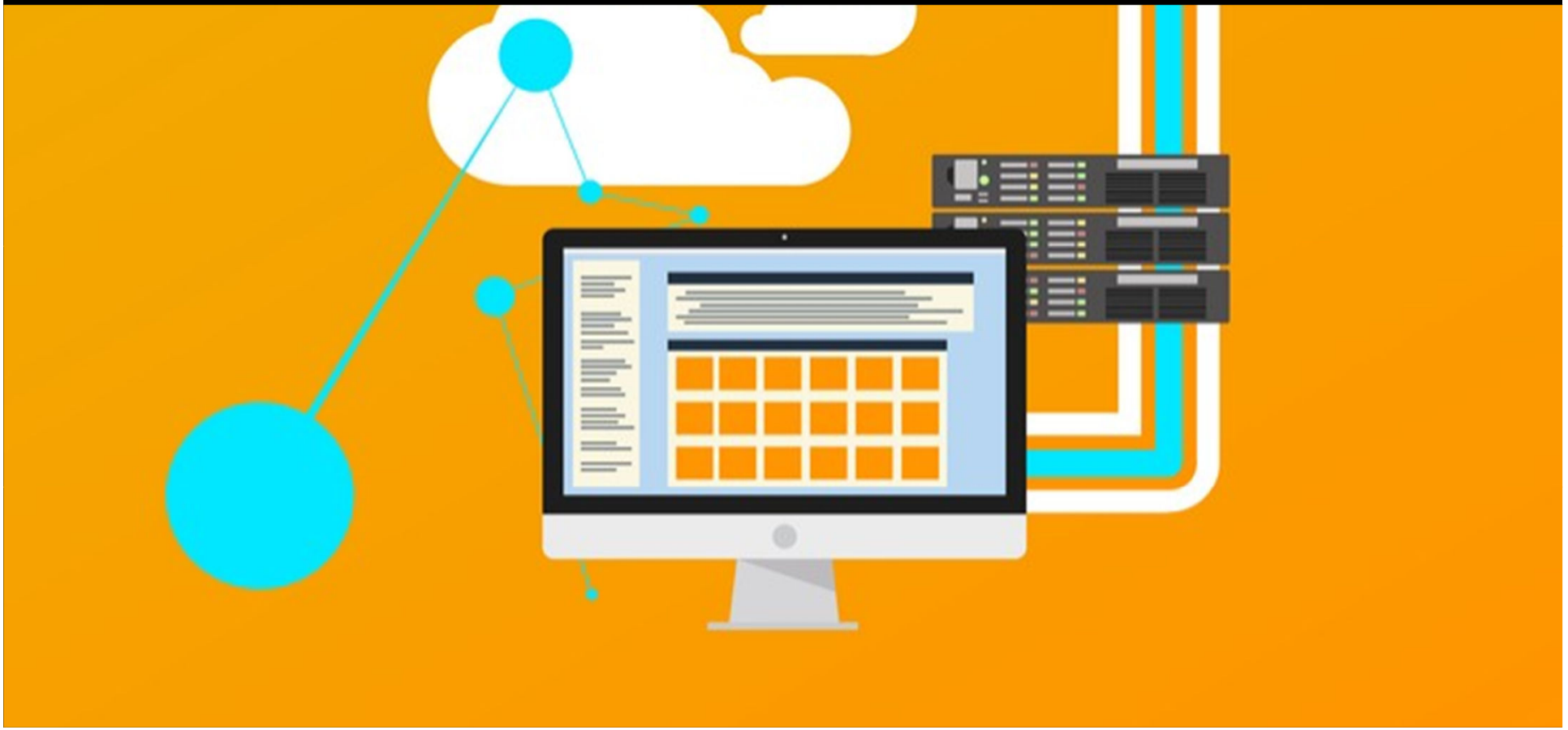
NorthIT

- What information can be gathered from Google?
- What information is being given away by the server response? (Powered by: X)
- Robots.txt file, if you don't want google to crawl it. I want to see what's being hidden?
- Consider what directories can be extracted from somebody outside of the standard user. (Directory brute-forcing.
- Subdomains?


NorthIT
Are you giving away too much information needlessly?

NorthIT™

CONFIG/DEPLOYMENT MANAGEMENT TESTING



Config/Deployment management testing

NorthIT

- Admin panels visible? Is it properly secured? If it was breached, what could potentially happen?
- What HTTP methods are allowed? (Methods: HEAD • GET • POST • PUT • DELETE • TRACE • OPTIONS • CONNECT)
- Strict transport security (Is it Enabled or not?)
- What HTTP Security Headers have been implemented?

Http Security-Headers

NorthIT

- What are the main Security-Headers we check for?

Server	This Server header seems to advertise the software being run on the server but you can remove or change this value.
Referrer-Policy	Referrer Policy is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.
Feature-Policy	Feature Policy is a new header that allows a site to control which features and APIs can be used in the browser.
Strict-Transport-Security	HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to enforce the use of HTTPS.
X-Content-Type-Options	X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".
Referrer-Policy	Referrer Policy is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.
X-Frame-Options	X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking.
X-XSS-Protection	X-XSS-Protection sets the configuration for the XSS Auditor built into older browser. The recommended value was "X-XSS-Protection: 1; mode=block" but you should now look at Content Security Policy instead.
Content-Security-Policy	Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets. Analyse this policy in more detail. You can sign up for a free account on Report URI to collect reports about problems on your site.


IDENTITY MANAGEMENT

Identity Management - Overview

NorthIT

- Permission/Roles matrix. Do you have one?
- Is the user registration process valid? Can you verify the details, are you asking for too much information needlessly?
- Can valid emails/usernames be extracted via the websites authentication mechanisms?

Username enumeration – Generic responses.

NorthIT

- Keep it generic – A response such as “The credentials supplied are not valid” should suffice.
  - Leaking information about if the account is valid or not in the responses. Can allow an attacker to probe the website for valid usernames/email addresses.
- Are usernames predictable? Do they have a specific format? ‘j.bloggs?’

NorthIT™

AUTHENTICATION TESTING

Authentication Testing - Overview

NorthIT

- Weak password policies.
  - Password complexity/entropy.
  - Using standard words contained within a dictionary.
  - Weak password hashing algorithm?
- Weak reset password functions.
  - Information leakage?
- Browser cache weaknesses.

Password jargon buster

NorthIT™

- **Password Complexity**: the rules associated with setting passwords to try and guarantee that the passwords used are both difficult-to-crack as well as difficult-to-guess.
- **Password Entropy**: the level of chaos or randomness present in a system -- in this case, a string of characters that make up a password.
- **Bits of Entropy**: the mathematical measurement, in bits, of how difficult it is to crack a password.

Password Complexity.

NorthIT

- ***Good passwords / passphrases***:
- should be 8 characters or longer, which forces you to use multiple words or extra symbols.
- should have upper case, lower case, symbols, and numbers; or at least three of those four groups.
- should not be a common word and should not be a common phrase.
- should not contain a date, a name, or other things that can be associated with you.
- should be created randomly or semi-randomly.
- should not be a suggestion when you type in the first few characters into Google.

Password Entropy.

NorthIT

- **Password entropy** is a measurement of how unpredictable a **password** is. **Password entropy** is based on the character set used (which is expansible by using lowercase, uppercase, numbers as well as symbols) as well as **password** length. ... Use a minimum of 8 characters selected from a 94-character set.
- **Example -** the password '***password***' would have a possible pool of 26 characters from the English alphabet.
- Changing the password to '***Password***' would increase your pool to 52 characters.

Password Entropy Table

NorthIT

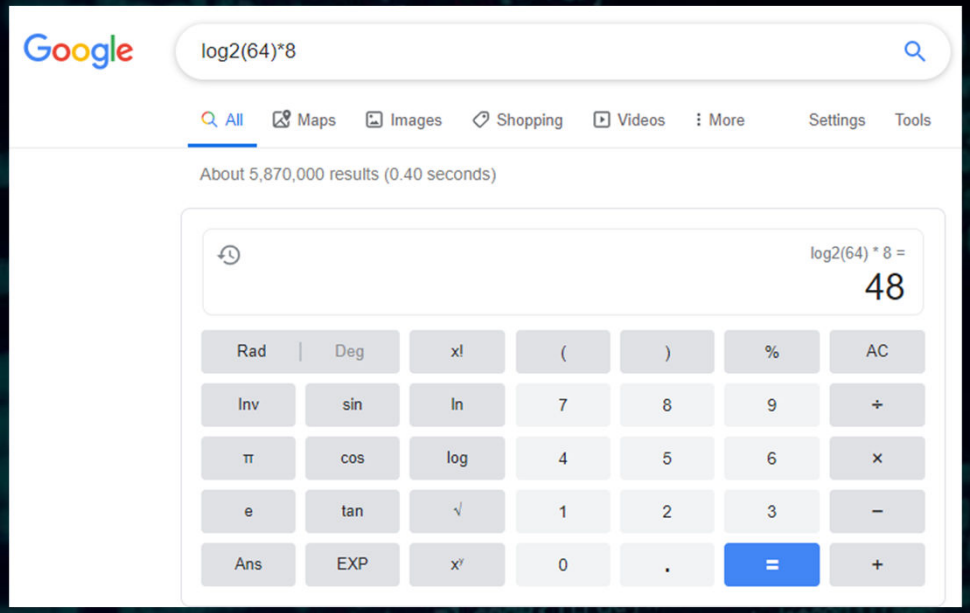
Type	Possible Characters
Lowercase	26
Uppercase	26
Numbers	10
Specials	33
All of the above	95

- Given the following password policy “The password must be at least 8 characters in length, containing a combination of lowercase, uppercase and numbers”
- The number possible characters would be 62 (26+26+10), it’s entropy would be 48.

Password entropy calculation.

NorthIT

- Entropy can be calculated using a calculator, or if you want. Use google.
- log2(numberofpossiblecharacters)*passwordlength



Password entropy – Exercise.

NorthIT

- Work out the password entropy for the given policies. (Based off of the minimum requirements.)
  - Your password must be at least 8 characters in length, containing a combination of lowercase, uppercase characters and numbers. - 47 Bits
  - Your password must be at least 9 characters in length, containing a combination of lowercase, uppercase characters. - 51 Bits
  - Your password must be at least 10 characters in length, containing upper/lowercase characters, numbers and specials. - 65 Bits

Password Hashing Algorithms

NorthIT

- Secure hashing algorithms:
  - Argon2 should be considered as your first choice for new applications.
  - PBKDF2 when FIPS certification or enterprise support on many platforms is required.
  - Scrypt where resisting any/all hardware accelerated attacks is necessary but support isn't.
  - Bcrypt where PBKDF2 or Scrypt support is not available.
- Design password storage assuming eventual compromise, this makes things harder for the attacker if they do breach the database.

Keeping passwords secure.

NorthIT

- You can increase password entropy by increasing length, or by increasing the character pool.
- Passwords must be truly random data from the total available pool in order for it to be secure.
- For example “Autumn2019**" has 78 bits of entropy: log2(95)*12, but is not very secure because:
  - It's vulnerable to a dictionary based attack.
  - It’s very common to see the season/year combination within password lists.
- If you have a habit of setting insecure passwords, using a password manager is vastly more secure than setting weak passwords.
- If possible use a password blacklist, to limit the weak passwords set by users.
- Use a strong hashing algorithm.

Top 10 Password of varying lengths.

NorthIT

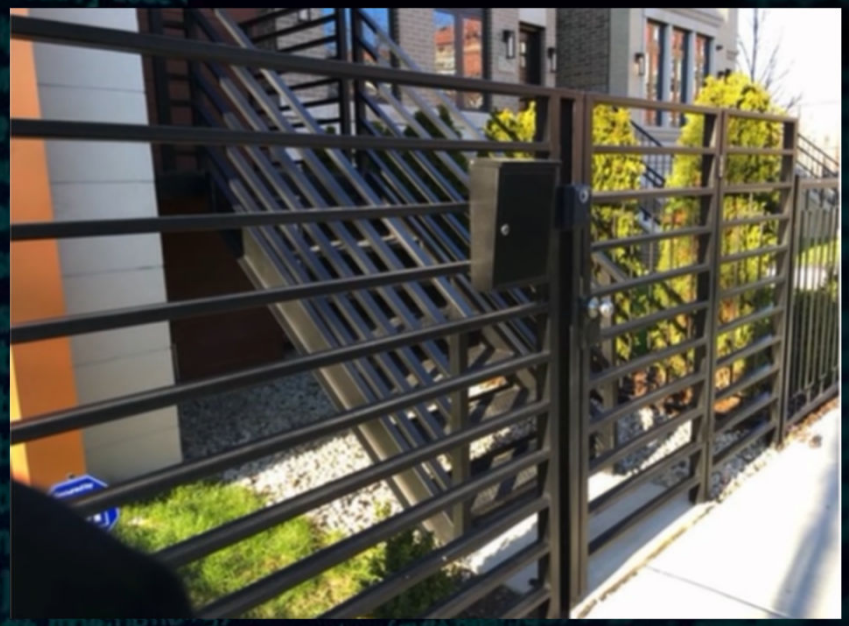
- People don't like to attempt to remember complex passwords.. (Use a password manager)
- Highlighted in green are common keyboard patterns. (Taken from 1 Billion leaked passwords)

Top 10 Passwords of varying lengths.							
Password Overall	Count	Length = 8	Count	Length = 9	Count	Length = 10	Count
123456	5,022,959	password	1,014,096	123456789	1,834,213	1234567890	252,934
123456789	1,834,213	12345678	762,805	password1	676,916	1q2w3e4r5t	200,420
qwerty	1,334,749	iloveyou	397,742	target123	205,931	qwertyuiop	142,295
password	1,014,096	1qaz2wsx	243,631	gwerty123	200,971	123456789a	37,815
12345678	762,805	myspace1	211,338	qwerty123	175,975	3rJs1la7qE	27,639
abc123	761,844	1q2w3e4r	203,962	asdfghjkl	161,236	VQsaBLPzLa	25,526
111111	717,664	1g2w3e4r	201,372	987654321	155,809	987654321	25,022
password1	676,916	zag12wsx	201,065	123123123	145,746	12345qwert	24,755
1234567	664,054	1234qwer	136,789	789456123	118,636	123456789	24,638
1234567890	635,751	computer	127,664	iloveyou1	108,348	1111111111	23,303

Remember….

NorthIT

- “It takes only a single vulnerability to undermine the security of the entire infrastructure.”
  - This includes passwords.



Password reset functions.

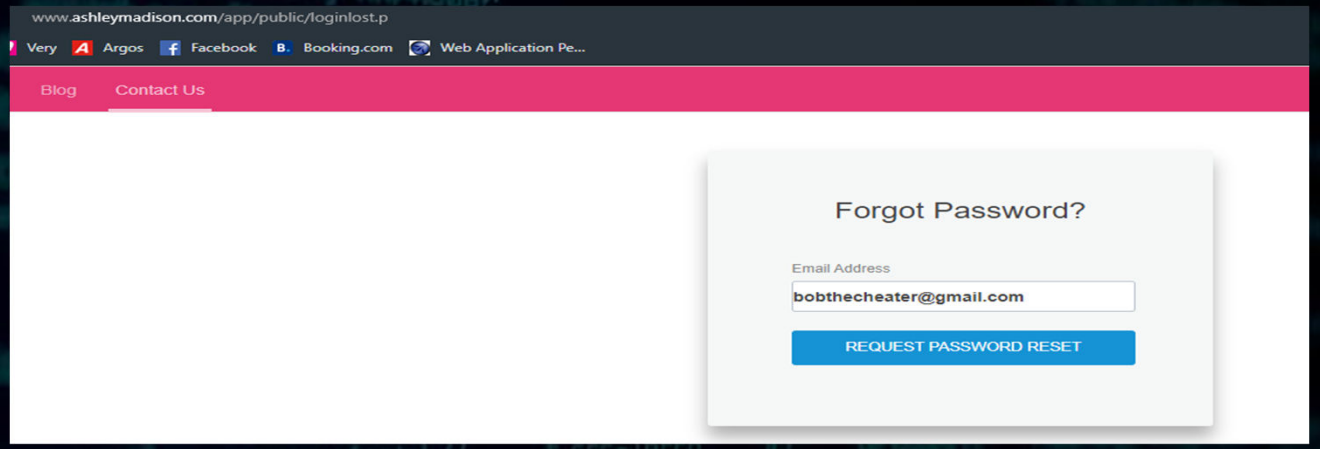
NorthIT

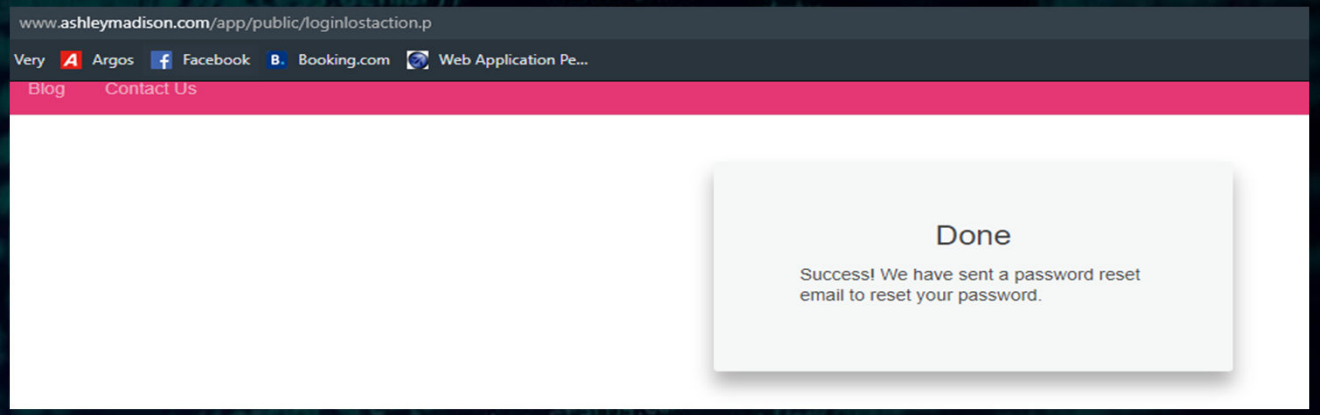
- Security vs Usability?
- Include generic responses when a user requests a password reset.
  - Generic as possible, try not to reveal information about valid user/email accounts.
- Attackers can enumerate possible valid credentials from the reset mechanisms.
- ***Example of a poor reset mechanism:***
  - **Invalid username:** e-mail address is not valid or the specified user was not found.
  - **Valid username:** Your password has been successfully sent to the email address you registered with.

Ashley Madison?

NorthIT™

- Imagine if a reset function on the Ashley Madison website revealed that the email was found?.





Browser cache weaknesses.

NorthIT™

- Cache and history are two different entities.
- If a user logs into the application, views sensitive information. Logs out, then tries using the browsers back function to display it again. What information is shown?
- Cache-Control- Revalidate wherever possible.
- Forbid browsers from storing information.


SESSION MANAGEMENT

Session Management - Overview

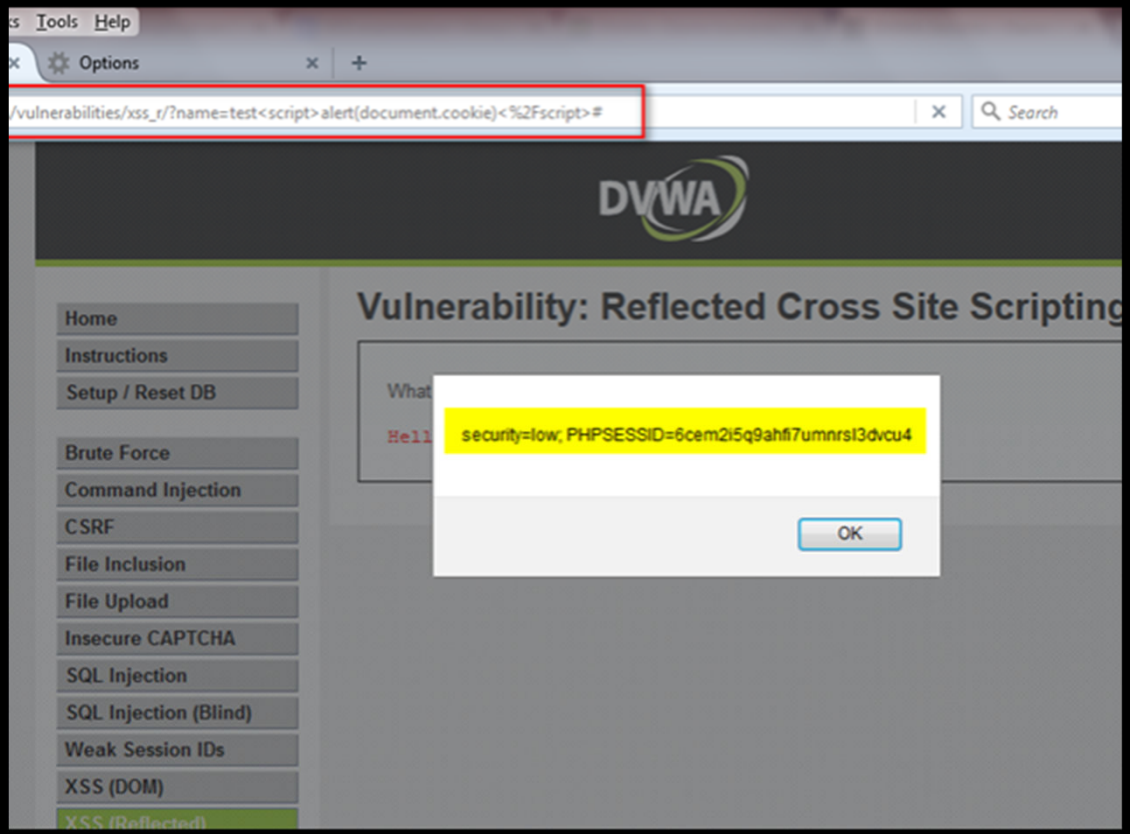
NorthIT

- Securing cookies.
- Session fixation.
- Server-side session termination.
- Cross-site request forgery.
- Logout functions.
- Session puzzling.

Secure cookie attributes.

NorthIT

- **Secure** - This attribute tells the browser to only send the cookie if the request is being sent over a secure channel such as HTTPS. This will help protect the cookie from being passed over unencrypted requests. If the application can be accessed over both HTTP and HTTPS, then there is the potential that the cookie can be sent in clear text.
- **HttpOnly** - This attribute is used to help prevent attacks such as cross-site scripting, since it does not allow the cookie to be accessed via a client side script such as JavaScript. ***Note that not all browsers support this functionality.***
- **Domain** - This attribute is used to compare against the domain of the server in which the URL is being requested. If the domain matches or if it is a sub-domain, then the path attribute will be checked next.



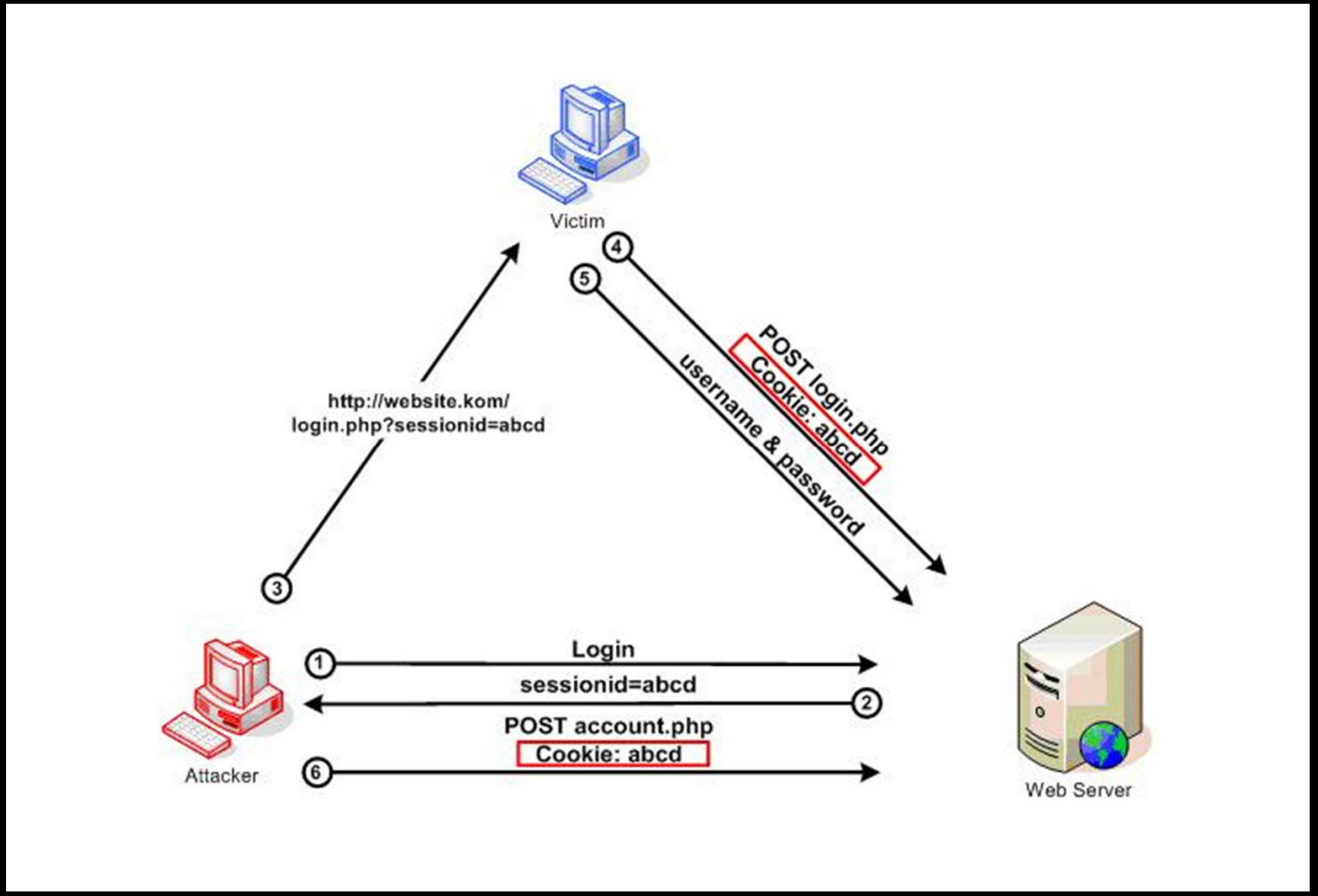
XSS – Stealing cookies

- This example shows the cookie being stolen using client side XSS attack.
- **<script>alert(document .cookie)<%2Fscript>**
- This could potentially lead to the attacker stealing the cookie and using this to authenticate. Providing that the application allows session hijacking.

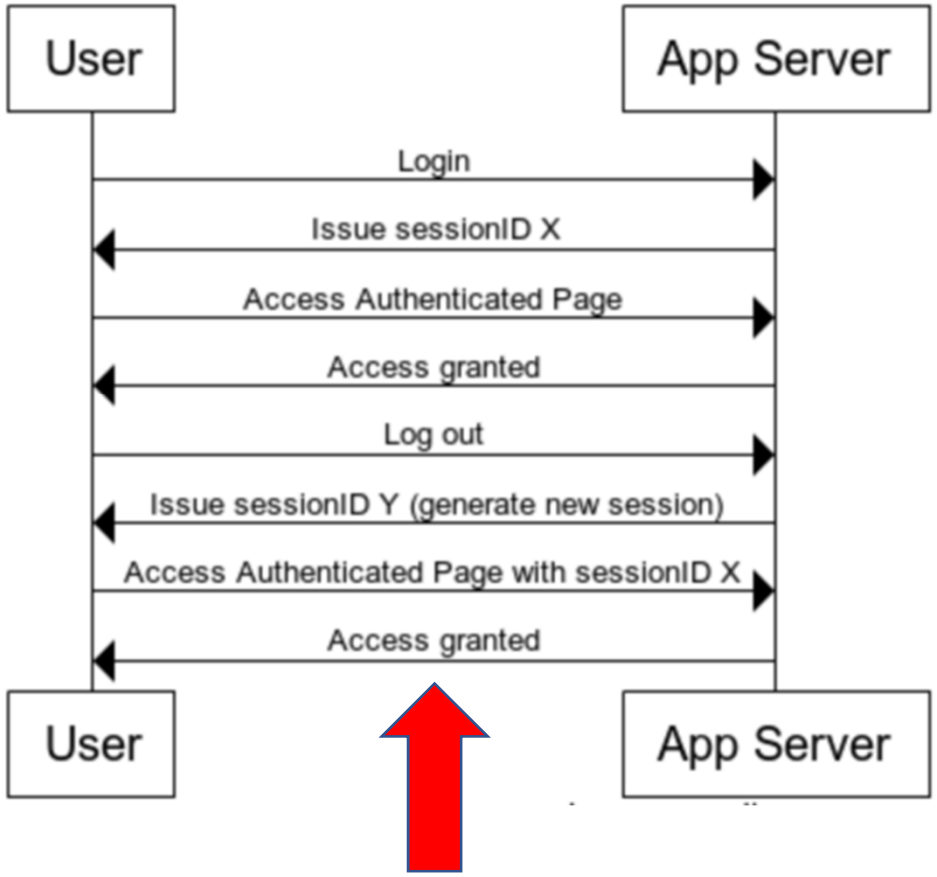
Session fixation .

NorthIT

- Websites should issue a new session ID upon successful authentication.
  - If a new session ID is not issued upon successful authentication, this could lead to a session fixation attack.


Victim
4
5
POST login.php
Cookie: abcd
username & password
http://website.kom/
login.php?sessionid=abcd
3
Attacker
1
Login
sessionid=abcd
2
POST account.php
Cookie: abcd
6
Web Server

Server-side session termination



Should not happen

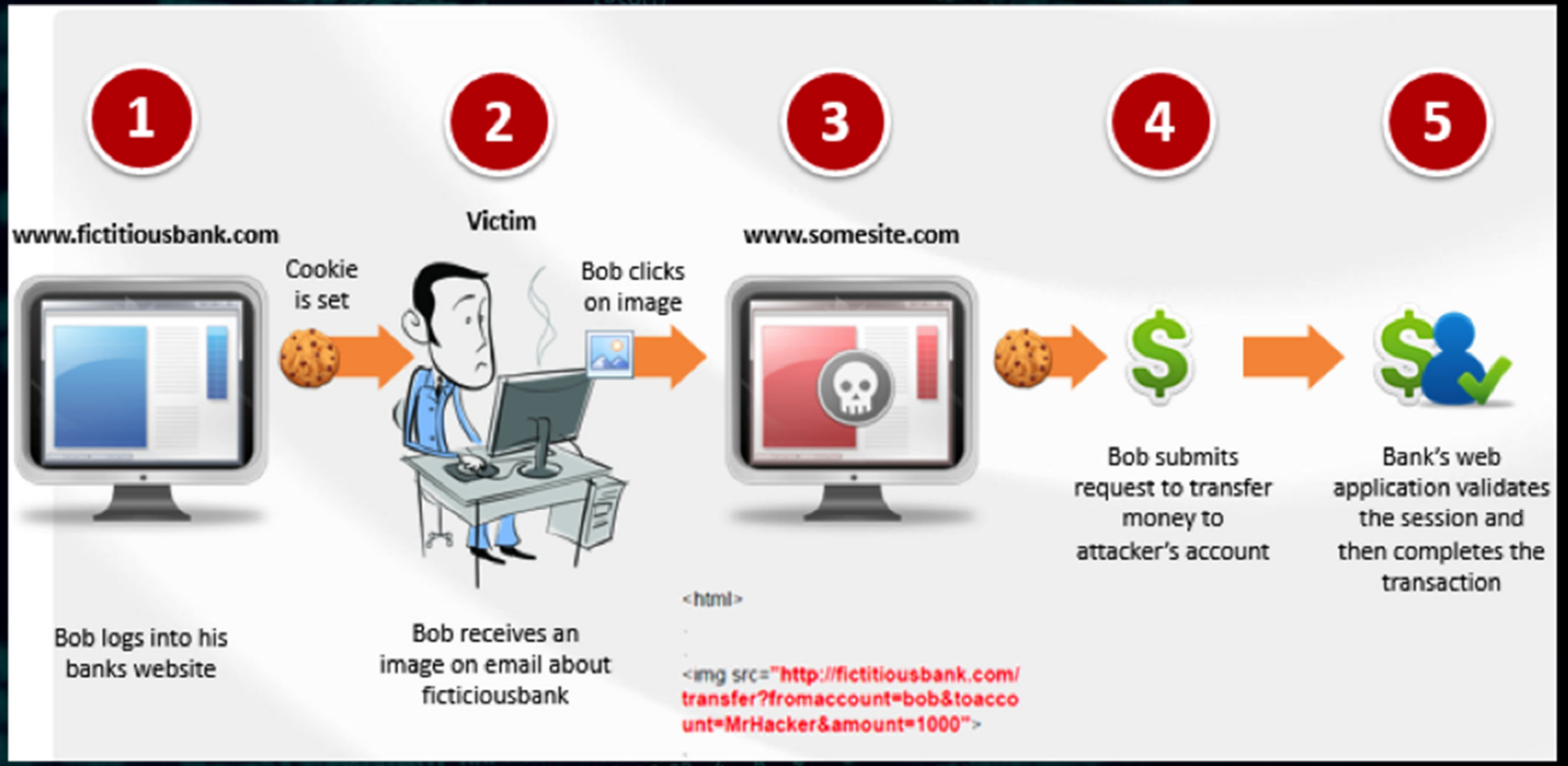
Cross-site request forgery.

NorthIT

- Briefly explained: CSRF is a type of attack which tricks the victim to do the malicious task on a victim authenticated web application on behalf of attackers interests.
- CSRF attacks generally consist of the following:
  - **1)** The first step is to trick the victim into clicking a link or loading up a page. This is normally done through social engineering. By using social engineering methods attacker will lure the user to click the link.
  - **2)**The second step is to send a “forged” or made up request to the victim’s browser. This link will send a legitimate-looking request to the web application. The request will be sent with the values that the attacker wants. Apart from them, this request will include any cookies that the victim has associated with that website.

CSRF EXAMPLE

NorthIT



Logout function.

NorthIT™

- Availability of user interface controls that allow the user to manually log out. (***Obvious, but often not properly thought about***)
- Session termination after a given amount of time without activity (***Session timeout***).
- Proper invalidation of server-side session state.

Injection Testing (Input).

- Arguably the most prevalent and critical aspect of the test. Certain forms of injection, in particular SQL Injection. Can lead to complete system takeover and lasting damage, both to reputation and users personal information.

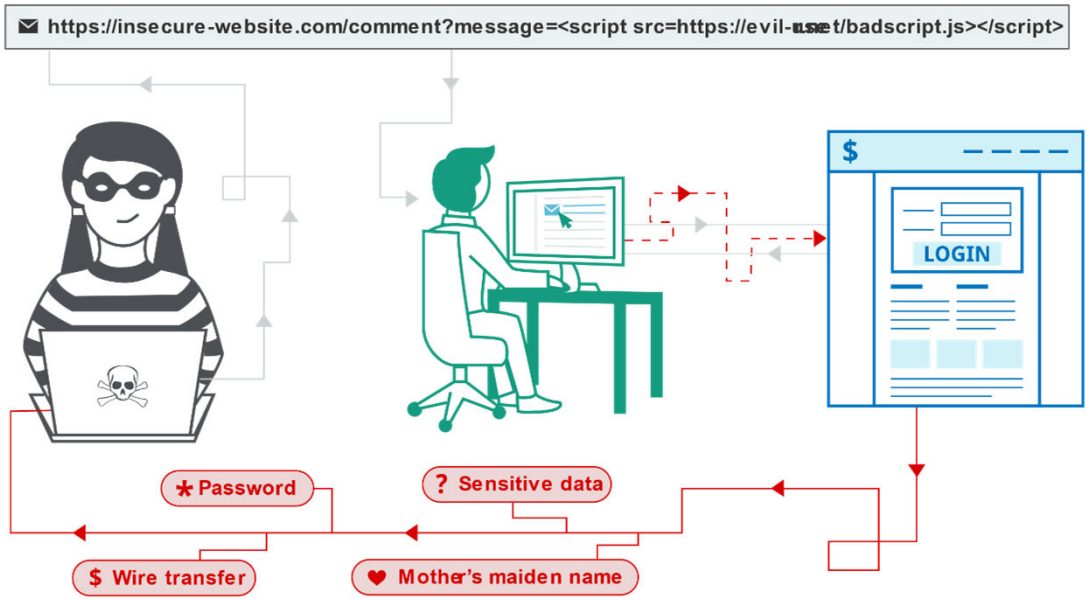
Injection Testing Overview

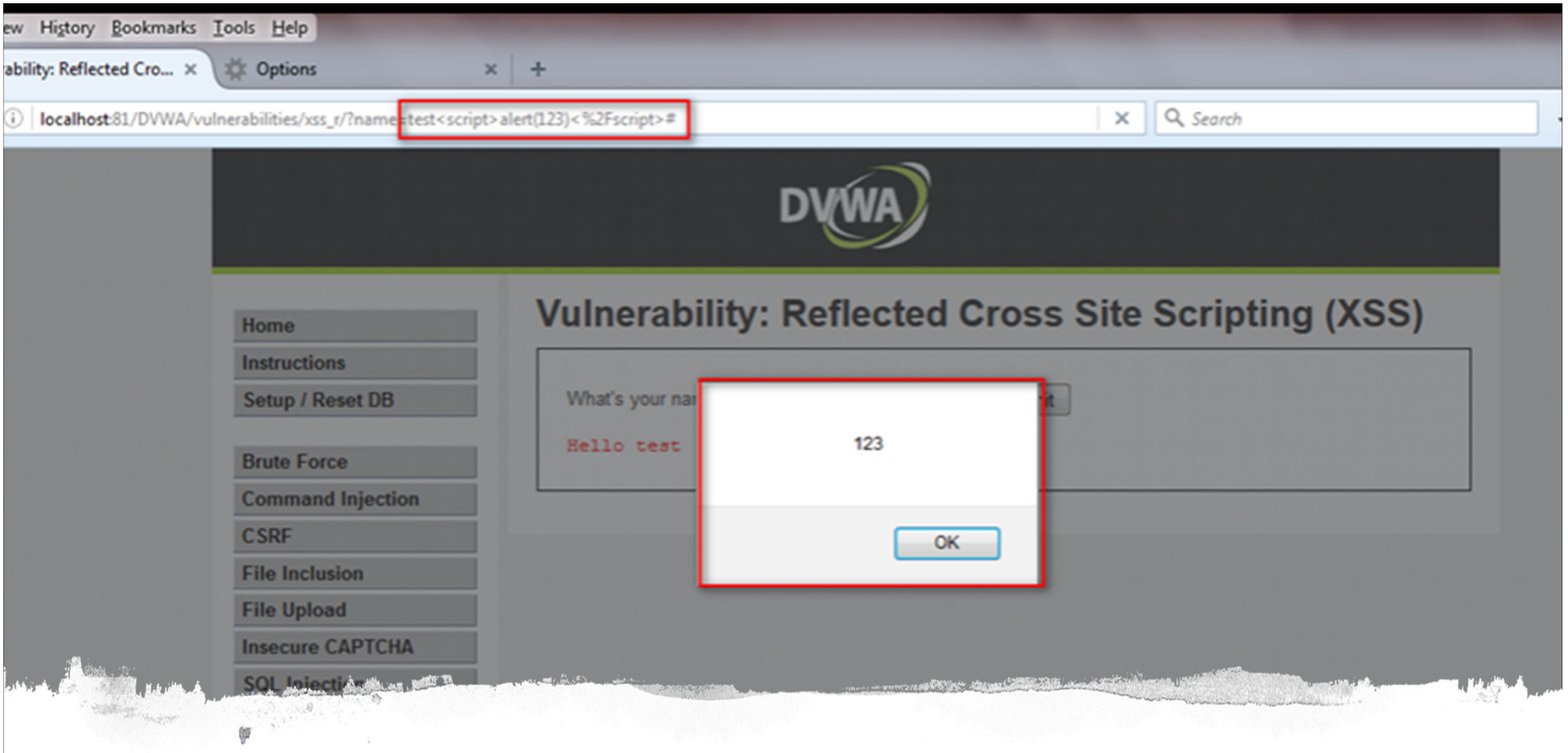
NorthIT

- Reflected XSS.
- Stored XSS.
- DOM XSS.
- SQL Injection.
- Local File Inclusion.
- Remote File Inclusion.
- XML Injection.
- Command Injection.
- This list is not exhaustive! We wouldn't have enough time to cover every attempted injection.

XSS Scripting

- XSS Scripting comes in 3 main categories.
  - **Reflected XSS attacks.**
  - **Stored XSS attacks.**
  - **DOM based XSS attacks (Document object model)**



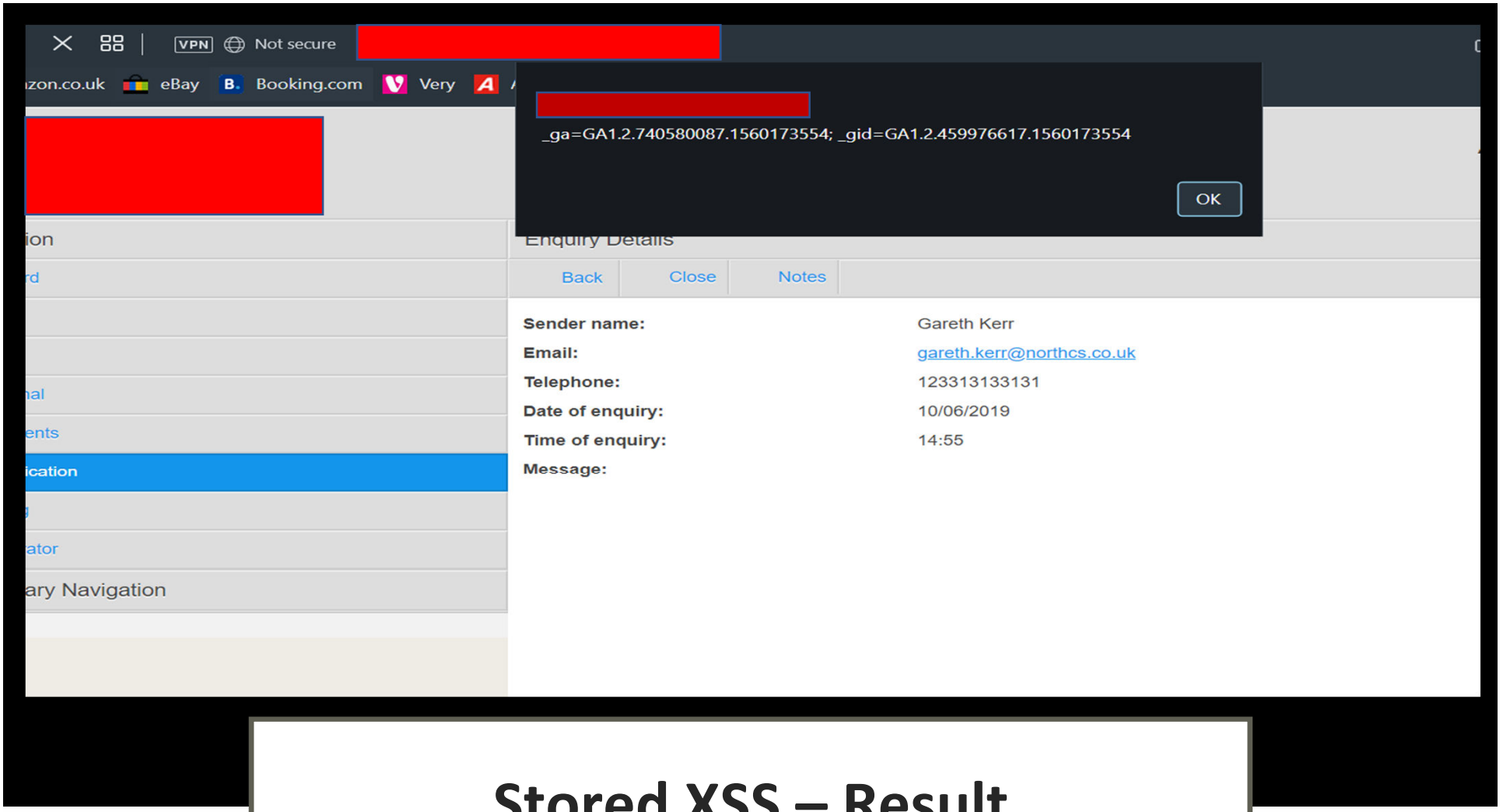


Reflected XSS.

- Where the malicious input comes from the current HTTP request.
- XSS Payload: **<script>alert('123')</script>**

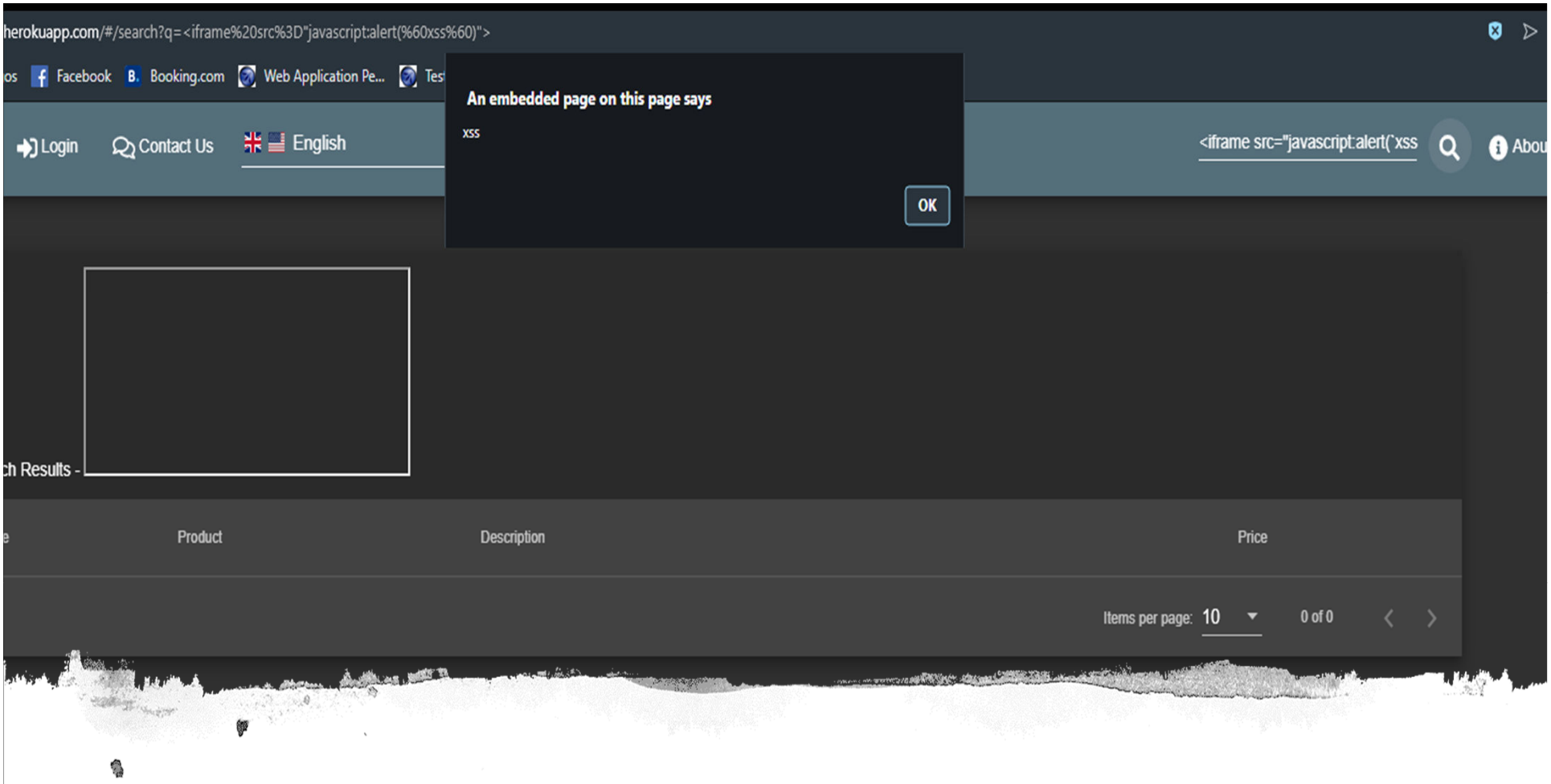
Stored XSS – Payload

- Where the malicious script comes from the website's database, this example comes from a recent presentation. This payload was sent to the database. XSS Payload: **<script>alert(document.cookie)</script>**



Stored XSS – Result

- The XSS payload was then retrieved from the database within the admin panel. This came through the enquiry panel.

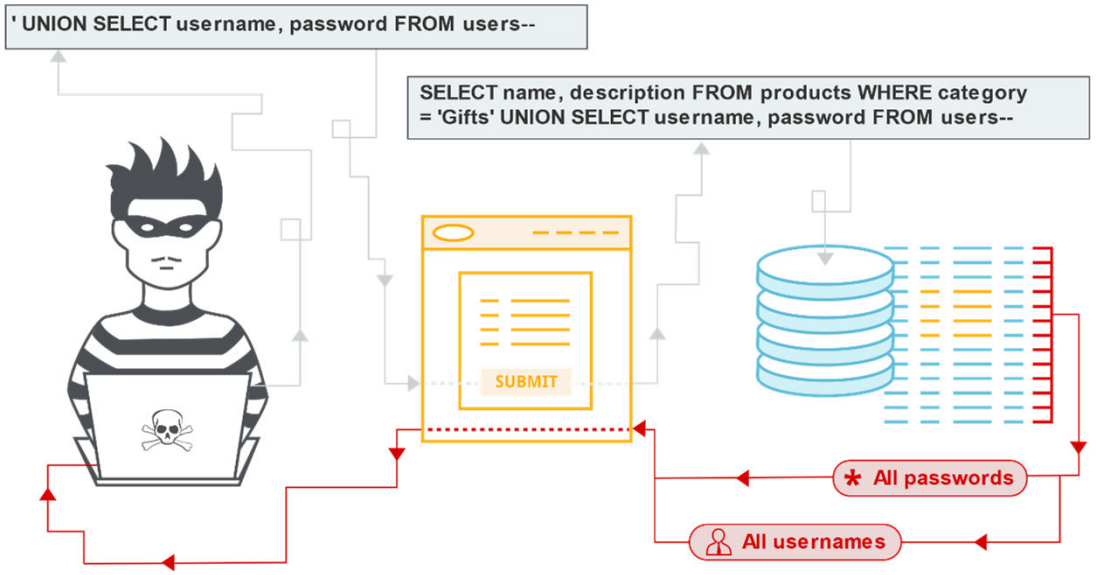


DOM XSS.

- Where the vulnerability exists in client-side code rather than server-side code.
- XSS Payload: **<iframe src="javascript:alert(`xss`)"**

SQL INJECTION

- SQL Injection - SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve.



SQL INJECTION

- SQL Injection comes in many wonderful forms. It can be used to enumerate users, obtain version information or even subvert the login of the application.
- Consider the following example.

- Consider a shopping application that displays products in different categories. When the user clicks on the Gifts category, their browser requests the URL:
- https://insecure-website.com/products?category=Gifts
- This causes the application to make an SQL query to retrieve details of the relevant products from the database:
- SELECT * FROM products WHERE category = 'Gifts' AND released = 1
- This SQL query asks the database to return:
- The restriction released = 1 is being used to hide products that are not released. For unreleased products, presumably released = 0.
- The application doesn't implement any defenses against SQL injection attacks, so an attacker can construct an attack like:
- https://insecure-website.com/products?category=Gifts'--
- This results in the SQL query:
- SELECT * FROM products WHERE category = 'Gifts'--' AND released = 1
- The key thing here is that the double-dash sequence -- is a comment indicator in SQL, and means that the rest of the query is interpreted as a comment. This effectively removes the remainder of the query, so it no longer includes AND released = 1. This means that all products are displayed, including unreleased products.

LFI - (Local File Inclusion)

NorthIT

- The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a "dynamic file inclusion" mechanisms implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation.
- This can lead to issues such as:
  - Code execution on the web server
  - Code execution on the client-side such as JavaScript which can lead to other attacks such as cross site scripting (XSS)
  - Denial of Service (DoS)
  - Sensitive Information Disclosure

LFI Example

- Exploit -https://exampleurl/section.php?page=../../../../etc/passwd%00
- Contents of the /etc/passwd file are displayed on the page. Dependent on the nature of the vulnerability, it may be possible to obtain a reverse shell.

RFI (Remote File Inclusion)

NorthIT™

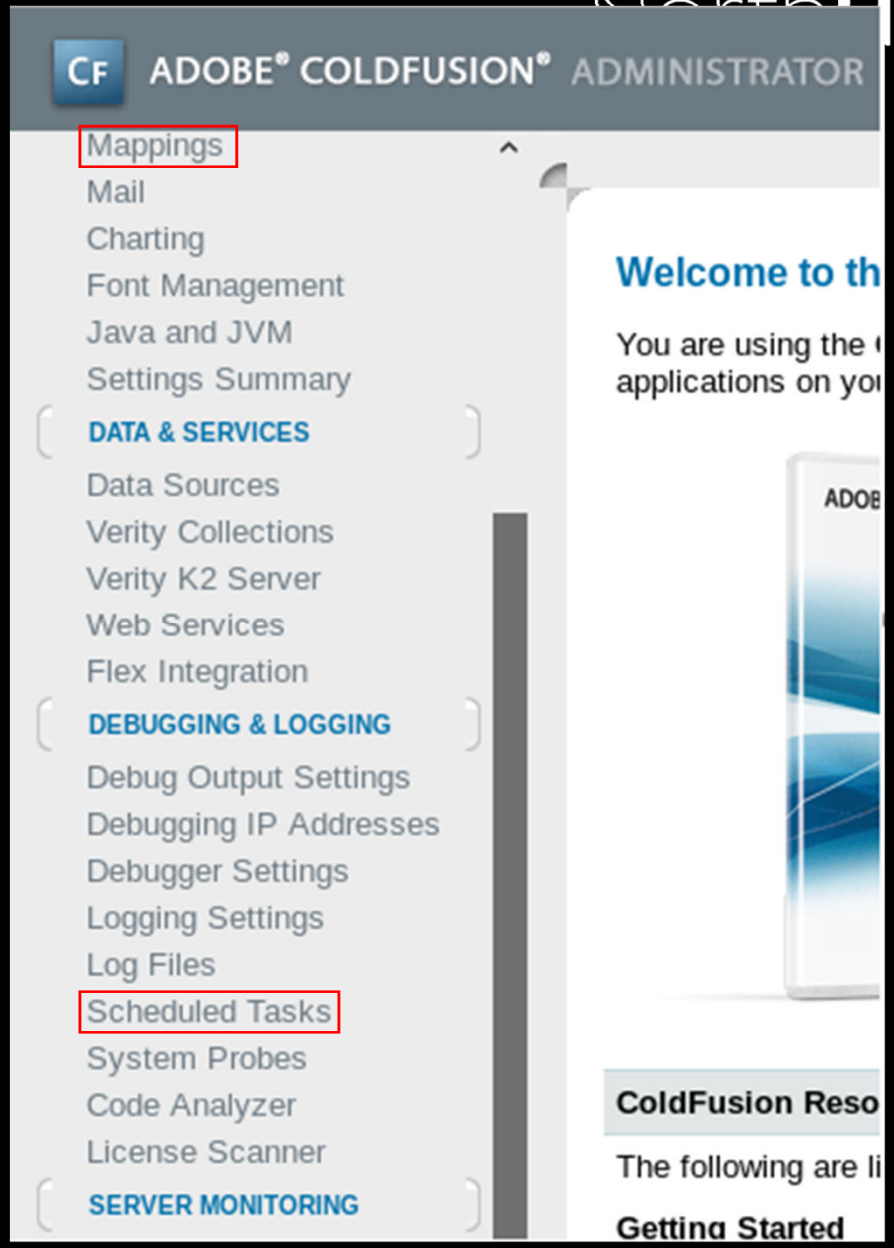
- The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a "dynamic file inclusion" mechanisms implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation.
- This can lead to issues such as:
  - Code execution on the web server
  - Code execution on the client-side such as JavaScript which can lead to other attacks such as cross site scripting (XSS)
  - Denial of Service (DoS)
  - Sensitive Information Disclosure

RFI Example

- Scenario: The task scheduler from a cold fusion 8 web administration panel. This is an example of how an administration panel, if accessed by an attacker. Can be utilized to gain further access to the system.
- Continued on the next slide….

RFI Example - Coldfusion

- The specific problem is highlighted in red. The ‘Mappings’ and ‘Scheduled Tasks’ interfaces.



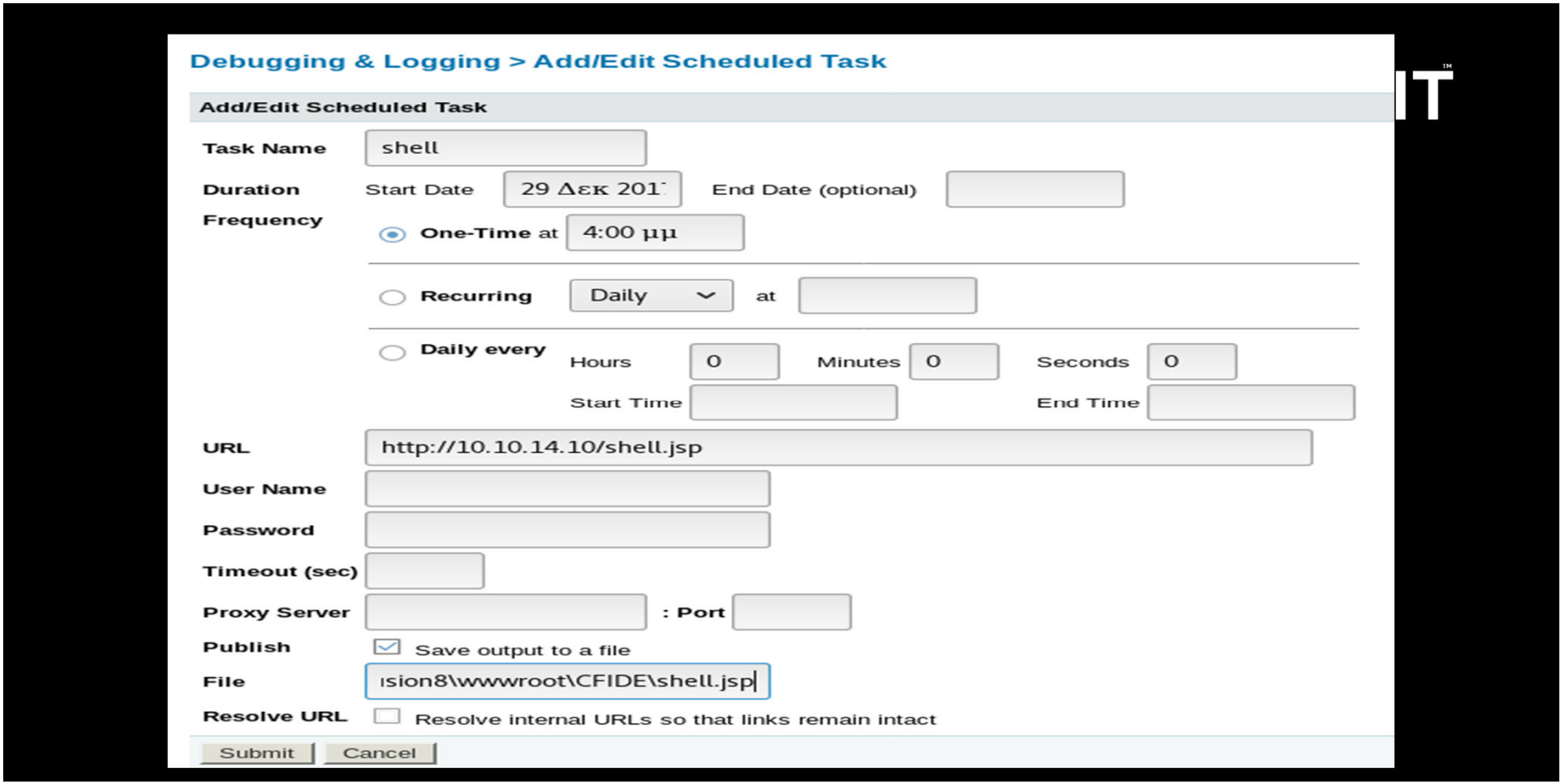
NorthIT

Active ColdFusion Mappings

Actions	Logical Path	Directory Path
	/CFIDE	C:\ColdFusion8\wwwroot\CFIDE
	/gateway	C:\ColdFusion8\gateway\cfc

RFI Example - Coldfusion

The Mappings directory, show us where we can save the remote file inclusion we will pull on the next slide, in the directory path.



RFI Example - Coldfusion

The task schedular, as seen above. Allows us to pull a file from a remote machine and save the file in the web root.

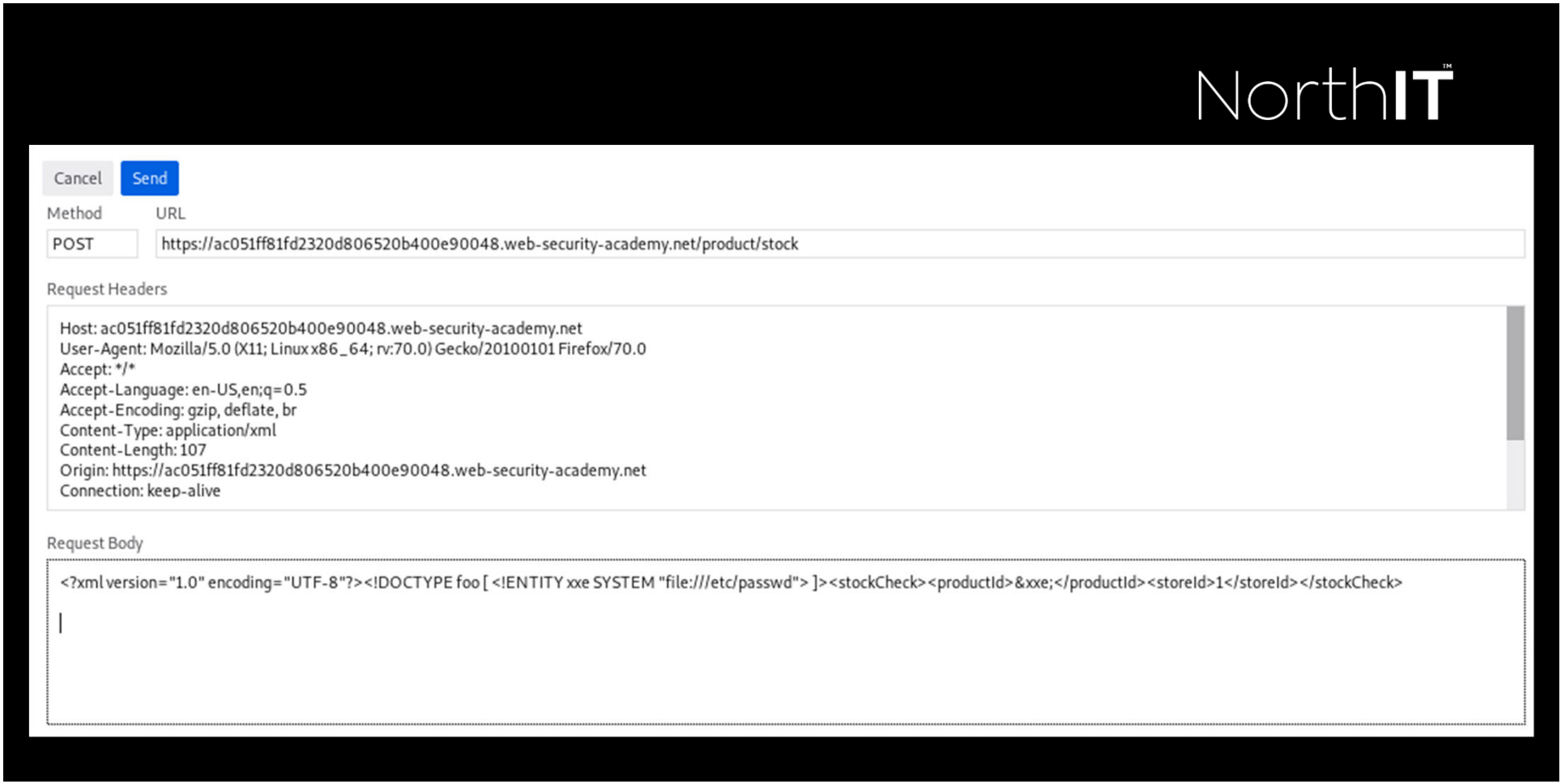
We could then execute the reverse shell from either the website root, or just via scheduling a task itself.

XXE Injection

NorthIT

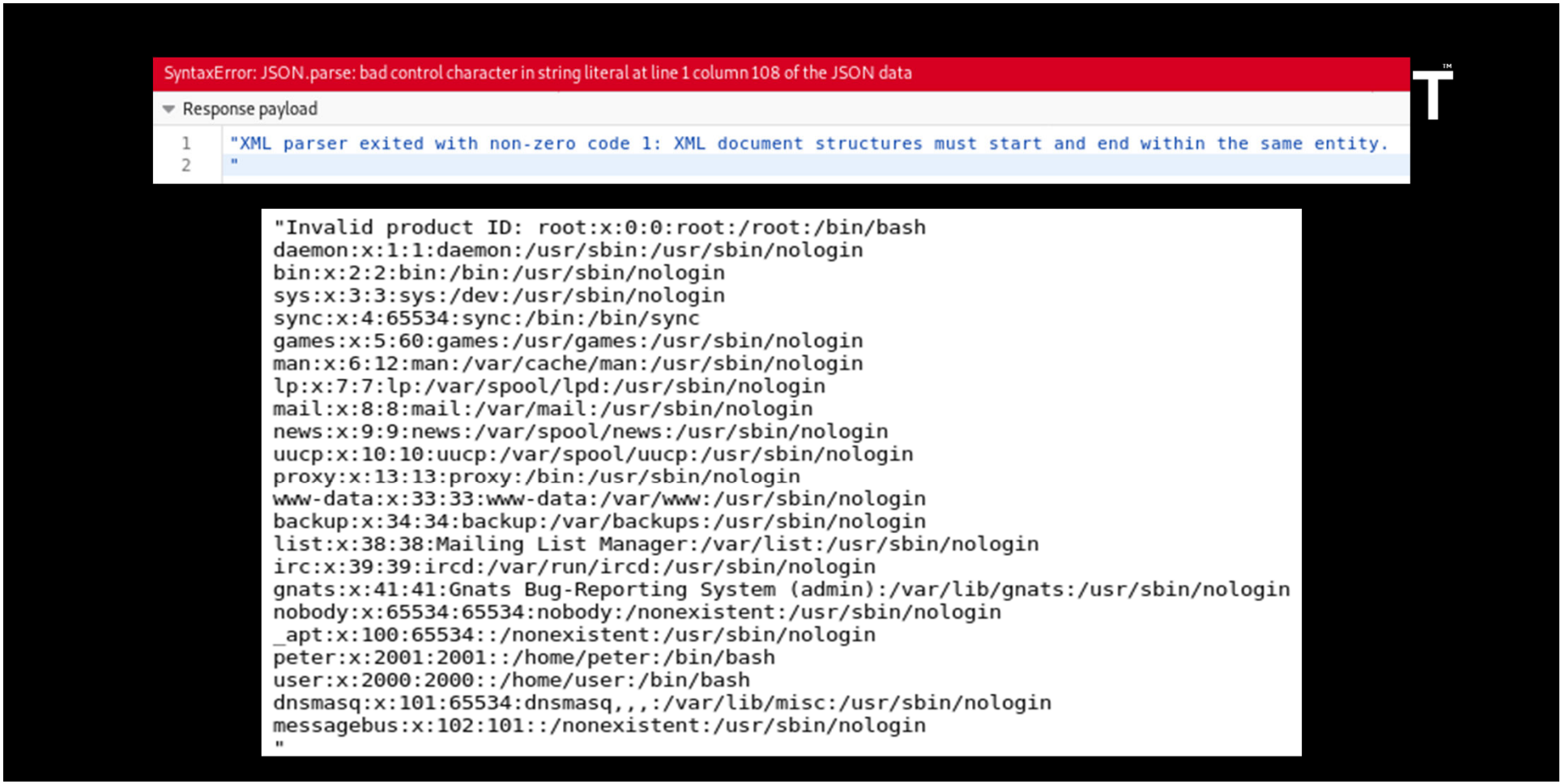
- XML external entity injection (also known as XXE) is a web security vulnerability that allows an attacker to interfere with an application's processing of XML data. It often allows an attacker to view files on the application server filesystem, and to interact with any backend or external systems that the application itself can access.

NorthIT



XXE Injection Demo - Payload

Here we inject an external XML entity titled ‘xxe’ which retrieves the system file ‘/etc/passwd’.



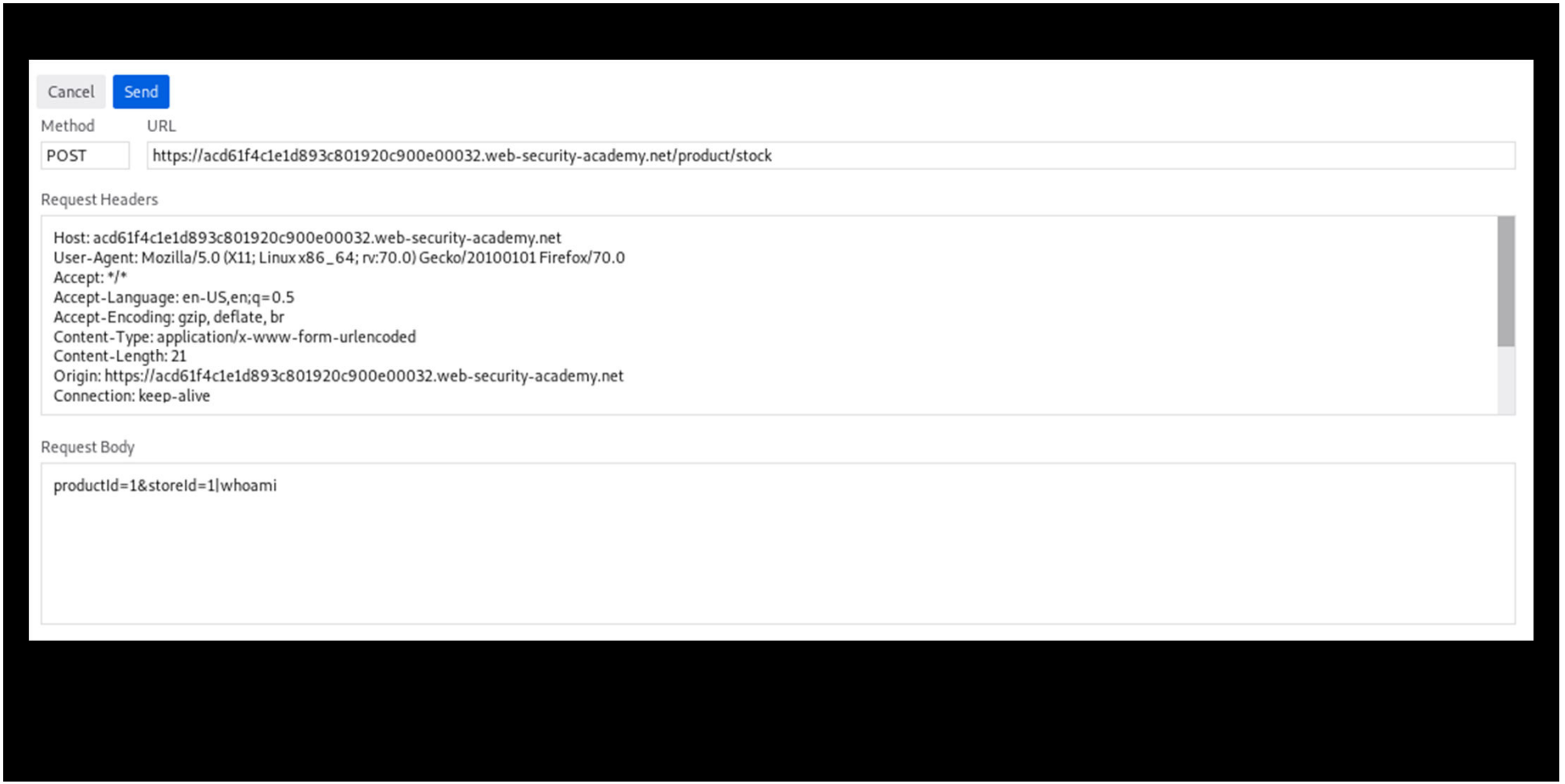
XXE Injection Demo - Result

The response generates an XML parsing error, which appears benign. However if we take a look at the page source, we can see it pulled in the contents of the file we requested (/etc/passwd)

OS Command Injection

NorthIT

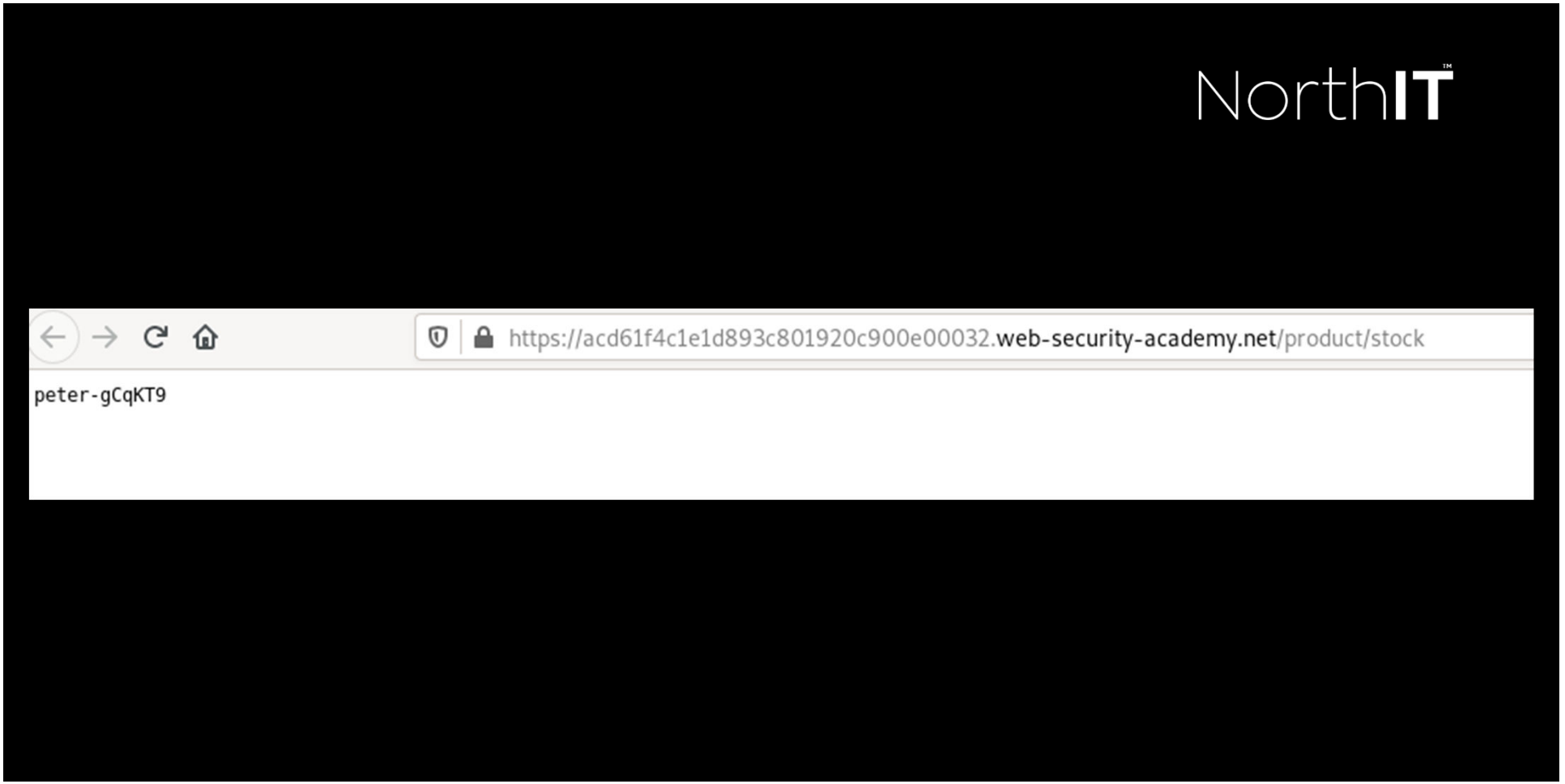
- OS command injection (also known as shell injection) is a web security vulnerability that allows an attacker to execute arbitrary operating system (OS) commands on the server that is running an application, and typically fully compromise the application and all its data.



OS Command Injection - Payload

This command outputs the stock status for the specified item, which is returned to the user.

Since the application implements no defenses against OS command injection, an attacker can submit the following input to execute an arbitrary command: ‘whoami’



OS Command Injection - Result

The username is returned to the screen. Generally speaking testers will only attempt something benign to begin with, simply to test the PoC. This could be any command in which the user is able to execute, including if they have access to delete files.

Business Logic Testing.

- Testing for business logic flaws requires thinking in unconventional methods.

File Uploads

NorthIT

- What files can be uploaded across the application?
  - Does the application stop you from uploading unexpected file-types?.
  - Are you incorporating a whitelist system? As opposed to controlling a blacklist?.
  - SVG Images? (XML Injection).