



Politechnika Wrocławska

Raport z audytu serwisu

OWASP BWAPP

Przedmiot prac

Testy penetracyjne wybranych podatności
aplikacji w wersji webowej

Data wykonania prac

24.01.2020-27.01.2020

Na potrzeby

ZIT - Projekt

Spis treści

1. Podsumowanie prac	3
2. Klasyfikacja podatności	4
a. KRYTYCZNA	4
b. WYSOKA	4
c. INFORMACYJNE	4
3. Metodyka dokumentowania podatności	5
4. Znalezione podatności	6
a. Typu SQL Injection	6
i. [SQL][CRITICAL]OWASP_LOGIN_01	6
ii. [SQL][CRITICAL]OWASP_USER_LOOKUP_1	7
iii. [SQL][CRITICAL]OWASP_ADD_BLOG_1	10
iv. [SQL][CRITICAL]OWASP_REGISTER_1	11
b. Typu XSS	13
i. [XSS][CRITICAL]OWASP_DNS_LOOKUP_1	13
ii. [XSS][CRITICAL]OWASP_USER_LOOKUP_2	14
iii. [XSS][CRITICAL]OWASP_USER_LOOKUP_3	16
iv. [XSS][HIGH]OWASP_BACKGROUND_COLOR_1	17
v. [XSS][CRITICAL]OWASP_USER_POLL_1	19
vi. [XSS][CRITICAL]OWASP_REGISTER_2	20
vii. [XSS][CRITICAL]OWASP_PASSWORD_GENERATOR_1	22
viii. [XSS][CRITICAL]OWASP_REGISTER_3	24
ix. [XSS][CRITICAL]OWASP_ADD_BLOG_2	25
c. Typu CSRF	28
i. [CSRF][HIGH]OWASP_USER_POLL_2	28
ii. [CSRF][CRITICAL]OWASP_ADD_BLOG_3	30
iii. [CSRF][CRITICAL]OWASP_REGISTER_4	32
5. Zalecenia bezpieczeństwa	35
i. [INFO]OWASP_LOGIN_02	35
ii. [INFO]OWASP_ERROR_LOG_1	36
iii. [INFO]OWASP_REGISTER_5	36
iv. [INFO]OWASP_LOGIN_REQUIRED_1	38
v. [INFO]OWASP_PASSWORD_STORAGE_1	38
vi. [INFO]OWASP_PASSWORD_STORAGE_2	39
vii. [INFO]OWASP_CRSF_TOKEN_1	40
viii. [INFO]OWASP_HTTP_METHODS_1	41
ix. [INFO]OWASP_HTTP_METHODS_2	42

1. Podsumowanie prac

Przedmiotem prac było przeprowadzenie audytu serwisu OWASP BWAPP pod kątem bezpieczeństwa. Serwis został sprawdzony pod kątem następujących podatności:

- SQL Injection
- Cross-Site Scripting
- Cross-Site Request Forgery

Podczas audytu ww. elementów znaleziono szereg podatności krytycznych oraz wysokich. Oznacza to niemalże gwarantowane ryzyko wycieku danych bądź nieautoryzowanego dostępu do danych wrażliwych.

Znalezione błędy bezpieczeństwa oznaczone są konkretną kategorią zależną od sklasyfikowanej podatności z wyżej wymienionych.

Testy bezpieczeństwa przeprowadzono zgodnie z wybranymi metodykami testowania zalecanymi przez OWASP Top 10.

W ramach audytu wykorzystano narzędzie automatyczne Burp Suite.

2. Klasyfikacja podatności

Ze względu na stopień zagrożenia wynikający z błędów bezpieczeństwa znalezionych podczas audytu, zdecydowano o sklasyfikowaniu ich w trzystopniowej skali, ze względu na ryzyko skutków ich wykorzystania oraz szansy na ich znalezienie.

a. KRYTYCZNA

Wykorzystanie tej podatności umożliwia przejęcie pełnej kontroli (w tym RCE) nad serwerem i/lub umożliwia dostęp do danych o dużym poziomie poufności i istotności. Podatności oznaczone tą kategorią powinny zostać bezzwłocznie naprawione, ponieważ dodatkowo nie wymagają od atakującego podjęcia szeregu czynności do uzyskania autoryzacji i/lub nie wymagają przeprowadzenia ataku socjotechnicznego.

b. WYSOKA

Wykorzystanie tej podatności umożliwia dostęp do danych o dużym poziomie poufności i istotności, jednak w celu ich praktycznego wykorzystania może być wcześniej wymagane spełnienie pewnych warunków (np. posiadanie konta użytkownika).

c. INFORMACYJNE

Ta kategoria służy do zwrócenia uwagi na dobre praktyki, których zastosowanie może zwiększyć ogólny poziom bezpieczeństwa serwisu. Punkty oznaczone tą kategorią nie są podatnościami per se. Informacje zawarte w tej kategorii są zaleceniami, które mogą prowadzić do uniknięcia wystąpienia błędów bezpieczeństwa serwisu.

3. Metodyka dokumentowania podatności

Znalezione podatności, oprócz ich klasyfikacji pod względem rodzaju podatności (oraz krótkiego opisu), jak i stopnia zagrożenia, zostały podzielone na następujące podpunkty:

1. Okoliczności znalezienia
2. POC (Proof of concept)
3. Wpływ na serwis
4. Zalecenia naprawy
5. Referencje

for educational purposes only

4. Znalezienie podatności

a. Typu SQL Injection

Ten rodzaj podatności występuje, gdy silnik bazy danych lub biblioteka pośrednicząca zapytaniom kierowanym do bazy nie otrzymuje wyraźnej instrukcji, w jaki sposób interpretować dane przekazywane przez użytkownika. Może dojść do sytuacji, gdy dane przekazywane przez użytkownika zostaną potraktowane jako część kodu programisty i zinterpretowane jako część zapytania.

i. [SQL][CRITICAL]OWASP_LOGIN_01

1. Okoliczności znalezienia

Po umieszczeniu znaku apostrof w polu Username występuje błąd SQL syntax.

2. POC

Pod adresem: <http://192.168.56.101/mutillidae/index.php?page=login.php>, w polu Username należy wpisać znak apostrof oraz kliknąć Login:

Please sign-in

Username

Password

Login

Przedstawiony błąd bazy jest efektem wprowadzenia nieprawidłowego znaku.

Error Message

Failure is always an option	
Line	170
Code	0
File	/owaspbwa/mutillidae-git/classes/MySQLHandler.php
Message	<pre>/owaspbwa/mutillidae-git/classes/MySQLHandler.php on line 165: Error executing query: connect_errno: 0 errno: 1064 error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 1 client_info: 5.1.73 host_info: Localhost via UNIX socket) Query: SELECT username FROM accounts WHERE username='''; (0) [Exception]</pre>
Trace	<pre>#0 /owaspbwa/mutillidae-git/classes/MySQLHandler.php(283): MySQLHandler->doExecuteQuery('SELECT username...') #1 /owaspbwa/mutillidae-git/classes/SQLQueryHandler.php(250): MySQLHandler->executeQuery('SELECT username...') #2 /owaspbwa/mutillidae-git/includes/process-login-attempt.php(54): SQLQueryHandler->accountExists('') #3 /owaspbwa/mutillidae- git/index.php(277): include_once('/owaspbwa/mutillidae...') #4 {main}</pre>
Diagnostic Information	Error querying user account

3. Wpływ na serwis

Serwis jest podatny na złośliwe wstrzyknięcie zapytań SQL w polach służących do logowania. Atakujący jest w stanie zalogować się jako dowolny użytkownik

(np. administrator), jeśli do uwierzytelniania wykorzysta warunek logiczny zwracający zawsze prawdę (np. admin' OR '1=1'--).

4. Zalecenia naprawy

Zapytania kierowane do bazy danych powinny zostać objęte parametryzacją (parameterized queries) bądź pre-przygotowaniem (prepared statements), aby nakreślić, że dane od użytkownika powinny być traktowane jako tekst, a nie kod.

5. Referencje

https://www.websec.ca/kb/sql_injection

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

ii. [SQL][CRITICAL]OWASP_USER_LOOKUP_1

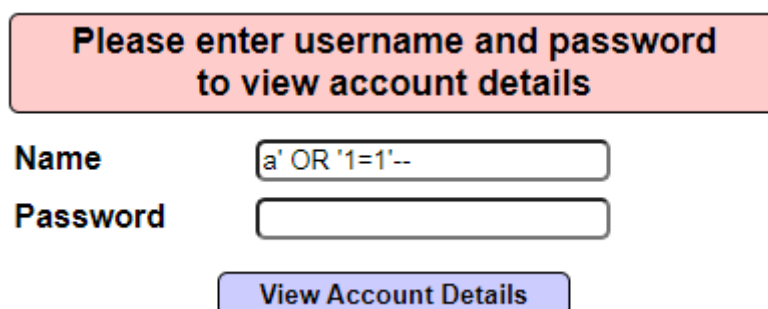
1. Okoliczności znalezienia

Po umieszczeniu znaku apostrof w polu Username występuje błąd SQL syntax. (patrz:[SQL][CRITICAL]OWASP_BWA_LOGIN_01)

Po wykorzystaniu warunku logicznego zwracającego prawdę baza danych zwróci listę wszystkich znajdujących się w niej kont użytkowników wraz z hasłami i sygnaturkami.

2. POC

Pod adresem <http://192.168.56.101/mutillidae/index.php?page=user-info.php>, w polu Name należy wpisać znak apostrof, dopisać warunek logiczny oraz kliknąć View Account Details:



Please enter username and password to view account details

Name

Password

View Account Details

Baza danych zwróci w ten sposób listę wszystkich zarejestrowanych użytkowników.

Please enter username and password to view account details

Name

Password

View Account Details

Dont have an account? [Please register here](#)

Results for "a' OR '1=1'-- ".26 records found.

Username=admin

Password=admin

Signature=g0t r00t?

Username=adrian

Password=someword

Signature=Zombie Films Rock!

Username=john

Password=monkey

Signature=I like the smell of confunk

Username=jeremy

Password=password

Signature=d1373 1337 speak

Username=bryce

Password=password

Signature=I Love SANS

3. Wpływ na serwis

Serwis jest podatny na złośliwe wstrzyknięcie zapytań SQL w polach służących do przeglądania informacji o użytkownikach. Atakujący jest w stanie pobrać listę wszystkich zarejestrowanych użytkowników wraz z hasłami i sygnaturami, jeśli do uwierzytelniania wykorzysta warunek logiczny zwracający zawsze prawdę (np. a' OR '1=1'--). W przypadku wykonania kolejnych zapytań do bazy danych z punktu widzenia tego panelu, atakujący jest w stanie pozyskać informację na temat wszystkich tabel, kolumn i wierszy bazy danych.

Results for "" UNION SELECT 1, database(),

Username=nowasp

Results for "" UNION SELECT 1, TABLE_NAME, 3, 4, 5, 6, 7 FROM information_schema.tables records found.

Username=accounts
Password=3
Signature=4

Username=balloon_tips
Password=3
Signature=4

Username=blogs_table
Password=3
Signature=4

Username=captured_data
Password=3
Signature=4

Username=credit_cards
Password=3
Signature=4

A także uzyskać dostęp do haszy haseł przechowywanych w bazie.

Results for "" UNION SELECT 1, secret, login, password,

Username=A.I.M. or Authentication Is Missing
Password=A.I.M.
Signature=~~6885858486f31843c5899c735d99457f045affd9~~

Username=Any bugs?
Password=bee
Signature=~~6885858486f31843c5899c735d99457f045affd9~~

4. Zalecenia naprawy

Zapytania kierowane do bazy danych powinny zostać objęte parametryzacją (parametized queries) bądź pre-przygotowaniem (prepared statements), aby nakreślić, że dane od użytkownika powinny być traktowane jako tekst, a nie kod. (patrz: [SQL][CRITICAL]OWASP_BWA_LOGIN_01)

5. Referencje

https://www.websec.ca/kb/sql_injection

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

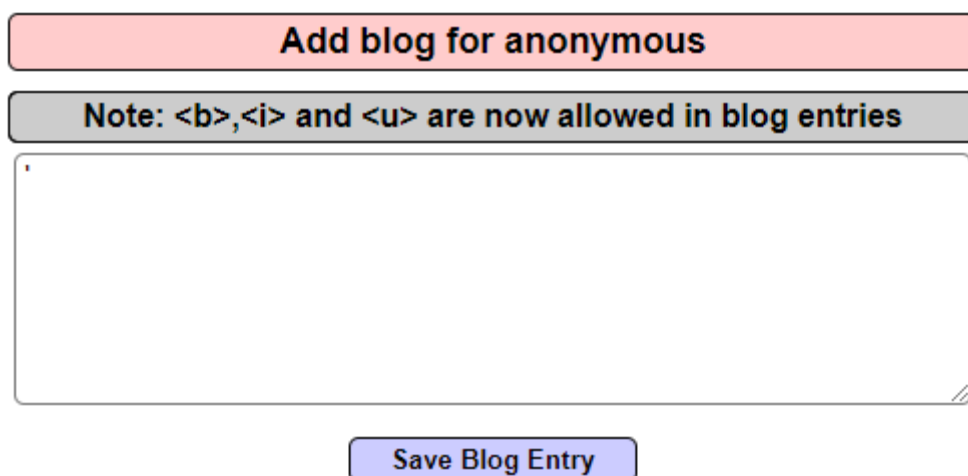
iii. [SQL][CRITICAL]OWASP_ADD_BLOG_1

1. Okoliczności znalezienia

Po umieszczeniu znaku apostrof w polu tekstowym dla bloga występuje błąd SQL syntax.

2. POC

Pod adresem: <http://192.168.56.101/mutillidae/index.php?page=add-to-your-blog.php>, w polu tekstowym należy wpisać znak apostrof oraz kliknąć Save Blog Entry:



Przedstawiony błąd bazy jest efektem wprowadzenia nieprawidłowego znaku.

Error Message

Failure is always an option	
Line	170
Code	0
File	/owaspbwa/mutillidae-git/classes/MySQLHandler.php
Message	<pre>/owaspbwa/mutillidae-git/classes/MySQLHandler.php on line 165: Error executing query: connect_errno: 0 errno: 1064 error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '', now())' at line 1 client_info: 5.1.73 host_info: Localhost via UNIX socket) Query: INSERT INTO blogs_table(blogger_name, comment, date) VALUES ('anonymous', '' ', now()) (0) [Exception]</pre>
Trace	<pre>#0 /owaspbwa/mutillidae-git/classes/MySQLHandler.php(283): MySQLHandler->doExecuteQuery('?????INSERT INT...) #1 /owaspbwa/mutillidae- git/classes/SQLQueryHandler.php(155): MySQLHandler->executeQuery('?????INSERT INT...) #2 /owaspbwa/mutillidae-git/add-to-your-blog.php(147): SQLQueryHandler->insertBlogRecord('anonymous', '' ') #3 /owaspbwa/mutillidae-git/index.php(614): require_once('/owaspbwa/mutillidae- git/classes/MySQLHandler.php') #4 {main}</pre>
Diagnostic Information	Error inserting blog for anonymous
Click here to reset the DB	

3. Wpływ na serwis

Serwis jest podatny na złośliwe wstrzyknięcie zapytań SQL w polu służącym do dodania bloga, w tym podatny na atak Blind SQL Injection.

4. Zalecenia naprawy

Zapytania kierowane do bazy danych powinny zostać objęte parametryzacją (parametized queries) bądź pre-przygotowaniem (prepared statements), aby nakreślić, że dane od użytkownika powinny być traktowane jako tekst, a nie kod.

5. Referencje

https://www.websec.ca/kb/sql_injection

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

https://owasp.org/www-community/attacks/Blind_SQL_Injection

iv. [SQL][CRITICAL]OWASP_REGISTER_1

1. Okoliczności znalezienia

Po umieszczeniu znaku apostrof w polu tekstowym do rejestracji występuje błąd SQL syntax.

2. POC

Pod adresem: <http://192.168.56.101/mutillidae/index.php?page=register.php>, w polu Username należy wpisać znak apostrof oraz kliknąć Create Account:

Please choose your username, password and signature

Username

Password

[Password Generator](#)

Confirm Password

Signature

Create Account

Przedstawiony błąd bazy jest efektem wprowadzenia nieprawidłowego znaku.

Error Message

Failure is always an option	
Line	170
Code	0
File	/owaspbwa/utillidae-git/classes/MySQLHandler.php
Message	<pre>/owaspbwa/utillidae-git/classes/MySQLHandler.php on line 165: Error executing query: connect_errno: 0 errno: 1064 error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '', '', '' at line 1 client_info: 5.1.73 host_info: Localhost via UNIX socket) Query: INSERT INTO accounts (username, password, mysignature) VALUES ('', '', '') (0) [Exception]</pre>
Trace	<pre>#0 /owaspbwa/utillidae-git/classes/MySQLHandler.php(283): MySQLHandler->doExecuteQuery('INSERT INTO acc...') #1 /owaspbwa/utillidae- git/classes/SQLQueryHandler.php(350): MySQLHandler->executeQuery('INSERT INTO acc...') #2 /owaspbwa/utillidae-git/register.php(90): SQLQueryHandler->insertNewUserAccount('', '', '') #3 /owaspbwa/utillidae-git/index.php(614): require_once('/owaspbwa/utill...') #4 {main}</pre>
Diagnostic Information	Failed to add account
Click here to reset the DB	

3. Wpływ na serwis

Serwis jest podatny na złośliwe wstrzyknięcie zapytań SQL w polu służącym do rejestracji.

4. Zalecenia naprawy

Zapytania kierowane do bazy danych powinny zostać objęte parametryzacją (parametized queries) bądź pre-przygotowaniem (prepared statements), aby nakreślić, że dane od użytkownika powinny być traktowane jako tekst, a nie kod.

5. Referencje

https://www.websec.ca/kb/sql_injection

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

b. Typu XSS

Ten rodzaj podatności polega na osadzeniu w treści atakowanej strony i/lub adresie URL i/lub w DOM (Document Object Model) strony kodu (zwykle JavaScript), który wyświetlony innym użytkownikom bądź administratorom serwisu może prowadzić do wykonania przez nich niepożądanych akcji lub do zdalnego wykonania kodu (RCE). Skrypt umieszczony w ten sposób może obejść niektóre mechanizmy kontroli dostępu do danych lub wykorzystać uprawnienia ofiary, których w normalnej sytuacji atakujący nie mógłby posiadać.

i. [XSS][CRITICAL]OWASP_DNS_LOOKUP_1

1. Okoliczności znalezienia

Po umieszczeniu kodu HTML `<input>` w polu przeznaczonym na wpisanie adresu IP serwis wykonuje kod i zwraca go na stronie.

2. POC

Pod adresem `http://192.168.56.101/mutillidae/index.php?page=dns-lookup.php`, w polu Hostname/IP należy wpisać `<input>` oraz kliknąć Lookup DNS:

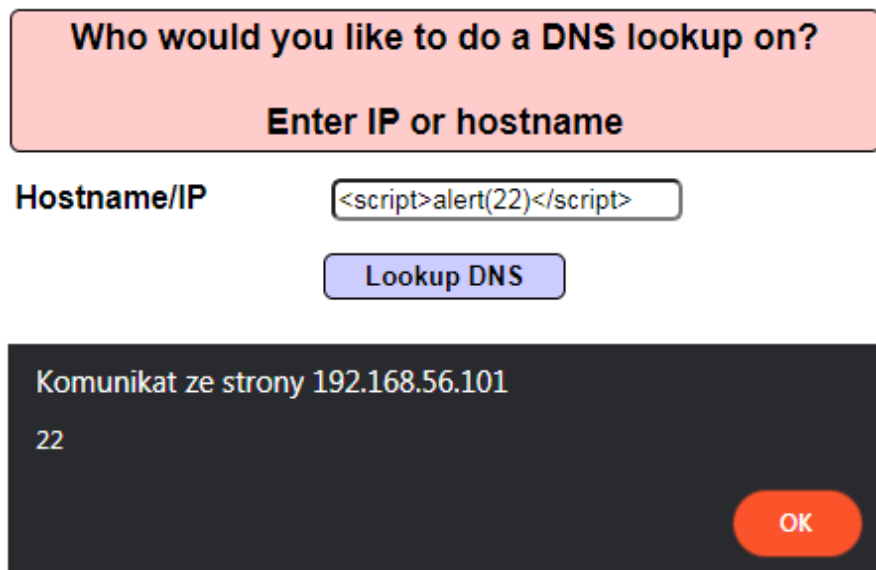
The screenshot shows a web application interface with a pink header box containing the text "Who would you like to do a DNS lookup on?" and "Enter IP or hostname". Below this, the label "Hostname/IP" is followed by a text input field containing the HTML code `<input>`. To the right of the input field is a blue button labeled "Lookup DNS".

Przedstawione pole Input oznacza brak enkodowania znaków specjalnych.

The screenshot shows the same web application interface as above, but with the input field empty. Below the input field is a blue button labeled "Lookup DNS". At the bottom of the page, there is a grey box labeled "Results for" followed by an empty text input field.

3. Wpływ na serwis

Serwis jest podatny na wykonanie złośliwego kodu JavaScript.



The screenshot shows a web application interface for a DNS lookup. At the top, a pink box contains the text "Who would you like to do a DNS lookup on?". Below this, a label "Enter IP or hostname" is positioned above a text input field. The input field contains the JavaScript code "<script>alert(22)</script>". To the left of the input field is the label "Hostname/IP". Below the input field is a blue button labeled "Lookup DNS". Below the button, a dark grey box displays the message "Komunikat ze strony 192.168.56.101" and the number "22". An orange "OK" button is located in the bottom right corner of this message box.

4. Zalecenia naprawy

Dane od użytkownika powinny być enkodowane i przekazywane do dalszego przetwarzania z encjami HTML.

5. Referencje

[https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_\(XSS\)](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

ii. [XSS][CRITICAL]OWASP_USER_LOOKUP_2

1. Okoliczności znalezienia

Po umieszczeniu kodu HTML w polu przeznaczonym na wyszukiwanie użytkownika serwis wykonuje kod i zwraca go na stronie.

2. POC

Pod adresem <http://192.168.56.101/mutillidae/index.php?page=user-info.php>, w polu Name należy wpisać `<h1>Oops</h1>` oraz kliknąć View Account Details.

Please enter username and password to view account details

Name

Password

View Account Details

Przedstawiony rezultat oznacza brak enkodowania znaków specjalnych.

Please enter username and password to view account details

Name

Password

View Account Details

Dont have an account? [Please register here](#)

Results for "

Oops

".0 records found.

3. Wpływ na serwis

Serwis jest podatny na wykonanie złośliwego kodu JavaScript.

Please enter username and password to view account details

Name

Password

View Account Details

Komunikat ze strony 192.168.56.101

22

OK

4. Zalecenia naprawy

Dane od użytkownika powinny być enkodowane i przekazywane do dalszego przetwarzania z encjami HTML.

5. Referencje

[https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_\(XSS\)](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

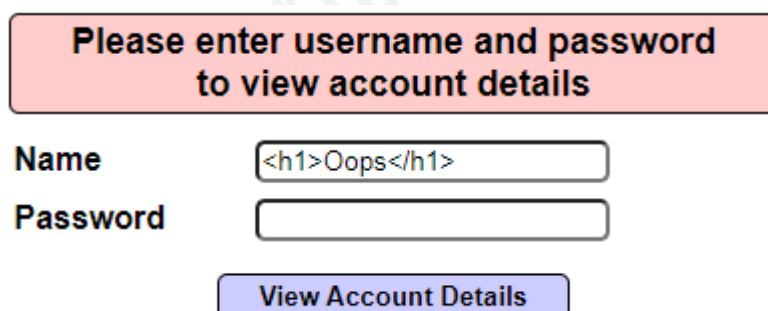
iii. [XSS][CRITICAL]OWASP_USER_LOOKUP_3

1. Okoliczności znalezienia

Po umieszczeniu kodu HTML w polu przeznaczonym na wyszukiwanie użytkownika serwis wykonuje kod i zwraca go w adresie URL.

2. POC

Pod adresem `http://192.168.56.101/mutillidae/index.php?page=user-info.php`, w polu Name należy wpisać `<h1>Oops</h1>` oraz kliknąć View Account Details.



Please enter username and password to view account details

Name

Password

[View Account Details](#)

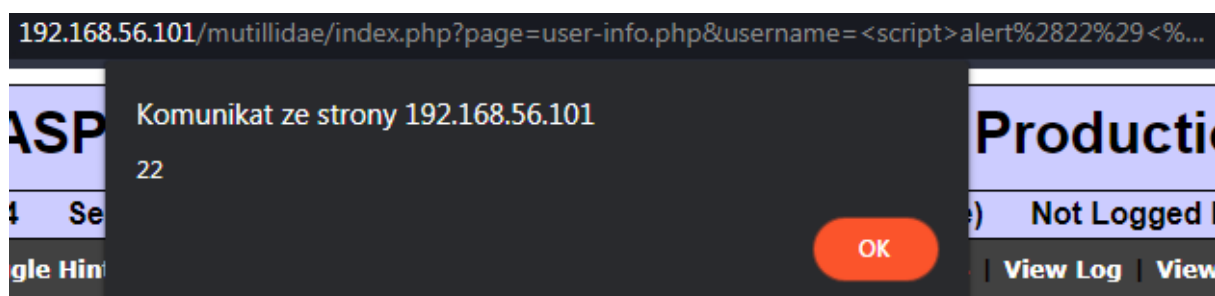
Przedstawiony rezultat oznacza brak enkodowania znaków `>` i `<`.

```
dex.php?page=user-info.php&username=<h1>Oops<%2Fh1>&
```

3. Wpływ na serwis

Serwis jest podatny na wykonanie złośliwego kodu JavaScript. Dodatkowo, atakujący może przesłać spreparowany adres do ofiary, u której zostanie on wykonany po kliknięciu.

`&username=<script>alert%2822%29<%2Fscript>`



4. Zalecenia naprawy

Dane od użytkownika powinny być enkodowane i przekazywane do dalszego przetwarzania z encjami HTML.

5. Referencje

[https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_\(XSS\)](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

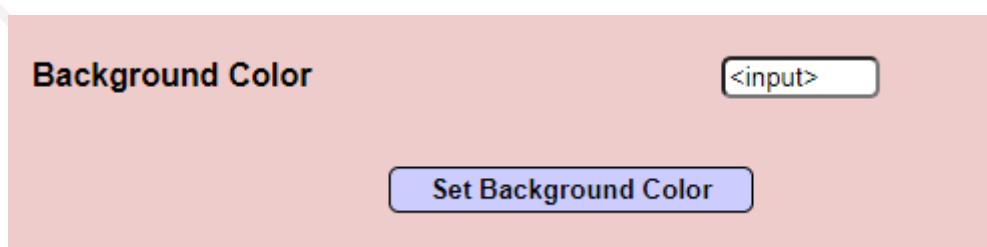
iv. [XSS][HIGH]OWASP_BACKGROUND_COLOR_1

1. Okoliczności znalezienia

Po umieszczeniu kodu HTML w polu przeznaczonym na zdefiniowanie koloru tła serwis wykonuje kod i zwraca go na stronie.

2. POC

Pod adresem <http://192.168.56.101/mutillidae/index.php?page=set-background-color.php>, w polu Background Color należy wpisać `<input>` oraz kliknąć Set Background Color.



Przedstawiony rezultat oznacza brak enkodowania znaków HTML.

Background Color

Set Background Color

The current background color is

3. Wpływ na serwis

Serwis jest podatny na wykonanie złośliwego kodu JavaScript.

Background Color

Set Background Color

Komunikat ze strony 192.168.56.101

22

OK

4. Zalecenia naprawy

Dane od użytkownika powinny być enkodowane i przekazywane do dalszego przetwarzania z encjami HTML.

5. Referencje

[https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_\(XSS\)](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

v. [XSS][CRITICAL]OWASP_USER_POLL_1

1. Okoliczności znalezienia

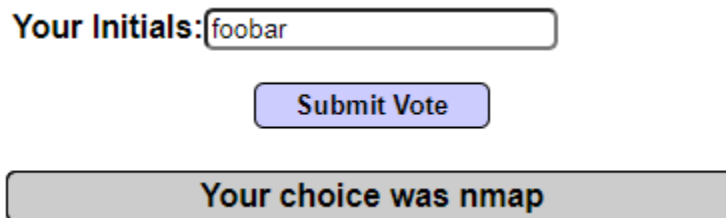
Po spreparowaniu adresu URL serwis wykonuje kod z query i zwraca go na stronie.

2. POC

Oddanie głosu w ankiecie sprawia, że strona załaduje się ponownie z wybranym głosem oraz podanymi inicjałami. Przykładowo przy wybraniu **nmap** i podaniu inicjałów **foobar**, adres URL będzie wyglądał następująco:

```
http://192.168.56.101/mutillidae/index.php?page=user-poll.php&csrf-token=&choice=nmap&initials=foobar&user-poll-php-submit-button=Submit+Vote
```

a głosowanie jak poniżej:

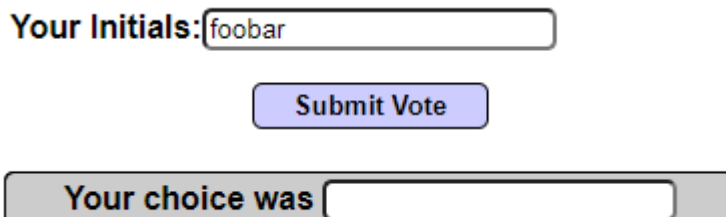


The screenshot shows a web interface for a poll. At the top, it says "Your Initials:" followed by a text input field containing "foobar". Below this is a blue button labeled "Submit Vote". At the bottom, a grey box displays the message "Your choice was nmap".

Teraz należy zmodyfikować adres URL, który pozwoli na zauważenie podatności, np.. poprzez podmianę głosu na `<input>` jak poniżej:

```
http://192.168.56.101/mutillidae/index.php?page=user-poll.php&csrf-token=&choice=%3Cinput%3E&initials=foobar&user-poll-php-submit-button=Submit+Vote
```

a głosowanie wyświetli się jak poniżej:

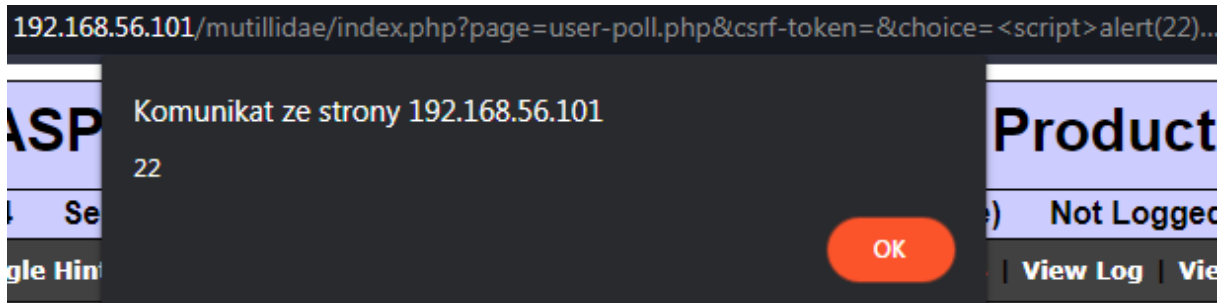


The screenshot shows the same web interface as before. The "Your Initials:" field still contains "foobar" and the "Submit Vote" button is present. However, the grey box at the bottom now only says "Your choice was" followed by an empty text input field, indicating that the vote was not properly recorded.

Prezentowany rezultat oznacza brak enkodowania znaków HTML po stronie serwera, mimo że w adresie URL mylnie mogą zostać zinterpretowane jako enkodowane.

3. Wpływ na serwis

Serwis jest podatny na wykonanie złośliwego kodu JavaScript. Ponadto ofiara może zostać poproszona o kliknięcie w złośliwy link, który sprawi, że stanie się nieświadomym uczestnikiem głosowania (np. z opcją celowo wybraną przez atakującego).



4. Zalecenia naprawy

Dane od użytkownika powinny być enkodowane i przekazywane do dalszego przetwarzania z encjami HTML.

5. Referencje

[https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_\(XSS\)](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

vi. [XSS][CRITICAL]OWASP_REGISTER_2

1. Okoliczności znalezienia

Po umieszczeniu kodu HTML w polu Username zostanie on wykonany na stronie po rejestracji.

2. POC

Pod adresem <http://192.168.56.101/mutillidae/index.php?page=register.php>, należy wypełnić pola do rejestracji, a w polu Username należy wpisać `<h1>Oops</h1>` oraz kliknąć Create Account.

Please choose your username, password and signature

Username	<input type="text" value="<h1>Oops</h1>"/>	
Password	<input type="password" value="...."/>	Password Generator
Confirm Password	<input type="password" value="...."/>	
Signature	<input type="text" value="totally safe register"/>	
<input type="button" value="Create Account"/>		

Przedstawiony rezultat oznacza brak enkodowania znaków HTML i wykonanie kodu.

Account created for

Oops

3. Wpływ na serwis

Serwis jest podatny na wykonanie złośliwego kodu JavaScript. Ponadto możliwe jest zalogowanie dokładnie tym loginem do serwisu. Wówczas na stronie w panelu użytkownika pojawi się ponownie sformatowany username.

Logged In User:
Oops
(totally safe register)

Możliwa jest również rejestracja skryptem wywołującym alert. Wówczas po zalogowaniu jest on natychmiast wykonywany, jeszcze przed załadowaniem większości strony.

Username

Password

Login

192.168.56.101/mutillidae/index.php?popUpNotificationCode=AU1

Komunikat ze strony 192.168.56.101

22

OK

4. Zalecenia naprawy

Dane od użytkownika powinny być enkodowane i przekazywane do dalszego przetwarzania z encjami HTML.

5. Referencje

[https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_\(XSS\)](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

vii. [XSS][CRITICAL]OWASP_PASSWORD_GENERATOR_1

1. Okoliczności znalezienia

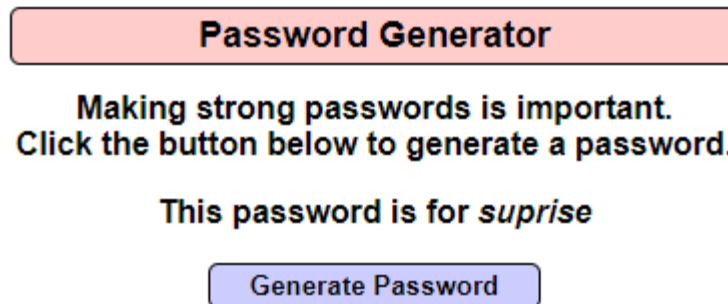
Po spreparowaniu adresu URL serwis wykonuje kod z query i zwraca go na stronie.

2. POC

Przejsście na adres URL sprawia, że strona załadowuje się z adresem, który jako parametr w zapytaniu przyjmuje username **anonymous**. Ten sam username wyświetla się także na stronie. Spreparowanie adresu URL sprawia, że modyfikowany username pojawia się, zgodnie z przewidywaniami, także na stronie. Na przykład spreparowanie adresu w poniższy sposób:

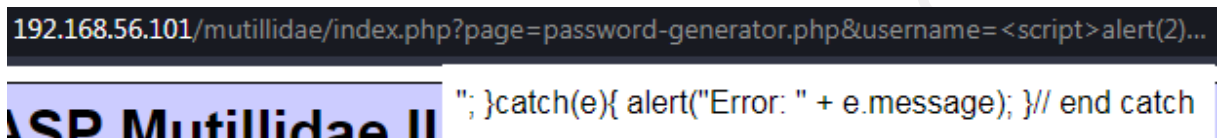
`http://192.168.56.101/mutillidae/index.php?page=password-generator.php&username=<i>suprise</i>`

sprawi że username wyświetli jak poniżej:



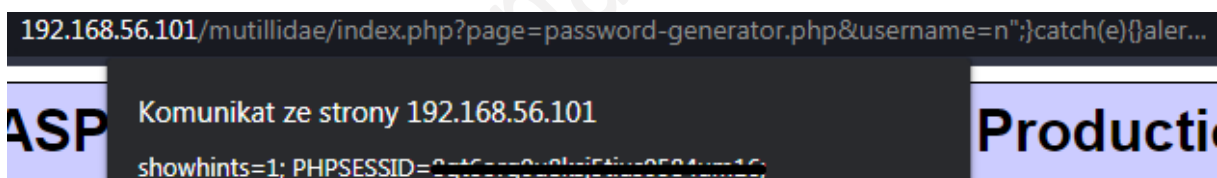
3. Wpływ na serwis

Choć na pierwszy rzut oka serwis stara się ignorować wstrzyknięcie skryptu do adresu URL, to nadal jest podatny na wykonanie złośliwego kodu JavaScript.



Odpowiednie spreparowanie adresu URL w miejscu parametru sprawi, że skrypt opuści blokujący znacznik i będzie mógł się wykonać na stronie i przykładowo wyświetlić ciasteczka ofiary.

`http://192.168.56.101/mutillidae/index.php?page=password-generator.php&username=""; }catch(e){ alert("Error: " + e.message); }// end catch`



4. Zalecenia naprawy

Dane od użytkownika powinny być enkodowane i przekazywane do dalszego przetwarzania z encjami HTML.

5. Referencje

[https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_\(XSS\)](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

viii. [XSS][CRITICAL]OWASP_REGISTER_3

1. Okoliczności znalezienia

Po umieszczeniu kodu HTML w polu Username i/lub Signature zostaje on wykonany na stronie po rejestracji, po zalogowaniu i w każdym miejscu (w obrębie całego serwisu), w którym na stronie zostają wyświetlone informacje o tym użytkowniku.

2. POC

Pod adresem <http://192.168.56.101/mutillidae/index.php?page=register.php>, należy wypełnić pola do rejestracji, a w polu Username i/lub Signature (tym razem tylko w Signature) należy wpisać `<h1>Here I am</h1>` oraz kliknąć Create Account.

Please choose your username, password and signature

Username

foobar

Password

....

Password Generator

Confirm Password

....

Signature

`<h1>Here I am</h1>`

Create Account

Przedstawiony rezultat oznacza brak enkodowania znaków HTML i wykonanie kodu.

Logged In User: foobar (
Here I am
)

3. Wpływ na serwis

Serwis jest wyjątkowo podatny na wykonanie złośliwego kodu JavaScript, ponieważ będzie się on znajdował na serwerze i wykona się u każdego użytkownika, który na danej stronie wyświetli jakiegokolwiek informacje o złośliwym użytkowniku. Rejestracja użytkownika z alertem w opisie sprawia, że u każdego przeglądającego (np. u administratora) zostaje wyświetlony alert.

Username

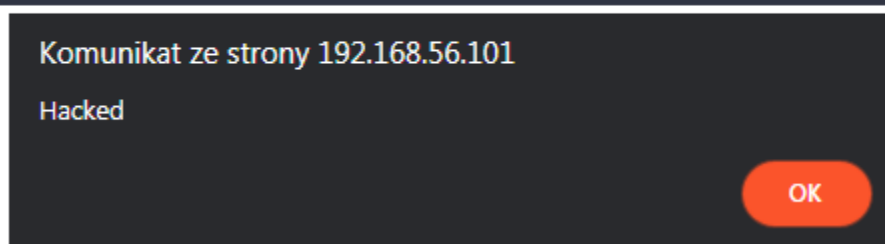
Password [Password Generator](#)

Confirm Password

Signature

[Create Account](#)

192.168.56.101/mutillidae/index.php?page=user-info.php&username=sam_sepiol%



4. Zalecenia naprawy

Dane od użytkownika powinny być enkodowane i przekazywane do dalszego przetwarzania z encjami HTML.

5. Referencje

[https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_\(XSS\)](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

ix. [XSS][CRITICAL]OWASP_ADD_BLOG_2

1. Okoliczności znalezienia

Po umieszczeniu kodu HTML w polu przeznaczonym na dodanie bloga zostaje on wykonany na stronie po dodaniu, a także u każdego użytkownika, u którego na stronie pojawi się przegląd bloga użytkownika dodającego złośliwy kod.

2. POC

Pod adresem `http://192.168.56.101/mutillidae/index.php?page=add-to-your-blog.php`, należy wprowadzić kod HTML, np. `<h1>Stored damage</h1>`

Add blog for anonymous

Note: ,<i> and <u> are now allowed in blog entries

```
<h1>Stored damage</h1>
```

Save Blog Entry

Przedstawiony rezultat oznacza brak enkodowania znaków HTML i wykonanie kodu.

Comment

Stored damage

3. Wpływ na serwis

Serwis jest wyjątkowo podatny na wykonanie złośliwego kodu JavaScript, ponieważ będzie się on znajdował na serwerze i wykona się u każdego użytkownika, który na danej stronie wyświetli jakiegokolwiek informację o złośliwym wpisie. Skrypt nie musi uruchamiać się od razu po załadowaniu strony z wpisem, ale może również wykorzystać naiwność użytkownika, zachęcając go np. do najeżdżenia myszką na złośliwy link. Przykład wykorzystania eventu DOM poniżej.

Add blog for anonymous

Note: ,<i> and <u> are now allowed in blog entries

```

```


Save Blog Entry

Komunikat ze strony 192.168.56.101

2

OK

3 Current Blog Entries

Date	
2021-01-26 19:07:26	 Pobierz testownik!!

4. Zalecenia naprawy

Dane od użytkownika powinny być enkodowane i przekazywane do dalszego przetwarzania z encjami HTML.

5. Referencje

[https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_\(XSS\)](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

c. Typu CSRF

Ten rodzaj podatności występuje, gdy ofiara nieświadomie przesyła spreparowane przez atakującego żądanie do serwera, które nie zostaje w żaden sposób zweryfikowane przed wykonaniem. Dotyczy to zwykle serwisów wymagających zalogowania lub innego ograniczenia, które dzięki wykorzystaniu zaufania serwisu do tożsamości ofiary można łatwo pominąć, jeśli ta postanowi wysłać zmodyfikowane żądanie http do serwisu.

i. [CSRF][HIGH]OWASP_USER_POLL_2

1. Okoliczności znalezienia

Po wzięciu udziału w ankiecie zostaje wygenerowane żądanie do serwera zawierające pusty CSRF Token. Token nie jest w żaden sposób weryfikowany po stronie serwera i możliwe jest wykonanie spreparowanego żądania.

2. POC

Przechwycenie żądania rozpoczyna się od uruchomienia i przygotowania narzędzia Burp Suite. Następnie należy włączyć bramkę Proxy w przeglądarce (np. Mozilla Firefox). Pod adresem <http://192.168.56.101/mutillidae/index.php?page=user-poll.php>, należy wziąć udział w głosowaniu i kliknąć Submit Vote.

Choose Your Favorite Security Tool

Initial your choice to make your vote count

- ☒ nmap
- ☐ wireshark
- ☐ tcpdump
- ☐ netcat
- ☐ metasploit
- ☐ kismet
- ☐ Cain
- ☐ Ettercap
- ☐ Paros
- ☐ Burp Suite
- ☐ Sysinternals
- ☐ inSIDDer

Your Initials:

Submit Vote

Żądanie zostanie przechwycone i odczytane przez narzędzie Burp Suite w zakładce Proxy.



W następnej kolejności należy zmodyfikować żądanie, np. z **nmap** na dostępny w głosowaniu **Burp Suite** i przekazać je dalej przyciskiem Forward.

```
1 GET /mutillidae/index.php?page=user-poll.php&csrf-token=&choice=BurpSuite&initials=Intercept&user-poll-php-submit-button=Submit+Vote HTTP/1.1
2 Host: 192.168.56.101
3 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: pl,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.56.101/mutillidae/index.php?page=user-poll.php
9 Cookie: showhints=1; PHPSESSID=dgs2poq6th7mdc0rogighlad96; acopendivids=swingset,jotto,phphb2,redmine; acgroupswithpersist=nada
10 Upgrade-Insecure-Requests: 1
11
12
```

W ten sposób serwer otrzymał głos na **Burp Suite** od osoby **Intercept**, mimo że głosowała pierwotnie na **nmap**.

Your Initials:

Your choice was Burp Suite

3. Wpływ na serwis

Serwis jest podatny na atak z wykorzystaniem podatności CSRF. Bez odpowiedniej weryfikacji żądania atakujący jest w stanie spreparować żądanie i podesłać je ofierze, prosząc ją o odwiedzenie serwisu pod przygotowanym linkiem. W ten sposób atakujący jest w stanie wpłynąć na wybrany głos w ankiecie.

Zakładając istnienie podstawionej witryny internetowej, atakujący jest w stanie, mając na uwadze, że możliwe jest wykorzystanie podatności CSRF, spreparować żądanie do serwera i podesłać je do ofiary np. w ten sposób:

```
<html>
<head>
</head>
<body>


</html>
```

Ofiara zauważy zdjęcie kota, a chwilę po załadowaniu zdjęcia zostanie wysłane żądanie, o którym nie będzie miała pojęcia. W przypadku wykorzystania do głosowania sesji cookies w jej przeglądarce, głosowanie mogło by zostać zinterpretowane jako wykonane właśnie z jej konta użytkownika, pomimo jej fizycznej zgody.

4. Zalecenia naprawy

Należy bezzwłocznie uruchomić tokenizację żądań na stronie. Należy włączyć atrybut SameSite dla ciasteczek sesyjnych. Należy zaprzestać używania nagłówka GET dla operacji wysyłających żądanie modyfikacji.

5. Referencje

<https://owasp.org/www-community/attacks/csrf>

https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

ii. [CSRF][CRITICAL]OWASP_ADD_BLOG_3

1. Okoliczności znalezienia

Po dodaniu wpisu do bloga zostaje wygenerowane żądanie do serwera zawierające pusty CSRF Token. Token nie jest w żaden sposób weryfikowany po stronie serwera i możliwe jest wykonanie spreparowanego żądania.

2. POC

Przechwycenie żądania rozpoczyna się od uruchomienia i przygotowania narzędzia Burp Suite. Następnie należy włączyć bramkę Proxy w przeglądarce (np. Mozilla Firefox). Pod adresem `http://192.168.56.101/mutillidae/index.php?page=add-to-your-blog.php`, dodać wpis i kliknąć Save Blog Entry.

Add blog for anonymous

Note: , <i> and <u> are now allowed in blog entries

`nothing suspicious at all`

Save Blog Entry

Żądanie zostanie przechwycone i odczytane przez narzędzie Burp Suite w zakładce Proxy.

```
1 POST /mutillidae/index.php?page=add-to-your-blog.php HTTP/1.1
2 Host: 192.168.56.101
3 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: pl,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 100
9 Origin: http://192.168.56.101
10 Connection: close
11 Referer: http://192.168.56.101/mutillidae/index.php?page=add-to-your-blog.php
12 Cookie: showhints=1; PHPSESSID=dgs2poq6th7mdc0rogighlad96; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
13 Upgrade-Insecure-Requests: 1
14
15 csrf-token=&blog_entry=nothing+suspicious+at+all+&add-to-your-blog-php-submit-button=Save+Blog+Entry
```

W następnej kolejności należy zmodyfikować żądanie, np. z **nothing suspicious at all** na **very suspicious** i przekazać je dalej przyciskiem Forward.

```
1 POST /mutillidae/index.php?page=add-to-your-blog.php HTTP/1.1
2 Host: 192.168.56.101
3 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: pl,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 100
9 Origin: http://192.168.56.101
10 Connection: close
11 Referer: http://192.168.56.101/mutillidae/index.php?page=add-to-your-blog.php
12 Cookie: showhints=1; PHPSESSID=dgs2poq6th7mdc0rogighlad96; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
13 Upgrade-Insecure-Requests: 1
14
15 csrf-token=&blog_entry=very+suspicious+&add-to-your-blog-php-submit-button=Save+Blog+Entry
```

W ten sposób serwer otrzymał wpis o treści **very suspicious**, mimo że wpis brzmiał pierwotnie **nothing suspicious at all**.

Date	Comment
2021-01-27 05:35:56	very suspicious

3. Wpływ na serwis

Serwis jest podatny na atak z wykorzystaniem podatności CSRF. Bez odpowiedniej weryfikacji żądania atakujący jest w stanie spreparować żądanie i podesłać je ofierze, prosząc ją o odwiedzenie serwisu pod przygotowanym linkiem. W ten sposób atakujący jest w stanie wpłynąć na zawartość wpisu.

Zakładając istnienie podstawionej witryny internetowej, atakujący jest w stanie, mając na uwadze, że możliwe jest wykorzystanie podatności CSRF, spreparować żądanie do serwera i podesłać je do ofiary np. w ten sposób:

```
<html>
<head>
</head>
<body>


</body>
</html>
```

Ofiara zauważy zdjęcie kota, a chwilę po załadowaniu zdjęcia zostanie wysłane żądanie, o którym nie będzie miała pojęcia. W przypadku wykorzystania do wpisu sesji cookies w jej przeglądarce, wpis zostanie zinterpretowany jako dodany umyślnie przez ofiarę, przez co mogą zostać wykorzystane jej uprawnienia, których w normalnej sytuacji atakujący nie mógłby pozyskać (np. wpis w miejscu o ograniczonym dostępie). W zawartości wpisu możliwe jest wstrzyknięcie XSS'a jak pokazano wyżej. W efekcie, dodany wpis sprawi, że innym ofiarom zostanie wyświetlony monit z zawartością ciasteczka. Daje to możliwość wykonania złośliwego kodu JavaScript.

Welcome To The Blog

showhints=1; PHPSESSID=dgs2poq6th7mdc0rogighlad96; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada

4. Zalecenia naprawy

Należy bezzwłocznie uruchomić tokenizację żądań na stronie. Należy włączyć atrybut SameSite dla ciasteczek sesyjnych. Należy zaprzestać używania nagłówka GET dla operacji wysyłających żądanie modyfikacji.

5. Referencje

<https://owasp.org/www-community/attacks/csrf>

https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

iii. [CSRF][CRITICAL]OWASP_REGISTER_4

1. Okoliczności znalezienia

Po ukończeniu procesu rejestracji użytkownika zostaje wygenerowane żądanie do serwera zawierające pusty CSRF Token. Token nie jest w żaden sposób weryfikowany po stronie serwera i możliwe jest wykonanie spreparowanego żądania.

2. POC

Przechwycenie żądania rozpoczyna się od uruchomienia i przygotowania narzędzia Burp Suite. Następnie należy włączyć bramkę Proxy w przeglądarce (np. Mozilla Firefox). Pod <http://192.168.56.101/mutillidae/index.php?page=register.php>, przeprowadzić proces rejestracji i kliknąć Create Account.

Username

Password Password Generator

Confirm Password

Signature

Żądanie zostanie przechwycone i odczytane przez narzędzie Burp Suite w zakładce Proxy.

```
POST /mutillidae/index.php?page=register.php HTTP/1.1
Host: 192.168.56.101
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: pl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 156
Origin: http://192.168.56.101
Connection: close
Referer: http://192.168.56.101/mutillidae/index.php?page=register.php
Cookie: showhints=1; PHPSESSID=dgs2poq6th7mdc0rogighlad96; acopendivids=swingset,jotto,phpb2,redmine; acgroupswithpersist=nada
Upgrade-Insecure-Requests: 1

csrf-token=&username=normaluser&password=test&confirm_password=test&my_signature=Hi+its+me+normal+user%21%21%21&register-php-submit-button=Create+Account
```

W następnej kolejności należy zmodyfikować żądanie, np. z **normaluser** w parametrze username na **newbiehacker** i przekazać je dalej przyciskiem Forward.

```
POST /mutillidae/index.php?page=register.php HTTP/1.1
Host: 192.168.56.101
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: pl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 156
Origin: http://192.168.56.101
Connection: close
Referer: http://192.168.56.101/mutillidae/index.php?page=register.php
Cookie: showhints=1; PHPSESSID=dgs2poq6th7mdc0rogighlad96; acopendivids=swingset,jotto,phpb2,redmine; acgroupswithpersist=nada
Upgrade-Insecure-Requests: 1

csrf-token=&username=newbiehacker&password=test&confirm_password=test&my_signature=Hi+its+me+normal+user%21%21%21&register-php-submit-button=Create+Account
```

W ten sposób serwer otrzymał wpis z loginem **newbiehacker**, mimo że wpis brzmiał pierwotnie **normaluser**.

Account created for newbiehacker.

3. Wpływ na serwis

Serwis jest podatny na atak z wykorzystaniem podatności CSRF. Bez odpowiedniej weryfikacji żądania atakujący jest w stanie spreparować żądanie i podesłać je ofierze, prosząc ją o odwiedzenie serwisu pod przygotowanym linkiem. W ten sposób atakujący jest w stanie wpłynąć na zawartość wpisu.

Zakładając istnienie podstawionej witryny internetowej, atakujący jest w stanie, mając na uwadze, że możliwe jest wykorzystanie podatności CSRF oraz że przykładowo rejestracją nowych użytkowników zajmuje się jedynie administrator witryny, spreparować żądanie do serwera i podesłać je do ofiary np. w ten sposób:

```
<html>
<head>
</head>
<body>


</body>
</html>
```

Ofiara zauważy zdjęcie kota, a chwilę po załadowaniu zdjęcia zostanie wysłane żądanie, o którym nie będzie miała pojęcia. W przypadku wykorzystania do wpisu sesji cookies w jej przeglądarce, wpis zostanie zinterpretowany jako dodany umyślnie przez ofiarę, przez co zostaną wykorzystane jej uprawnienia, których w normalnej sytuacji atakujący nie mógłby – autoryzację rejestracji nowego użytkownika. W zawartości wpisu możliwe jest wstrzyknięcie XSS'a jak pokazano wyżej. W efekcie, dodany wpis sprawi, że innym ofiarom zostanie wyświetlony monit z zawartością ciasteczka. Daje to możliwość wykonania złośliwego kodu JavaScript.

Logged In User: intruz (

showhints=1; username=intruz; uid=26; PHPSESSID=dgs2poq6th7mdc0rogighlad96; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada

4. Zalecenia naprawy

Należy bezzwłocznie uruchomić tokenizację żądań na stronie. Należy włączyć atrybut SameSite dla ciasteczek sesyjnych. Należy zaprzestać używania nagłówka GET dla operacji wysyłających żądanie modyfikacji.

5. Referencje

<https://owasp.org/www-community/attacks/csrf>

https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

5. Zalecenia bezpieczeństwa

W tym rozdziale ujęto ogólne zalecenia poprawy bezpieczeństwa serwisu, obejmujące kwestie m.in. przechowywania haseł, przetwarzania danych i ich widoczności dla użytkowników serwisu (w tym użytkowników niezalogowanych).

i. [INFO]OWASP_LOGIN_02

1. Problem

Pod adresem: <http://192.168.56.101/mutillidae/index.php?page=login.php> znajduje się panel logowania. W przypadku podania nieprawidłowego loginu wyświetlany jest komunikat o nieistniejącym koncie, zaś w przypadku podania loginu prawidłowego i nieprawidłowego hasła wyświetlany jest komunikat o nieprawidłowym hasle. Atakujący może w ten sposób za pomocą ataku bruteforce znaleźć w pierwszej kolejności istniejący login konta, a następnie przy jego pomocy złamać hasło.

Account does not exist	Password incorrect
Please sign-in	Please sign-in
Username <input type="text"/>	Username <input type="text"/>
Password <input type="text"/>	Password <input type="text"/>

2. Zalecenie

Należy ujednolicić komunikat niezależnie od wprowadzanych danych tak, że atakujący nie będzie w stanie dowiedzieć się, które z wprowadzanych danych są nieprawidłowe, a które zostały odnalezione w bazie danych. Należy również wydłużyć o pewną ilość czasu przetwarzanie żądania tak, by atakujący nie był w stanie zauważyć, które żądanie zostało wysłane do bazy (np. ze względu na zgodność przekazanych danych), a które zostało odrzucone (wobec czego zostało zrealizowane szybciej). (patrz: Key Stretching)

3. Referencje

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

<https://simplicable.com/new/key-stretching-definition>

ii. [INFO]OWASP_ERROR_LOG_1

1. Problem

W przypadku wystąpienia błędów podczas korzystania z serwisu wyświetlają się informacje diagnostyczne, które nie powinny być widoczne na środowisku produkcyjnym. Przykład komunikatu po wystąpieniu błędu związanego z SQL syntax:

Error Message	
Failure is always an option	
Line	170
Code	0
File	/owaspbwa/mutillidae-git/classes/MySQLHandler.php
Message	<pre>owaspbwa/mutillidae-git/classes/MySQLHandler.php on line 165: Error executing query: connect_errno: 0 errno: 1064 error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1 client_info: 5.1.73 host_info: Localhost via UNIX socket) Query: SELECT username FROM accounts WHERE username='' ; (0) [Exception]</pre>
Trace	<pre>#0 /owaspbwa/mutillidae-git/classes/MySQLHandler.php(283): MySQLHandler->doExecuteQuery('SELECT username...') #1 /owaspbwa/mutillidae-git/classes/SQLQueryHandler.php(250): MySQLHandler->executeQuery('SELECT username...') #2 /owaspbwa/mutillidae-git/includes/process-login-attempt.php(54): SQLQueryHandler->accountExists('') #3 /owaspbwa/mutillidae- git/index.php(277): include_once('/owaspbwa/mutillidae-git/classes/MySQLHandler.php') #4 {main}</pre>
Diagnostic Information	Error querying user account

2. Zalecenie

Należy uniemożliwić dostęp do informacji diagnostycznych nieupoważnionym osobom. Możliwe, że środowisko produkcyjne nie zostało odpowiednio przygotowane do użytku i umożliwia dostęp do informacji, do których normalnie nie powinno (np. pozostało w trybie DEBUG). W takiej sytuacji wszystkie logi z błędów powinny być kierowane przykładowo do pliku, do którego dostęp mają tylko upoważnione osoby.

3. Referencje

<https://dev.to/flippedcoding/difference-between-development-stage-and-production-d0p>

iii. [INFO]OWASP_REGISTER_5

1. Problem

Podczas rejestracji w serwisie możliwe jest pominięcie pól związanych z hasłem, przez co możliwe jest stworzenie konta, które będzie zawierało w bazie danych jedynie login.

Please choose your username, password and signature

Username

Password Password Generator

Confirm Password

Signature

Create Account

Account created for loremipsum. 1 rows inserted.

2. Zalecenie

Należy zmodyfikować kod HTML odpowiadający za stronę do rejestracji na serwisie tak, aby pola odpowiedzialne za login, hasło i potwierdzenie hasła były wymagane do wypełnienia. Za pomocą tego prostego kodu HTML można sprawić, że wszystkie pola będą wymagane do wypełnienia przed wysłaniem żądania do serwera.

```
<!DOCTYPE html>
<html>
<body>
<h1>Register</h1>
<form action="/action_page.php">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required>
  <label for="password">Password:</label>
  <input type="text" id="password" name="password" required>
  <label for="confirm">Confirm password:</label>
  <input type="text" id="confirm" name="confirm" required>
  <input type="submit" name="Submit" required>
</form>
</body>
</html>
```

Register

Username:

Password:

Confirm password:

Prześlij



Wypełnij to pole.

3. Referencje

<https://www.geeksforgeeks.org/html-input-required-attribute/>

https://www.w3schools.com/tags/att_required.asp

iv. [INFO]OWASP_LOGIN_REQUIRED_1

1. Problem

Podczas rejestracji w serwisie możliwe jest przeglądanie całej zawartości serwisu, w tym wszystkich danych poufnych, bez konieczności zalogowania się jako użytkownik.

2. Zalecenie

Większość serwisu, szczególnie ta zawierająca dane o użytkownikach lub ich wpisach powinna być niewidoczna lub niedostępna dla osoby niezalogowanej. Może to zapobiec wielu próbom niechcianego dostępu do danych, a także będzie pierwszym krokiem do utrudnienia rekonesansu serwisu.

`login_required()` does the following:

- If the user isn't logged in, redirect to `settings.LOGIN_URL`, passing the current absolute path in the query string. Example: `/accounts/login/?next=/polls/3/`.
- If the user is logged in, execute the view normally. The view code is free to assume the user is logged in.

3. Referencje

<https://techterms.com/definition/login>

<https://auth0.com/blog/should-you-make-your-users-login/>

<https://docs.djangoproject.com/en/3.1/topics/auth/default/>

v. [INFO]OWASP_PASSWORD_STORAGE_1

1. Problem

Baza danych przechowuje hasła w tzw. plain text (zwykły, niezaszyfrowany tekst). Jest to skrajnie nieodpowiedzialny sposób przechowywania poufnych danych, które służą w procesie uwierzytelniania.

```
Username=admin  
Password=admin  
Signature=4  
  
Username=somepassword  
Password=admin  
Signature=4  
  
Username=monkey  
Password=john  
Signature=4  
  
Username=password  
Password=jeremy  
Signature=4  
  
Username=password  
Password=bryce  
Signature=4  
  
Username=samurai  
Password=samurai  
Signature=4
```

2. Zalecenie

Należy zastosować mechanizm szyfrujący hasła przed zapisem do bazy danych.

3. Referencje

<https://www.passcamp.com/blog/dangers-of-storing-and-sharing-passwords-in-plaintext/>

<https://www.sciencedirect.com/topics/computer-science/plaintext-password>

vi. [INFO]OWASP_PASSWORD_STORAGE_2

1. Problem

Wyliczone funkcje skrótu haseł nie są dodatkowo wspomagane przez sól ani pieprz, przez co są podatne na atak tęczyowych tablic. W łatwy sposób można zidentyfikować także posiadanie tego samego hasła przez różnych użytkowników.

Username=A.I.M. or Authentication Is Missing
Password=A.I.M.
Signature=~~0005050400101010c5000c705d004571045aff10~~

Username=Any bugs?
Password=bee
Signature=~~0005050400101010c5000c705d004571045aff10~~

2. Zalecenie

Należy zastosować solenie oraz pieprzenie haseł.

3. Referencje

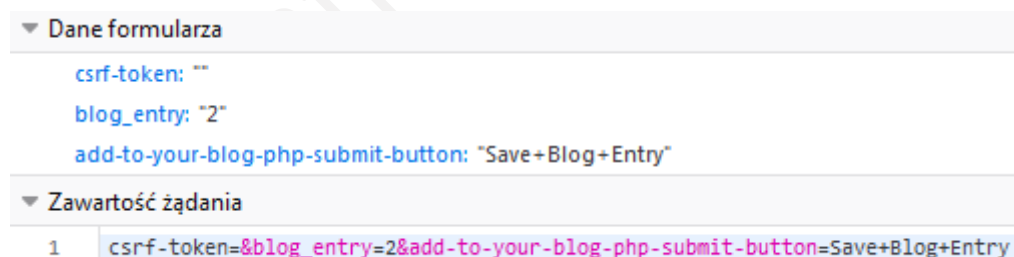
<https://sekurak.pl/kompendium-bezpieczenstwa-hasel-atak-i-obrona/>

<https://medium.com/@berto168/salt-pepper-spice-up-your-hash-b48328caa2af>

vii. [INFO]OWASP_CSRF_TOKEN_1

1. Problem

Token CSRF przesyłany w żądaniach jest pusty bądź stały dla wszystkich użytkowników, przez co serwis jest podatny na ataki typu CSRF. Dla przykładu żądanie HTTP wysyłane podczas dodawania wpisu na bloga zawiera następujące ciało wiadomości:

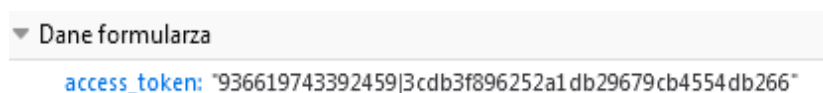


Jak można zauważyć, token CSRF jest pusty.

2. Zalecenie

Należy zastosować tokenizację żądań, tak, aby uniemożliwić atakującemu wykonanie działań wykorzystujących uprawnienia ofiary. Jednym z powszechnie zalecanych rozwiązań jest wykorzystanie tzw. Middleware (oprogramowanie pośredniczące).

Przykładowe zastosowanie:



3. Referencje

https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

<https://www.netsparker.com/blog/web-security/csrf-cross-site-request-forgery/>

<https://docs.djangoproject.com/en/3.1/ref/csrf/>

viii. [INFO]OWASP_HTTP_METHODS_1

1. Problem

Żądanie HTTP GET jest w stanie doprowadzić do efektów ubocznych na serwerze i nie powinno służyć przekazywaniu informacji na serwer, a jedynie ich pozyskiwaniu.

```
▼ GET
Scheme: http
Host: 192.168.56.101
Filename: /mutillidae/index.php

page: user-poll.php
csrf-token:
choice: nmap
initials: 111
user-poll-php-submit-button: Submit Vote
```

W tym przypadku zostaje wysłane żądanie z nagłówkiem GET, które sprawia, że w ankiecie zostaje oddany głos.

2. Zalecenie

Metoda GET powinna służyć jedynie pozyskaniu informacji. W przypadku ankiety, żądanie powinno jedynie zwrócić stan ankiety, a nie prowadzić do uczestnictwa w niej. Dane przekazywane w ten sposób są niezabezpieczone i umieszczone w adresie URL.

Do przesyłania informacji należy wykorzystać inne metody HTTP takie, jak np. POST czy PUT.

3. Referencje

<https://searchsoftwarequality.techtarget.com/tip/Why-use-POST-vs-GET-to-keep-applications-secure>

<https://www.fullcontact.com/blog/2016/04/29/never-put-secrets-urls-query-parameters/>

ix. [INFO]OWASP_HTTP_METHODS_2

1. Problem

Dane przekazywane w ciele żądania nie są w żaden sposób szyfrowane przed nieupoważnionym odczytem w drodze od klienta do serwera.

```
POST /mutillidae/index.php?page=register.php HTTP/1.1
Host: 192.168.56.101
Content-Length: 135
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.56.101
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/87.0.4280.88 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://192.168.56.101/mutillidae/index.php?page=register.php
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: showhints=1; PHPSESSID=jag93h8eov7ckphbhtld9sq5p03
Connection: close

csrf-token=&username=aaaaaaaa&password=bbbbbbb&confirm_password=bbbbbbb&my_signature=cccccccc&register-php-submit-button=Create+Account|
```

Najbardziej wrażliwe informacje w tym żądaniu, czyli hasło i jego potwierdzenie, są przekazywane w ciele wiadomości zupełnie niezaszyfrowane, dzięki czemu atakujący, który aktualnie nasłuchuje ruchy sieciowe ofiary (np. poprzez bramkę Proxy), jest w stanie je odczytać i uzyskać dostęp do konta ofiary, która właśnie się zarejestrowała.

2. Zalecenie

Dobłą praktyką jest wykorzystywanie szyfrowanej wersji protokołu HTTP – HTTPS.

Oprócz tego można wykorzystać dodatkowe szyfrowanie pomiędzy klientem a serwerem, np. z użyciem szyfrowania asymetrycznego – szyfrowanie u klienta odbywa się z pomocą klucza publicznego, zaś deszyfrowanie po stronie serwera z wykorzystaniem klucza prywatnego.

W przypadku wykorzystania szyfrowania HTTPS, atakujący zamiast uzyskać żądanie tej postaci:

```
GET /hello.txt HTTP/1.1

User-Agent: curl/7.63.0 libcurl/7.63.0 OpenSSL/1.1.1 zlib/1.2.11

Host: www.example.com

Accept-Language: en
```

otrzymuje ciąg podobnej postaci:

```
t8Fw6T8UV81pQfyhDkhebbz7+oiwldr1j2gHBB3L3RfTRsQCpaSnSBZ78Vme+DpDVJ
PvZdZUZHPzbbcmSW1+3xXGseERHg9YDmpYk0VVDiRvw1H5miNieJeJ/FNUjgH0BmVR
WII6+T4MnDwmCMZUI/orxP3HGwYCSIvyzS3MpmSe4iaWKC0HQ==
```

3. Referencje

<https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/>