CYBER PATH™
POLICE & ACADEMIA
TALENT HORIZONS

THE
CYBER
RESILIENCE
CENTRE
FOR THE SOUTH EAST

Sample Company

Services Provided: Web App Assessment

Created For: Jane Doe - Owner

Report Submitted: 06/03/2024

Student Assessor: Jack Nimble

NATIONAL
CYBER
RESILIENCE
CENTRE
GROUP

Contents

Your Assessment

Thank you for entrusting our team at the South East Cyber Resilience Centre to help you. Our unique partnership between Policing, Business and Academia strives to support organisations like yours on their journey to Cyber Resilience. We recognise that this may be the first time you have considered contracting a cyber service and you might be unsure what to expect or how to act on these findings. Our team are dedicated to making the process as simple and transparent as possible, to help you understand the risks highlighted in this report and how to improve on them. Please raise any questions at all with us. We are here to help you learn as much as possible – there are no silly questions here!

Assessment Information	
Service completed	Sample.net
Assessment Completed By	Jack
Date Completed	03/03/2024
Overseen By	Abdullah Khan - Cyber PATH Student Supervisor

Executive Summary

This executive summary intends to present the potential key risks that our team collated from the recent assessment service that we were entrusted to deliver for you. This report is presented in two sections.

- Executive Summary / Risk Summary – included for your risk-focused senior stakeholders who may not have a technical background.

- Technical Detail – an incredibly detailed and informative capture of all our teams work and output. This section is intended for your technical audience who may be involved in the mitigation.

On this engagement, we delivered one of our services to your organisation. With the scope of this service, sample.net, agreed with your organisation in advance of our engagement.

Web Application Testing

Our team assessed key components, configurations, and software that are key to your website on this service delivery using the OWASP methodology. The output from this assessment noted that:

Outdated & vulnerable components - Our team identified versions of PHP 8.2.8 and jQuery 2.1.3 that were confirmed as having known vulnerabilities – expressed in the technical report using their Common Vulnerabilities or Exposures identifiers (CVEs). In this case, the identified CVE for PHP allows for memory corruption or remote code execution – practically allowing a malicious user to run commands or scripts on the web server. The CVEs identified for jQuery allow for a malicious user to exploit the way that the application runs, to the effect of cross-site scripting, as well as the execution of untrusted code in some circumstances.

Reflected Cross Site Scripting (XSS) injection - Reflected XSS allows an attacker to inject malicious scripts/code into web pages. This can result in a person with malicious intent stealing sensitive information from legitimate users such as login credentials or session cookies. The assessment team found the site to be vulnerable to reflected XSS injection which is detailed in the A03:Injection section of the technical report.

Session tokens - A session token is used to identify and authenticate users after they have logged in. This prevents a user from the need to enter their credentials as they move through the site continuously. Tokens should be private to the user and live for a short time, until they log out or close the browser. If session tokens are not properly protected a person with bad intent can use the tokens to access other user's accounts potentially. The sites' lack of protection on the session tokens meant that the assessment team could log in as a coordinator account from a basic user account in the event that they intercepted a session token for the coordinator. Practically, this may facilitate a low privileged user to gain access to a higher privilege account. Further details on this may be found in section A01:Broken Access Control.

The website owner made several changes to the site and its security measures during the assessment. The tests conducted during the assessment revealed vulnerabilities which were then rectified by the owner, who implemented additional protections during the assessment period in response to testing traffic. While these measures helped reduce attempts to brute-force directories and login credentials, the security posture of the site at a given snapshot in time was difficult to evidence as vulnerabilities may have been treated by the client, which have been evidenced in the report.

Thank You

We are truly grateful to have been entrusted in providing this service to you. Our team's focus is on presenting risk as a basis for further internal discussion. We hope that the following risk position and subsequent detailed technical report will provide a clear position on our findings. We are, of course, here at your convenience should you or your team have further questions or wish to explore the points we have raised in our work for you.

Risk Summary

The following table presents a summary of the risks identified throughout the assessment. It can be interpreted based on the colour and order of information. red = important, yellow=attention required, blue= will likely not pose an ongoing threat but is worth your attention in determining if further action is required.

We would encourage you to treat all the risks and prioritise them according to the table below. However, you may choose not to treat any of these weaknesses and accept the risks; this is an informed decision for your risk owners.

Summary of High-Risk Findings	
1	**Outdated & Vulnerable components.** As time passes, new vulnerabilities within software components are identified. This can be by researchers or potentially those with malicious intent. These vulnerabilities are fixed and released to users by means of software updates/patches. Software running on the site was discovered to be running versions which are unpatched and outdated. The version of PHP in use by the application contained a critical vulnerability that could allow a malicious user to run commands and scripts against the application. Additionally, a vulnerable jQuery JavaScript library was identified in use by the application. This version was identified as 'End of Life', which means it is no longer receiving security updates from its manufacturer.
2	**Reflected Cross Site Scripting (XSS) injection** Reflected XSS allows an attacker to inject malicious JavaScript code into the application. This injection is entered in the URL and is reflected to the user through their browser, hence reflected in the XSS injection name. The script can be used to steal sensitive information such as cookies or login credentials among conducting actions. Reflected XSS was found on the website once a user has logged in and was identified as affecting a parameter on the website. This provides an opportunity for a malicious user to send a crafted URL to a valid user and recover information that can be used in conjunction with the improper handling of session tokens to escalate privileges on the website or access other user sessions.
3	**Improper handling of session tokens** A session token is used to identify and authenticate users after they have logged in. If tokens are not properly managed or protected, attackers may manipulate, forge, or steal these tokens to gain access to privileged accounts. By doing this an attacker bypasses access controls, gain higher privileges and potentially compromise sensitive data or perform malicious actions. The site does not protect, hide, or change a user's token from being a visitor to being logged in. The token is kept the same, even if the user were to log out and come back to the

	site. This can lead to a weakness in the session management mechanism, which in turn allows a malicious user to gain unauthorised access to higher levels of privileged accounts.

Summary of Moderate-Risk Findings	
4	**Missing Security Headers** Headers are elective security directives that provide an additional layer of protection above the basic security measures built within web browsers. Recommended security headers were identified as not being enabled in responses from the application. As a result, it may be possible for a suitably placed malicious user view sensitive user information disclosed by the application, conduct attacks such as cross-site scripting (XSS) against users as well as potentially downgrade the encryption used by the application.
5	**Implement proper use of headers** The "X-Forwarded-For" header is a standard HTTP header that is used to identify the originating address of a client connecting to a web server. By allowing your server to accept this header it can allow a malicious user to bypass the application lock out mechanism. This could allow the attacker to brute force the site continuously.

Summary of Low-Risk Findings	
6	**Insecure Cookie Attributes** A cookie is a small piece of information sent from a website and stored on a user's computer by their web browser while they are browsing. Cookies are used to remember information about the user, such as login details or the contents of a shopping basket for example. The risk with an insecure cookie is that it can be intercepted by attackers, potentially leading to session hijacking, sensitive information disclosure or data breaches.

Sample.net - Reconnaissance

Before any vulnerability testing of the site was attempted, a reconnaissance stage was conducted. This is vital during any real cyber-attack as this provides an interested threat actor with the information, they need to mount an informed attack on the websites.

Domain Information

The site 'sample.net' domain has been purchased/leased through the domain registrar Amazon. A domain registrar is a person or entity who helps you to buy and register a domain for a certain length of time. The length of time you have the domain for is shown in the expiry section (Figure 1). The domain was purchased on the 02/Jun/2023 and the expiry of the domain is on the 02/Jun/2024. You will be offered a renewal option near the expiry date.

Figure 1: Showing the domain information for sample.net

Mail Protection

The mail provider for the domain 'sample.net' was determined to be Amazon and at the time of testing, there was a valid DMARC record, and DMARC policy were enabled for this domain (Figure 2).

Figure 2: Showing DMARC record and policy information for sample.net

DMARC is used as another layer of security for email protection above *SPF* and *DKIM*. If the email being sent to a client does not pass SPF or DKIM, certain actions can be taken against that email using the DMARC rules. These actions can be to monitor the email and report to the domain holder, quarantine the email (send to junk mail) or reject the email meaning the recipient will never receive the mail. As shown above, a DMARC record was found for the domain and meaning the extra security layer is present along with the actions of quarantining and rejecting the email in case the email fails DMARC checks.

HTTPS/SSL

Secure Socket Layer (SSL) is enabled for the site. The SSL certificate is valid from the 22/11/2023 to the 21/11/2024 as highlighted in Figure 3.

Figure 3: Issuer and Validity of SSL for sample.net

The SSL certificate is issued by Sectigo and has an A rating, as shown in Figure 4, indicating that the cryptography of your website is verified as working. A+ is the most desirable grade, however A is still good with satisfactory security. This rating is in part due to the configuration of the site to accept only TLS 1.2 as shown in Figure 5.

Figure 4: SSL report for sample.net

Figure 5: TLS cipher suites being used

Directory Discovery

Even though the site uses services to prevent directory enumeration it was possible to identify an interesting directory on the website (xxx.xxx). This directory could be accessed by a user who is not authenticated, and the page displayed a video of how to use the site when logged in as a user, shown in Figure 6. This allows an attacker to gather valuable information about the website or hone their exploits against the internal site. The directory has since been removed by the client.

Figure 6: Directory found showing video of how to use the site

Website Information

The website, www.sample.net, is built with PHP 8.2.8, which is the server-side programming language. The site makes use of JavaScript libraries, jsDelivr & CDNJS. JavaScript libraries is a collection of pre-written JavaScript code that allows for easier development of JavaScript-based applications. The site runs MySQL as the backend database. The email security to the site is provided by Amazon and the web server for the site is Microsoft IIS 10.0 (Figure 7).

Figure 7: W3techs results for sample.net

Methodology Applied

OWASP Top 10

The framework for this assessment is the OWASP (Open Web Application Security Project) web security testing guide, and the OWASP Top 10. By using the OWASP guide and Top 10, we cover a comprehensive range of web security aspects, including vulnerability identification, risk assessment, and mitigation recommendations. The OWASP Top 10 is a widely recognised and influential list of the ten most critical web application security risks. It serves as a guideline for developers, security professionals, and organisations to prioritise their efforts in securing web applications. The list is updated periodically to reflect emerging threats and vulnerabilities in the digital landscape.

The table below shows how the site is mapped to the OWASP Top 10, where the X marker identifies a vulnerability within that category and a dash (–) identifying no vulnerability found. Three controls (A08, A09, A10) within the Top 10 are not relevant for this assessment, as they are focus on the developmental aspects of website design and not the security assessment of a website.

Each of the OWASP Top 10 is listed below with a brief explanation.

1. **Broken Access Control:** Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit this to access unauthorised functionality and/or data, such as accessing other user accounts, viewing sensitive files, and modifying other users' data.
2. **Cryptographic Failures:** This results in sensitive information being inadvertently exposed by the web application. It can be caused by weak configuration of the application, missing or weak encryption allowing attackers to intercept data, storing private data in a public location, etc.
3. **Injection:** Malicious code is inserted into the communications between a website and database, or user, to perform unauthorised actions such as accessing or manipulating sensitive data and bypassing website restrictions.
4. **Insecure Design:** Referring to controls and systems that are insecure at the design phase, rather than at implementation. Due to this, insecure designs cannot be fixed by secure implementations as the design did not factor in the necessary security controls.
5. **Security Misconfiguration:** This is commonly a result of default configurations that are insecure, incomplete configurations or ones built for a specific purpose, open cloud storage, misconfigured responses from websites, and detailed error messages containing sensitive information.
6. **Vulnerable and Outdated Components:** Vulnerabilities are regularly found in third-party components used in websites. Using components with known vulnerabilities may undermine defences and enable various attacks and impacts.
7. **Identification and Authentication Failures:** Functions related to these are often implemented incorrectly. Incorrect implementation allows attackers to compromise passwords and *session*s, or to assume the identity of another user temporarily or permanently.
8. **Software and Data Integrity Failures:** Relying on components from an untrusted source risks unauthorised access to be gained, or malicious code to be implemented on the site. Automatic software updates without proper verification risk malicious data being downloaded and applied.
9. **Security Logging and Monitoring Failures:** Logging and monitoring failures allow attackers to further attack systems, maintain unauthorised access, move to attack more systems, and tamper, extract, or destroy data.
10. **Server-Side Request Forgery (SSRF):** Attackers take advantage of this vulnerability by forcing a web application to make requests to a web address chosen by the attacker. This can be used to connect to internal services, or to connect to external services and potentially leak sensitive data.

Table 1: OWASP Top 10 results for sample.net

OWASP Top - 2021	Sample.net
A01: Broken Access Control	X
A02: Cryptographic Failures	X
A03: Injection	X
A04: Insecure Design	X
A05: Security Misconfiguration	X
A06: Vulnerable and Outdated Components	X
A07: Identification and Authentication Failures	X
A08: Software and Data Integrity Failures	n/a
A09: Security Logging and Monitoring Failures	-
A10: Server-Side Request Forgery	n/a

A01: Broken Access Control

Broken access control is a security flaw that allows unauthorised users to access restricted resources. A malicious user can access sensitive data about the systems, users, or other vital assets. The attacker can create, modify, or delete the data causing further damage to the organisation's reputation and business.

When testing for broken access controls, it is critical to check if a non-authenticated user can access the website's administrators page or other login pages they are not authorised to view. While scanning for these different login pages, test pages used for development can be searched to ensure they are removed from the directory. Further testing includes searching for *authentication* pages or pop-up dialogue boxes asking for user credentials, as unauthorised access is a possible weakness. A final test to conduct for broken access is directory traversal attacks, aiming to access directories and files containing sensitive information.

The site has instances of broken access control which is detailed below:

Session Management & Access Control

Session management is the process of securely managing and maintaining user sessions within a web application, so a user does not need to repeatably need to enter credentials on each request to the server to use the site. A session is a temporary interaction between a user and a system, typically initiated when the user logs in and ends when the user logs out or after a period of inactivity.

Access control is a security measure that regulates who or what can view or use resources in the application. It is a fundamental aspect of information security that aims to protect sensitive information and resources from unauthorised access, modification, or misuse.

Upon visiting the site, the visitor is given a cookie (PHPSESSID), and when the visitor logs into the platform with valid credentials that cookie becomes their session id without any adjustments (Session Management).

The PHPSESSID cookie issued by the application stays the same even if a user:

- Opens different instances of the browser
- Closes the browser and reopens it
- Logs out of the application and re-logs in

Using this flaw, it was possible to do lateral and vertical movement within the site as an authenticated user (Access Control).

Lateral Movement

Lateral movement is where a user can log into another user's account with the same permissions level. Student19 and Student20 where two accounts which were given the permission of only "User" privileges. The two figures below, Figure 8 & Figure 9 , show student20 logged in (point 1 in Figure 8) with a cookie value (point 2 in Figure 8) and on the other instance student19 being logged in (point 1 in Figure 9) with its own cookie value (point 2 in Figure 9). The cookie values have been deliberately changed to be unique and different in this test to prove it is exploitable.

Figure 8: Student20 logged into platform with unique cookie value

Figure 9: Student19 logged into platform with unique cookie value

To exploit the session, student19's cookie is inserted into student20's account and refreshed. As shown in Figure 10 the account has changed from student20 to student19. The user now has access to student19's account details, and can view their personally identifiable information

Figure 10: Session takeover of student19

Vertical Movement

Vertical movement within a site is where a user can gain access to an account which has higher privileges to what the current user is authenticated for. In this case the 'Co-ordinators' account are privilege user accounts and 'User' are lower privileged accounts. In this test vertical movement will happen from a 'User' account to a 'Co-ordinators' account. To prove the concept the cookies are cleared to make them unique and different.

Below shows student19's details and cookie value (Figure 11). The Co-ordinator account (Figure 12) shows the user is "XX"(1), the cookie value(2), and the extended settings menu privileged to only coordinator accounts(3).

Figure 11: Student19 logged into platform with unique cookie value

Figure 12: Abdullah Khan logged into platform with unique cookie value

Similarly to before the cookie value is replaced within "student19" account with the value of the coordinator account, "XX". Figure 13 shows that the co-ordinators account can now be accessed with the extended menu.

Figure 13: Vertical movement compromise

In this instance, having a static cookie makes the site vulnerable to session hijacking when exploited in conjunction with XSS, and can cause privilege escalation for malicious actors. Additionally, as these cookies do not have flags configured for them, they are vulnerable to client-side script which could access the cookies and return them to an attacker. More information on the cookie flags can be seen in section A05 Security Misconfiguration.

A02: Cryptographic Failures

Cryptographic Failures are when data is not passed securely from one endpoint to another, and when the stored data within a website is not encrypted correctly. The data sent must be encrypted so that if the data were to be intercepted by a malicious user, they would not be able to decrypt it to access sensitive information such as user credentials or cookies. Another example of cryptographic failures is HTTP headers that do not use appropriate security mechanisms and have too much information about the systems available.

When testing for cryptographic failures, checks are done to ensure that data transported to and from the server is secure and encrypted such as reviewing if data entered by a client is delivered using strong encryption and not plain text. Another check is to see if cookie randomness is high or not, otherwise malicious users will be able to guess values and log in as other users. Furthermore, checking the *ciphers* used and the header data given as a response to requests made.

Below is the informational disclosure given out via the response headers of the site.

Information Disclosure via Response Headers

A response header is part of the response sent by a server to a client in response to an HTTP request. HTTP is the protocol used for communication between web browsers and the servers. When a user/web browser sends a request to a server, the server processes the request and sends back a response. This response typically includes a body containing the requested contents (HTML to display the site) and response headers providing metadata about the response.

When headers are configured insecurely, response headers leak information about backend technology which is used in the application, as shown below in Figure 14. As we can see that the server is Microsoft's ISS version 10.0, and the framework is PHP version 8.2.8.

Figure 14: Response header information

A03: Injection

Injection attacks can be conducted in a variety of different methods, the most common being SQL Injection. Injection attacks are used to access unauthorised information such as user credentials, company data and other sensitive information. Injection attacks occur because the data passed in is not validated before being sent to the server. An attacker can send malicious code, which is then executed, helping to retrieve classified information.

To test injection attacks within a website, malicious data is passed into data fields such as search boxes, comments sections and signing-up areas. The target is to access sensitive data from within a server such as user credentials, directories, or other vital files. An injection attack consists of attacks such as *SQL Injection*, *OS Command Injection* and *Cross Site Scripting.*

Cross-Site Scripting (XSS)

XSS is a security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can then perform unauthorised actions on behalf of the users, such as stealing session cookies, redirecting to malicious websites, or modifying the content of the web page. The types of XSS and their characteristics include:

- Reflected XSS happens when a website sends back user input (like in an error message or search result) without checking if it is safe. This can make the user's browser do things it should not, like showing harmful content. It often involves a user clicking a link with the malicious code in it.

- Stored XSS occurs when user input (like in forums or comments) is saved on a website and then shown to others without checking for safety. This can be dangerous because the harmful content gets saved and can affect many users.

- DOM Based XSS is when the attack changes the website's setup in the user's browser. This does not alter the actual website but makes the browser run harmful scripts. It is tricky because the website itself does not change; the browser just acts differently.

The site was found to be vulnerable to Reflected XSS. As a proof of concept to prove the vulnerability being present on the site, the below script was crafted. This trivial script would send out an alert message of the number 1 to be displayed on the screen.

It was possible to trigger the XSS using the following string to generate an alert box:

```
<script>alert(1)</script>
```

The test script was injected into the URL of the site in an encoded format. An example of the script which was used to test for reflected XSS is:

```
https://www.sample.net/frontend/templates/master.php?page=f6qxl</title><script>a
lert(1)</script>j3piv
```

As a result, a popup message appeared, showing that the site is vulnerable to reflected XSS (Figure 15).

Figure 15: Testing for XSS vulnerability on sample.net

In conclusion the website is vulnerable to an injection attack. An attacker could use this vulnerability to gain a user's cookie using a specially crafted URL. This would then allow them to conduct additional attacks against the application based on the vulnerabilities identified in section A01 Broken Access Control.

It should be noted that a vast number of the automated testing was prevented by lockout mechanisms which were implemented on the site during the assessment period.

A04: Insecure Design

Insecure design is when a website has not considered security while it was being built and therefore there are many different issues such as no validation within the code written, inadequate use of authentication or generally a lack of security controls. This issue can be critical depending on which security measures are taken and implemented.

When testing for insecure design, the focus is on the authentication processes used, the different use of validation and if they can be bypassed, and finally identifying legitimate and illegitimate users from the traffic sent to the website.

Below shows the vulnerability found within this section of insecure design for the website.

Username enumeration via response timing

Enumerating the users of the site is made possible through the timing of the response from the server. A malicious actor can use this process to identify valid usernames for the website then use this information in a brute force attack with a password wordlist to compromise a user's account.

As proof of this Figure 16 shows a request for an invalid user, student20@nationalcrcgroup, and the response shows the generic error message of invalid email and/or password. Looking at the bottom right we can see the response timing is 73 milliseconds.

This was repeated with another invalid user with a similar name to the valid user (Figure 17), student19@nationalcrcgroup.co.ukm, and the response shows the generic error message. The response time for this is also 73 milliseconds.

Figure 16: Invalid user account request/response with 73 millisecond response time

Figure 17: Invalid user account request/response with 73 millisecond response time

The request was repeated but a valid user was used, student20@nationalcrcgroup.co.uk, and the response shows the generic error message. Though this time the response time was 149 milliseconds (Figure 18). This was tested with the other valid user account student19@nationalcrcgroup.co.uk with a response time of 133 milliseconds (Figure 19).

Figure 18: Valid user account request/response with 149 millisecond response time

Figure 19: Valid user account request/response with 133 millisecond response time

If a server takes longer to respond to a valid user, it could indicate that the user account is valid and can be enumerated. This could potentially lead to various attacks by a person with bad intent who can gain access to a valid username.

A05 Security Misconfiguration

Security misconfiguration is when the website is not configured correctly or not implemented at all. Another security misconfiguration is that if malicious or wrong data is entered into the data fields, it should not return information about the system that is running or an error code that outputs information that can allow attackers to gain more insight into the restrictions in place.

Below are the specific vulnerabilities found within the security misconfigurations section for the website.

Security Headers

Security headers provide critical instructions to browsers for secure content handling across a website. They are essential in mitigating cyber-attack risks and ensuring secure communication between the user's browser and the website. The absence or improper configuration of these headers can significantly increase the site's vulnerability to cyber threats.

Table 2 below shows the various security headers and a description of what the headers are used for. Table 3 show the various security headers and if they are implemented(green) or missing(red). It should be noted that security header may be enabled but third-party protection can block detailed scans of the website and result in this information being returned as a 'not present' value. The results should be manually checked by the developer to confirm the presence or absence of the security headers.

Figure 20: Nmap results for the target.

Figure 21: Security headers output.

Table 2: Missing Security Headers and descriptions

Name Of Header	**Description**
X-Frame-Options	Prevents website from being displayed in a frame or frame. This helps avoid 'clickjacking' attacks, where users are tricked into clicking something different than what they think they are clicking.
Strict-Transport-Security (HSTS)	Enforces secure (HTTPS) connections to the server. This helps prevent attacks like protocol downgrading and cookie hijacking.
Content-Security-Policy (CSP)	Controls what resources the browser is allowed to load for a page. It is useful for preventing XSS (A03: Injection) and data injection attacks.
X-Content-Type-Options	Prevents the browser from making assumptions about the type of content it is loading, which could potentially lead to security issues. It is vital to ensure that the browser handles only the types of data it is supposed to, thereby enhancing security.
Referrer-Policy	Controls how much information about where a web request comes from (like the webpage's address) is shared. It helps keep private details that might be in the webpage's address from being accidentally shared.

Permissions-Policy	Enables a website to turn on or off specific features of your browser for safety reasons. For example, it can decide whether a site can use your camera or microphone.

Table 3: Implementation/missing security headers for sample.net

Sites	X-Frame-Options	HSTS	CSP	X-Content-Type-Options	Referrer-Policy	Permissions-Policy
Sample.net						

Insecure Cookie Flag

A cookie is a small piece of data sent from a website and stored on a user's computer by their web browser while they are browsing. Cookies are used to remember information about the user, such as login details or the contents of a shopping basket for example.

An insecure cookie flag means that the cookie can be transmitted over non-HTTPS (unencrypted) connections, which is a security risk. The problem with an insecure cookie is that it can be intercepted by attackers, potentially leading to session hijacking or data breaches.

For a secure cookie, the "PHPSESSID" should have the 'Secure' flag set, ensuring it is only sent over HTTPS connections. Figure 22 shows the current insecure state of the user session cookie for URL.

When performing authentication testing the username and password were entered in the login page sample.net. The credentials along with the PHP session ID are visible in cleartext. This is represented in the figure.

Figure 22: PHPSESSID Cookie for URL.

The HttpOnly flag is also not set for the PHPSESSID. The HttpOnly flag prevents the cookie value from being read or set by client-side JavaScript. This makes certain client-side attacks, such as cross-site scripting (XSS), harder to exploit by preventing them from capturing the cookie’s value via an injected script.

Clickjacking

Clickjacking is a deceptive online technique designed to manipulate a user into clicking on a web page element without their knowledge or consent. This technique overlays an invisible element on a legitimate webpage, tricking users into interacting with content they did not intend to engage with. The risk to a user of the site is that they could unwittingly download malware, visit malicious pages, provide credentials or information.

A simple website was created to test for the ability to clickjack the sites. Figure 23 shows the simple HTML code to test for the clickjacking vulnerability and Figure 24 shows the successful clickjacking technique of the site. Using specific security headers (X-Frame-Options) could prevent this type of attack.

Figure 23: HTML code for clickjacking test

Figure 24: Successful Clickjacking test against Sample.net

A06 Vulnerable and Outdated Components

Outdated website components often have common vulnerabilities which are assigned a *CVE* (Common Vulnerabilities and Exposures) reference number. The *CVE* (vulnerability) is scored by a number created using a recognised vulnerability scoring system (*CVSS*), which refers to the impact and seriousness of the vulnerability on a scale of 0 - 10. The higher the number is, the more critical the vulnerability. All software and components should be updated and patched regularly to their latest version to avoid websites being open to attack from cybercriminals. Below shows the following software versions, which are outdated and vulnerable with relevant CVSS scores (Table 4).

It's important to note that these vulnerabilities have been identified based on the self-reported version number - which was compared against known and public exploits, and therefore may be dependent on the vulnerable functionality being used as part of the application.

Table 4: Showing vulnerable components for sample.net

Vulnerable Components			
Component	**CVE**	**CVSS (v3)**	**Explanation**
PHP 8.2.8	CVE-2023-3824	**9.8**	This vulnerability can be exploited by buffer overflow attacks which can lead to memory corruption or remote code execution.
jQuery 2.1.3	CVE-2015-9251	**6.1**	It is vulnerable to Cross-site Scripting (XSS) attacks when a cross-domain Ajax request is performed without the dataType option, causing text/javascript responses to be executed.
	CVE-2019-11358	**6.1**	It mishandles jQuery.extend(true, {}, ...) because of Object.prototype pollution. If an unsanitized source object contained an enumerable __proto__ property, it could extend the native Object.prototype.
	CVE-2020-11023	**6.1**	When passing HTML containing <option> elements from untrusted sources - even after sanitizing it - to one of jQuery's DOM manipulation methods (i.e. .html(), .append(), and others) may execute untrusted code.
	CVE-2020-11022	**6.1**	Passing HTML from untrusted sources - even after sanitizing it - to one of jQuery's DOM manipulation methods (i.e. .html(), .append(), and others) may execute untrusted code.

A07 Identification and Authentication Failures

This category refers to security weaknesses in how an application handles user identity and authentication. This includes issues like weak password policies, flawed session management, and inadequate multifactor authentication, which can allow attackers to impersonate legitimate users, gain unauthorised access, or hijack user sessions. These failures can lead to breaches of user accounts and sensitive data.

Below are the specific vulnerabilities found within the identification and authentication failure section for the site.

Section A01:Broken Access Control discusses the flaw within the sessionid allowing for users to hijack user sessions and increase their privileges laterally and vertically.

Bypassing IP Address Block

The site has a protection in place to prevent brute forcing of user accounts. After 10 incorrect tries of logging in the user's IP address will be blocked by the application as shown in Figure 25. The figure shows the request to access the "student19" account and the response of the IP address being blocked.

Figure 25: Student19 account has IP blocked

This could be circumvented by adding the "X-Forwarded-For" flag into the request allowing our IP address to change and allow us to continue to brute force a user's account. In the below examples we used the header to specify another IP which would allow us to bypass the block and access our account as shown in the figures below (Figure 26 & Figure 27).

Figure 26: Request with X-Forwarded-For header and student19 creds

Figure 27: IP Block bypassed allowing access to the application

A08 Software and Data Integrity Failures

Software and data integrity issues are caused by unprotected code and infrastructure. A case in point is when an application makes use of plugins, libraries, or modules obtained from untrusted sources, repositories, or content delivery networks (CDNs). A non-secure continuous integration/continuous delivery pipeline introduces the possibility of unwanted access, malicious code, or system compromise. Many pieces of software now have auto-update capabilities, in which updates are obtained and deployed to previously trusted applications without appropriate integrity checking, this can present the opportunity for an attacker to submit their own updates to be disseminated and run on all systems. It was not possible to check if the website was vulnerable to this.

The website utilises multiple third-party JavaScript libraries, plugins, and fonts. Ensure that all this software and data components are from the expected source by using digital signatures and other mechanisms before updating or installing new software. Be careful of automatic updates, as a small change in an update containing malicious code could compromise systems. Below highlights the 4 externally hosted JavaScript files on Cloudflare. By hosting the files via the third party you are reliant on the third party to keep these scripts up to date. If the third party is compromised or a malicious user is able to manipulate a user's traffic it may be possible for a malicious user to change these files to include malicious code without notification. This code could then be used to target legitimate users of the application. Further information can be found at this link.

A09 Security Logging and Monitoring Failures

Logging and monitoring are one of the most vital parts of protecting your organisation and preventing cyber-attacks. It will allow you to recognise activity patterns on your website and see possible indicators of compromise. It will also allow you to potentially identify the source and see the extent of the damage caused by the attack. All this information will prove vital for damage assessment as an organisation but also when informing the Information Commissioner's Office (ICO) of your breach.

The client has monitoring enabled, which was identified during the assessment. The monitoring capability notified the client of the assessment's teams brute forcing component of the testing phase. This allowed the client to implement further protection mechanisms on the site several times.

A10 Server-Side Request Forgery (SSRF)

Server-Side Request Forgery (SSRF) is an exploit in which an attacker exploits the functionality of a server. This is to change or gain access to data stored within that server's domain that would otherwise not be accessible. This vulnerability can occur when an attacker can manipulate the requests that a server makes to other resources on the internet. This could allow them to interact with the resources that they should not have access to, such as internal systems or services that are meant to be restricted. As the server does not make request to other resources on the internet this was not applicable.

There's more to the South East Cyber Resilience Centre…

There are many additional ways to engage with SECRC. If you haven't already, you can register as a Core Member of the South East community, which is free of charge. This membership includes practical, government-approved guidance, as well as regular information updates to keep you informed of our other help and services on offer, including:

- **Educational Events** - we run regular webinars and events on a range of topics relevant to your small business or third sector organisation
- **Affordable solutions** - we offer a range of paid services like this one which are designed to address the most pertinent risks affecting SMEs, as identified by policing and Government.
- **Cyber Essentials Certification** - we have a Cyber Essentials Partners forum, which is made up of IASME approved Cyber Essentials Certifiers. If you are planning on achieving Cyber Essentials or Cyber Essentials Plus certification, we can refer you to the Cyber Essentials Partners forum of local suppliers in the region that provide this.

About the Cyber Resilience Centre Network

The National Cyber Resilience Centre Group and Cyber Resilience Centres are funded and supported by the Home Office and policing in a not-for-profit partnership with the private sector and academia to strengthen our national cyber resilience across SMEs and the supply chain.

At a national level, NCRCG is building a coalition of police, government, large employers and organisations, and academia to ensure a collaborative and coherent approach to cyber resilience.

NCRCG and its National Ambassadors and the CRC network are committed to investing in the next generation of cyber experts. As such, NCRCG has launched Cyber PATH in partnership with the CRC network and over 45 universities.

The nine CRCs operate across England and Wales. They serve SMEs in their locality helping to build cyber resilience against threats that are specific to them. Cyber PATH empowers students to work with their regional CRC in meeting the requests brought to them by local businesses.

Each CRC retains the freedoms to deliver tailored, trusted, and affordable support, with NCRCG providing insight and solutions at a macro level.

You can learn more about the work of the NCRCG here. We can help your own customers and suppliers too, so spread the word.

Help Bundles and Additional Services

If you are ready for more support, we offer free Core membership and a selection of Help Bundles which include paid services and 12 months of support. This includes:

- Full website vulnerability testing
- Policy Review
- Staff awareness training
- And more....

Visit our selection of Help Bundles and Services here.
You can also engage with us through our social media channels – find us on LinkedIn, Facebook and Twitter.
Contact us at enquiries@secrc.co.uk if there's anything else we can help you with.