PHP – bezpieczeństwo aplikacji

Uwierzytelnianie HTTP

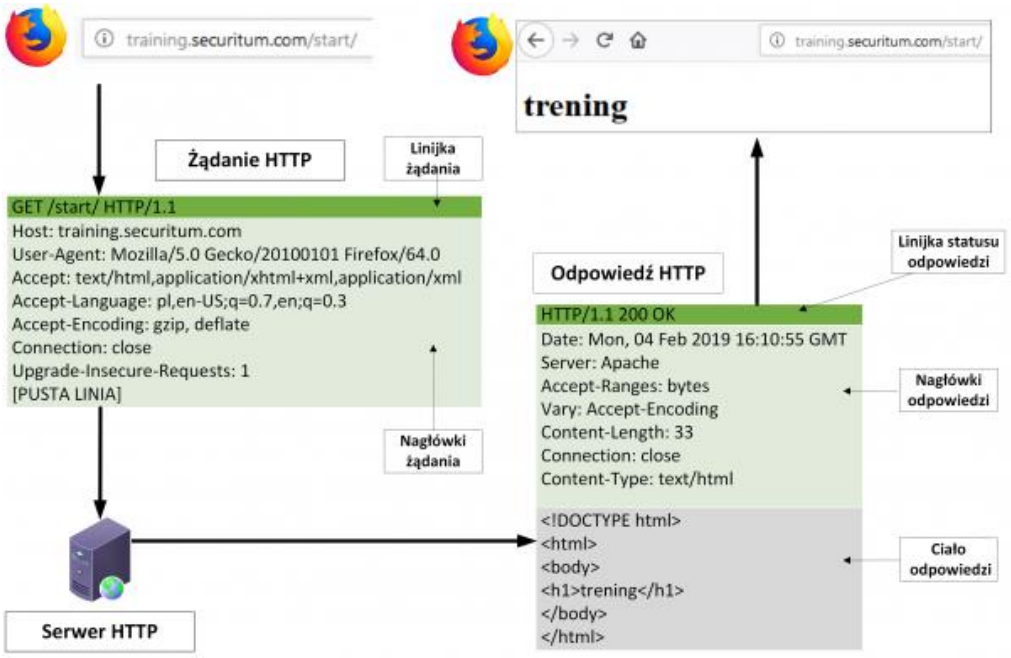
Uwierzytelnianie HTTP to prosty sposób na zabezpieczenie strony za pomocą kodu PHP. Polega na użyciu do ochrony strony administracyjnej hasła. Jeśli strona jest zabezpieczona przy użyciu uwierzytelniania HTTP, przed jej udostępnieniem pojawia się okno z prośbą o podanie nazwy użytkownika i hasła.

Kod PHP jest potrzebny do sprawdzenia nazwy i hasła podanych przez użytkownika. Dane te znajdują się w zmiennej superglobalnej $_SERVER. Skrypt PHP potrafi sprawdzić nazwę i hasło wpisane przez użytkownika oraz ustalić, czy należy przyznać dostęp do zabezpieczonej strony.

Uwierzytelnianie HTTP wymaga nagłówków

Działanie uwierzytelniania HTTP polega na tym, że serwer przed udostępnieniem chronionej strony prosi przeglądarkę o pobranie nazwy oraz hasła od użytkownika. Jeśli internauta poda prawidłowe dane, przeglądarka prześle stronę. Komunikacja między przeglądarką i serwerem odbywa się za pomocą nagłówków. Są to krótkie komunikaty tekstowe ze specjalnymi instrukcjami, które opisują żądane i dostarczane dane.

Komunikacja HTTP realizowana jest poprzez wysłanie żądania (request) do serwera, który następnie generuje odpowiedź (response).



Źródło: https://sekurak.pl/protokol-http-podstawy/

Kontrolowanie nagłówków za pomocą PHP

Przy użyciu PHP można kontrolować nagłówki przesyłane przez serwer do przeglądarki, co umożliwia wykonywanie opartych na nagłówkach zadań, na przykład uruchamianie uwierzytelniania HTTP. Do wysyłania nagłówków z serwera do przeglądarki z poziomu skryptu PHP służy wbudowana funkcja `header()`:

```
header('Content-Type: text/html');
```

Funkcję `header()` trzeba wywołać przed wysłaniem głównych danych. Jest to bardzo ścisły wymóg — jeśli przed nagłówkiem pojawi się choć jeden znak lub odstęp, przeglądarka odrzuci dane i zgłosi błąd. Dlatego wywołanie funkcji `header()` w skrypcie PHP musi znajdować się przed kodem HTML.

```php
<?php
header('Content-Type: text/html');
...
?>
<html lang="pl">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="Content-Language" content="pl" />
<title>Tytuł strony</title>
</head>
<body>
...
</body>
</html>
```

Uwierzytelnianie za pomocą nagłówków

Uwierzytelnianie strony administracyjnej przy użyciu nagłówków wymaga przygotowania bardzo specyficznego zestawu dwóch nagłówków, który informuje przeglądarkę, że przed udostępnieniem powinna poprosić użytkownika o nazwę i hasło. Do wygenerowania tych dwóch nagłówków użyjemy kodu PHP w skrypcie administracyjnym. Posłużą one do kontrolowania procesu przesyłania strony do przeglądarki.

HTTP/1.1 401 Unauthorized

WWW-Authenticate: Basic realm="Technik informatyk"

Po przetworzeniu nagłówków związanych z uwierzytelnianiem przeglądarka oczekuje na wpisanie danych przez użytkownika w oknie uwierzytelniania. W zależności od zachowania użytkownika:

Jeśli użytkownik poda prawidłową parę nazwa – hasło i kliknie przycisk Zaloguj się, serwer prześle do przeglądarki kod HTML strony admin.php. Przeglądarka wyświetli stronę administracyjną, a użytkownik będzie miał dostęp do strony.

Jeśli użytkownik poda błędną parę nazwa – hasło i kliknie przycisk Zaloguj się, serwer nakaże przeglądarce ponowne wyświetlenie prośby o dane. Przeglądarka będzie powtarzać ten proces, dopóki użytkownik będzie wpisywał nieprawidłowe informacje. Oznacza to, że jeżeli użytkownik nie zna nazwy i hasła, jedynym sposobem na zamknięcie okienka jest kliknięcie przycisku Anuluj.

Jeśli użytkownik kliknie przycisk Anuluj, aby pominąć uwierzytelnianie, serwer prześle do przeglądarki wyłącznie komunikat o braku dostępu — nie udostępni strony admin.php. Treść komunikatu znajduje się w kodzie PHP skryptu admin.php, który jest ściśle związany z nagłówkami. Ten kod wywołuje funkcję `exit()` języka PHP, która wyświetla wiadomość i kończy działanie skryptu:

```
exit('<h2>Technik informatyk</h2>Musisz wprowadzić
prawidłową parę '
. 'nazwa użytkownika – hasło, aby uzyskać dostęp do tej
strony.');
```

```php
<?php
// Nazwa użytkownika i hasło używane przy uwierzytelnianiu.
$userame = 'rock';
$password = 'roll';
if (!isset($_SERVER['PHP_AUTH_USER']) || !isset($_SERVER['PHP_AUTH_PW']) ||
($_SERVER['PHP_AUTH_USER'] != $username) || ($_SERVER['PHP_AUTH_PW'] != $password ))
{
 // Para nazwa – hasło jest błędna, dlatego należy przesłać nagłówki związane z uwierzytelnianiem.
header('HTTP/1.1 401 Unauthorized');
header('WWW-Authenticate: Basic realm= "Technik informatyk"');
exit('<h2>Gitarowe wojny</h2> Musisz wprowadzić prawidłową parę nazwa ' .
'użytkownika – hasło, aby uzyskać dostęp do tej strony.');
}
?>

<html lang="pl">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="Content-Language" content="pl" />
<title>Tytuł strony</title>
</head>
<body>
...
</body>
</html>
```

Problem z zabezpieczeniami…tworzenie skryptu do obsługi uwierzytelniania

Zabezpieczenie samej strony administracyjnej nie wystarczyło, ponieważ — dzięki odpowiednim informacjom — można uzyskać bezpośredni dostęp do skryptu np. z wynikami.

http://www.technikinformatyk.net.pl/removescore.php?id=10&name=Jakub%20Szczepaniak&date=2008-05-01%2020:36:45&score=3&screenshot=wynikjakuba.gif

Rozwiązaniem jest umieszczenie tego same kodu z nagłówkami uwierzytelniania w skrypcie z wynikami, a aplikacja będzie bezpieczna.

Kod z pliku `admin.php` kopiujemy do nowego pliku np. autoryzacja.php i zastąpić skopiowany fragment instrukcją require_once.

```
<?php
require_once('autoryzacja.php');
?>
<html>
```

Notatka do zeszytu

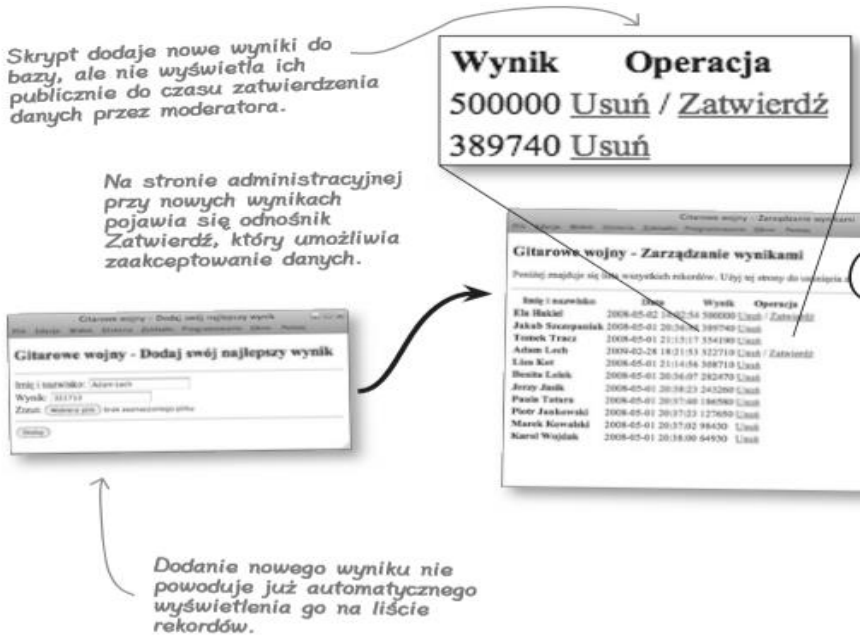
1. Uwierzytelnianie HTTP to prosty sposób na zabezpieczenie strony za pomocą kodu PHP.
2. W skryptach PHP można używać nagłówków do kontrolowania przesyłania stron przez serwer do przeglądarki.
3. Wbudowana funkcja **`header()`** języka PHP wysyła do przeglądarki nagłówki. Można ich użyć do przekierowania użytkownika do innej strony, określenia zawartości strony lub zażądania uwierzytelnienia.
4. Jeśli skrypt przesyła nagłówki do przeglądarki za pomocą funkcji **`header(),`** jej wywołanie należy umieścić przed kodem wysyłającym inne dane.
5. Jeśli strona jest chroniona uwierzytelnianiem HTTP, nazwa i hasło podane przez użytkownika są zapisywane w zmiennej superglobalnej **$_SERVER**.
6. Atrybut **`Basic realm`** w uwierzytelnianiu HTTP określa strefę bezpieczeństwa powiązaną z określoną parą danych nazwa – hasło. Umożliwia to wspólne zabezpieczenie grupy stron.
7. Wbudowana funkcja **`exit()`** języka PHP kończy działanie skryptu PHP i zapobiega wykonaniu dalszego kodu oraz przesłaniu następnych danych do przeglądarki.

Moderator

Nawet we współczesnym świecie czasem nie da się zastąpić ludzi i ich umysłów. Nic lepiej niż człowiek nie przeanalizuje informacji i nie oceni, czy wynik jest prawdziwy, czy nie. Mówimy tu o **moderowaniu**. Proces ten polega na zatwierdzaniu przesłanych do aplikacji sieciowej informacji przez człowieka przed ich publicznym udostępnieniem.

Ludzki moderator to doskonały sposób na sprawdzanie jakości danych przesyłanych przez użytkowników.

	#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne
☐	1	**id** 🔑	int(11)			Nie	*Brak*
☐	2	**date**	timestamp			Nie	current_timestamp()
☐	3	**name** 🔗	varchar(32)	utf8_polish_ci		Tak	*NULL*
☐	4	**score**	int(11)			Tak	*NULL*
☐	5	**screenshot**	varchar(64)	utf8_polish_ci		Tak	*NULL*
☐	6	**approved**	tinyint(1)			Tak	0



Wstrzykiwanie kodu SQL

Jest to technika, w której napastnik używa danych z formularza do zmiany działania zapytania. Zamiast podawać w polu zwykłe informacje, na przykład imię i nazwisko lub wynik, wpisuje fragment instrukcji SQL.

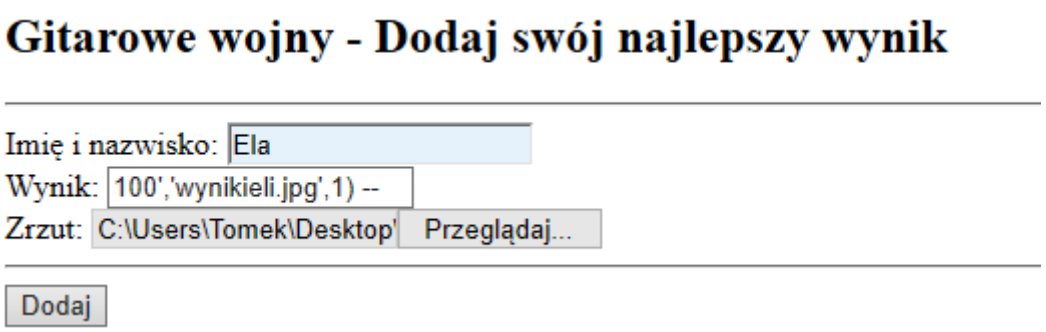
Pola formularza to słaby punkt zabezpieczeń w aplikacjach sieciowych, ponieważ umożliwiają wpisywanie danych przez użytkowników.

Aby zrozumieć, jak przeprowadza się ten sprytny atak, przyjrzyjmy się drodze danych z formularza do skryptu do dodawania wyników.

W polu Wynik powinna pojawić się pojedyncza liczba, na przykład 1000000, jednak zamiast tego znajduje się tu kilka wartości umieszczonych w cudzysłowach i rozdzielonych przecinkami, po których następują dwa myślniki.

```
INSERT INTO guitarwars
VALUES(0,NOW(),’Ela’,’1000000’,’wynikieli.jpg’,1)–
‘,’wynikieli2.jpg’,0)
```

Oszukiwanie bazy MySQL za pomocą komentarzy. Dwa myślniki (--) sprawiają, że dalsza część wiersza zostanie uznana za komentarz. Po dwóch myślnikach trzeba dodać odstęp (--), aby kod wykrył komentarz, a cały dalszy kod jest pomijany.



```
INSERT INTO guitarwars
VALUES(0,NOW(),'Ela','1000000','wynikieli.jpg',1)–
','wynikieli2.jpg',0)
```

Inna wersja komentarzy jednowierszowych wymaga użycia znaku **#** zamiast sekwencji **--** i też powoduje oznaczenie dalszego kodu w wierszu jako komentarza. SQL obsługuje też komentarze wielowierszowe. Do ich tworzenia — podobnie jak w języku PHP — służą sekwencje **/*** i ***/**.

Ochrona danych przed wstrzyknięciem kodu SQL.

Prawdziwą słabością jest brak walidacji zawartości pól formularza pod kątem niebezpiecznych znaków. Te znaki to dowolne symbole, które mogą zmienić działanie zapytania SQL: przecinki, cudzysłowy lub sekwencja komentarza (--). Nawet odstępy po danych mogą okazać się szkodliwe. Początkowe lub końcowe spacje można łatwo wyeliminować za pomocą wbudowanej funkcji **trim()** języka PHP. Wystarczy przekazać do niej wszystkie dane z formularza przed użyciem ich w zapytaniu SQL.

```
$name = trim($_POST['name']);
$score = trim($_POST['score']);
$screenshot = trim($_FILES['screenshot']['name']);
```

Funkcja **`trim()`** usuwa początkowe i końcowe odstępy z danych z formularza.

Jednak początkowe i końcowe odstępy to nie cały problem. Trzeba pamiętać też o przecinkach, cudzysłowach, znakach komentarza i wielu innych.

PHP udostępnia w tym celu inną wbudowaną funkcję, **`mysqli_real_escape_string()`**, która dodaje znaki ucieczki, aby potencjalnie niebezpieczne symbole nie zmieniły działania zapytania. Te symbole pojawią się w polach formularza, jednak nie będą miały wpływu na funkcjonowanie zapytania.

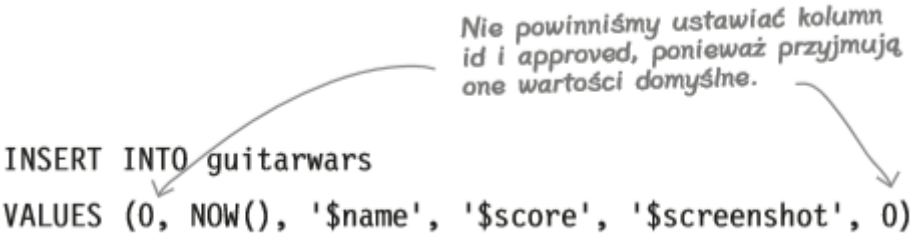
```
$name = mysqli_real_escape_string($dbc, trim($_POST['name']));
$score = mysqli_real_escape_string($dbc, trim($_POST['score']));
$screenshot = mysqli_real_escape_string($dbc,
trim($_FILES['screenshot']['name']));
```

Przetwarzanie zawartości pól formularza aplikacji Gitarowe wojny za pomocą funkcji trim() i mysqli_real_escape_string() znacznie zmniejsza prawdopodobieństwo wstrzyknięcia kodu SQL. Jednak te dwie funkcje nie dają pełnego bezpieczeństwa, a jedynie zmniejszają wrażliwość zapytania na ataki.

Bezpieczniejsza instrukcja INSERT (z parametrami)

Oprócz słabych zabezpieczeń pól formularza przy wstrzyknięciu kodu SQL istotny jest fakt, że kolumna approved znajduje się w strukturze tabeli po kolumnie

screenshot. Dlatego wystarczyło dodać 1 w końcowej części zapytania INSERT, a instrukcja zapisała tę wartość w kolumnie approved. Problem polega na tym, że zapytanie to wstawia dane do wszystkich kolumn, co niepotrzebnie zwiększa ryzyko.



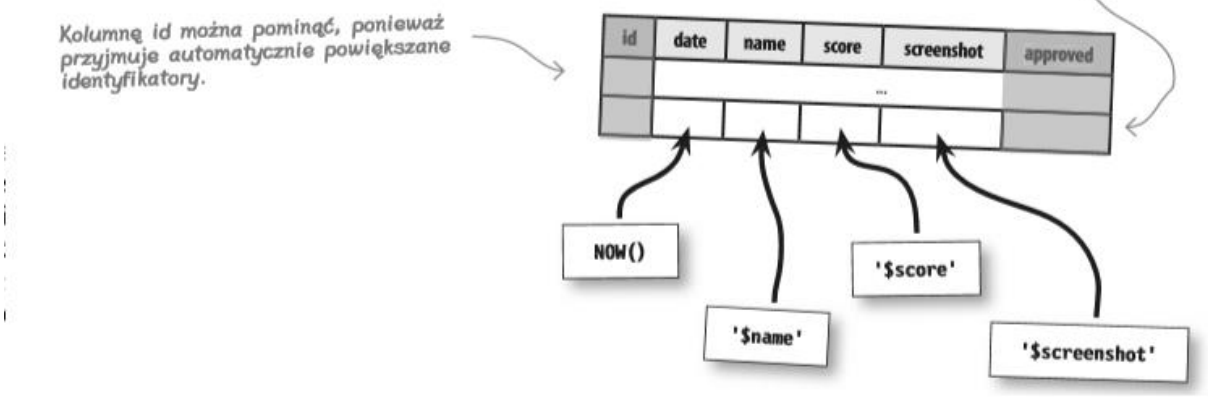
Nie trzeba bezpośrednio wstawiać danych w kolumnach id i approved, ponieważ pierwsza z nich przyjmuje automatycznie powiększane identyfikatory, a druga zawsze powinna mieć wartość 0. Lepsze podejście polega na wstawianiu tylko danych niezbędnych w nowym wierszu. Kolumny id i approved przyjmą wtedy wartości domyślne — AUTO_INCREMENT i 0.

Ta wersja instrukcji INSERT dokładnie określa, w której kolumnie zostaną zapisane poszczególne wartości. Pozwala to wstawić dane bez uwzględniania struktury tabeli. Ta postać zapytań INSERT jest zalecana, ponieważ zapisuje dane w precyzyjnie wskazanych kolumnach, a nie na podstawie założeń dotyczących układu tabeli.

```
INSERT INTO guitarwars (date, name, score, screenshot)
VALUES (NOW(), '$name', '$score', '$screenshot')
```

Nie można wstawić żadnej wartości do kolumny approved, ponieważ nie jest ona częścią zapytania.

Kolumnę id można pominąć, ponieważ przyjmuje automatycznie powiększane identyfikatory.



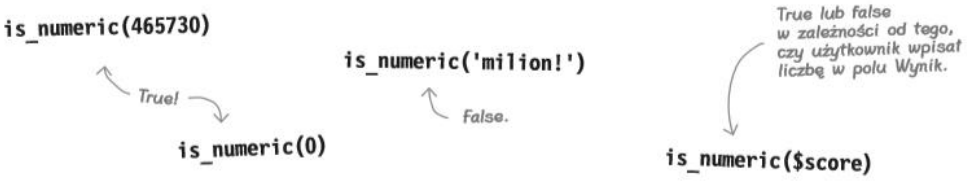
Walidacja formularzy

Ostatni etap minimalizowania ryzyka wstrzyknięcia kodu SQL polega na walidacji formularza w skrypcie do dodawania wyników. Przed ustaleniem, czy typ i rozmiar pliku ze zrzutem są zgodne z wymogami, należy się upewnić, że trzy pola formularza nie są puste:

```
if (!empty($name) && !empty($score) && !empty($screenshot)) {
  ...
}
```

Ta instrukcja if sprawdza, czy wszystkie pola formularza są wypełnione.

Ponieważ w polu Wynik powinna pojawić się liczba, warto sprawdzić nie tylko to, czy pole jest puste, ale też czy zawiera dane odpowiedniego rodzaju. Umożliwia to funkcja **`is_numeric()`** języka PHP, która zwraca true, jeśli podana wartość to liczba (w przeciwnym razie zwraca false).



Poniższa instrukcja warunkowa if obsługuje walidację w formularza do dodawania wyników. Użyta w niej funkcja is_numeric() pilnuje aby użytkownik mógł wpisać tylko liczbę.

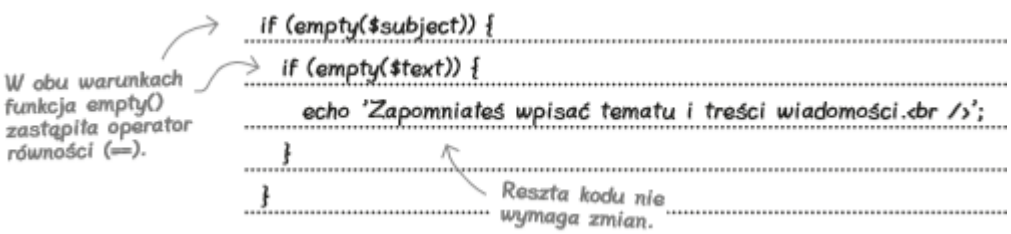
```
if (!empty($name) && is_numeric($score) && !empty($screenshot)) {
...
}
```

Inne funkcje PHP do sprawdzania zawartości zmiennych

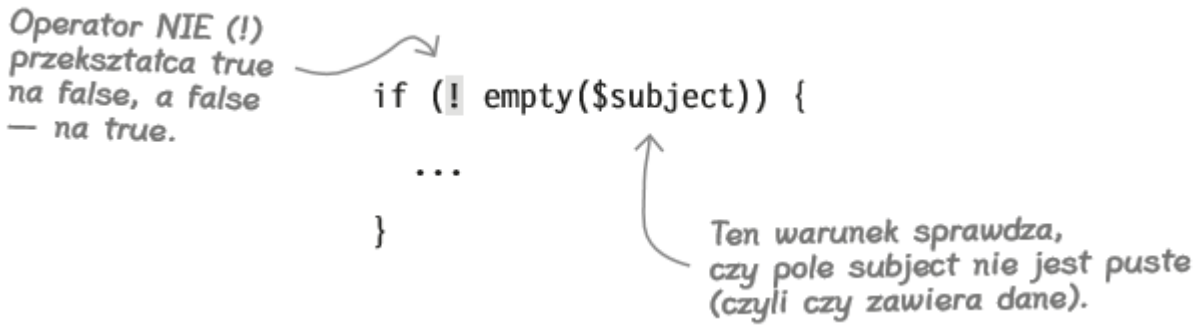
Do wykrycia pustych łańcuchów można użyć operatora ==, jednak istnieje lepsze narzędzie — wbudowane funkcje PHP. Funkcja **`isset()`** sprawdza, czy zmienna istnieje (a dokładniej — czy przypisano jej wartość). Funkcja **`empty()`** określa, czy zmienna zawiera wartość pustą. W PHP odpowiadają jej: 0, pusty łańcuch znaków (‘’ lub ””) oraz wartości false i NULL. Dlatego funkcja isset() zwraca true tylko wtedy, jeśli kod przypisał zmiennej wartość, a empty() — jeżeli zmienna jest pustym łańcuchem lub ma wartość 0, false albo NULL.

Dlatego skrypt, który przypisuje zmiennym $subject i $text wartości $_POST[‘subject’] i $_POST[‘elvismail’] z tablicy superglobalnej. Dlatego jeśli sprawdzimy je w funkcji isset(), zawsze otrzymamy wartość true (także wtedy, jeżeli zmienne nie będą zawierały tekstu). Oznacza to, że funkcja ta nie odróżnia pustego pola formularza od wypełnionego. Natomiast funkcja empty() potrafi wskazać puste zmienne, czego potrzebujemy do walidacji formularza.

Aby sprawdzić, czy formularz zawiera dane: temat i treść listu. Należy użyć skryptu.



Jeżeli natomiast chcemy sprawdzić warunek przeciwny. Na przykład przed wysłaniem wiadomości z danymi z formularza warto ustalić, czy pola nie są puste. Wykorzystujemy do tego celu negację.



Szyfrowanie haseł

Szyfrowanie w aplikacji polega na przekształceniu hasła na niezrozumiały format przy jego zapisywaniu w bazie. W każdej witrynie z obsługą logowania należy szyfrować hasła, aby użytkownicy mieli pewność, że ich dane są bezpieczne. Udostępnianie haseł — nawet w bazie danych — jest niedopuszczalne.

MySQL udostępnia funkcję SHA(), która służy do szyfrowania łańcuchów znaków. Efekt jej działania to zaszyfrowany łańcuch, który zawsze — niezależnie od długości hasła — zawiera 40 znaków w systemie szesnastkowym. Funkcja ta generuje 40-znakowy kod, który jest jednoznaczną reprezentacją hasła.

Tej samej funkcji SHA() używamy także przy logowaniu, aby sprawdzić, czy hasło wprowadzone przez użytkownika jest zgodne z zaszyfrowanym tekstem z bazy danych.

Funkcja SHA() działa tylko w jedną stronę i nie umożliwia odkodowania hasła. Pozwala to zapewnić bezpieczeństwo zaszyfrowanych danych. Nawet jeśli

napastnik włamie się do bazy i ukradnie wszystkie hasła, nie zdoła ich odszyfrować.

Nie musimy znać pierwotnego hasła, aby ustalić, czy użytkownik wpisał je poprawnie. Jest to możliwe, ponieważ funkcja SHA() generuje dla danego łańcucha zawsze ten sam 40-znakowy kod. Dlatego wystarczy zaszyfrować hasło podane przez użytkownika w czasie logowania i porównać kod z wartością kolumny przechowującej hasło.

MySQL udostępnia również funkcję MD5(), która szyfruje dane w podobny sposób jak funkcja SHA(). Jednak algorytm funkcji SHA() jest uważany za bezpieczniejszy.

```
INSERT INTO mismatch_user  (username, password, join_date) VALUES ('jimi', SHA('heyjoe'), NOW())
```

	user_id	username	password	join_date	first_name	last_name	gender	birthdate	city	state	picture
Edytuj Kopiuj Usuń	1	majak	745c52f30f82d4323292dcca9eea0aee87feecc5	2008-06-03 14:51:46	Maja	Kozak	K	1984-07-19	Szczecin	ZP	zdjmai.jpg
Edytuj Kopiuj Usuń	2	marekw	12a20bcb5ed139a5f3fc808704897762cbab74ec	2008-06-03 14:52:09	Marek	Wysocki	M	1973-05-13	Opole	OP	zdjmarka.jpg
Edytuj Kopiuj Usuń	3	olek	676a6666682bd41bef5fd1c1f629fa233b1307a4	2008-06-03 14:53:05	Olek	Kulak	M	1974-09-13	Gliwice	SL	zdjolka.jpg
Edytuj Kopiuj Usuń	4	zdanielak	1ff915f2fae864032e44cbe5a6cdd858500c9df7	2008-06-03 14:58:40	Zuzanna	Danielak	K	1977-02-23	Katowice	SL	zdjzuzanny.jpg
Edytuj Kopiuj Usuń	5	elahak	53a56acb2a52f3815a2518e75029b071c298477a	2008-06-03 15:00:37	Ela	Hakiel	K	1943-03-27	Radom	MZ	zdjeli.jpg
Edytuj Kopiuj Usuń	6	czarny	df00f36c0b795c30a0409778d7f9db27a970f74f	2008-06-03 15:00:48	Oskar	Czarnecki	M	1968-06-04	Warszawa	MZ	zdjoskara.jpg
Edytuj Kopiuj Usuń	7	belital	7c19dd287e03ae31ce190c4043b5a7f9795c41dc	2008-06-03 15:01:08	Benita	Lelek	K	1975-07-08	Kalisz	WP	zdjbenity.jpg
Edytuj Kopiuj Usuń	8	jacekf	3da70cd115b7c3a7cebc2b5282706f07d185de9e	2008-06-03 15:01:19	Jacek	Filipowski	M	1969-09-24	Olsztyn	WM	zdjjacka.jpg
Edytuj Kopiuj Usuń	9	gabrysia	08447be571e1c113f2f175472753ca5f5af454f3	2008-06-03 15:01:51	Gabriela	Pietruszka	K	1970-04-26	Konin	WP	zdjgabrysi.jpg
Edytuj Kopiuj Usuń	10	lysyjarek	230dcb28e2d1dc19ec14990721e85cd5c5234245	2008-06-03 15:02:02	Jarek	Hill	M	1964-12-18	Krosno	PK	zdjjarka.jpg
Edytuj Kopiuj Usuń	11	jnawrocki	e511d793f532dbe0e0483538e11977f7b7c33b28	2008-06-03 15:02:13	Jan	Nawrocki	M	1981-11-03	Gdynia	PM	zdjjana.jpg
Edytuj Kopiuj Usuń	12	alar	062e4a8476b1063e05caa69958e36a905f887619	2008-06-03 15:02:24	Ala	Rudzik	K	1972-09-18	Kielce	SK	zdjali.jpg

Skrypt logowania HTTP

Notatka do zeszytu:

1. **Moderowanie** - proces ten polega na zatwierdzaniu przesłanych do aplikacji sieciowej informacji przez człowieka przed ich publicznym udostępnieniem.
2. **Wstrzykiwanie kodu SQL** jest to technika, w której napastnik używa danych z formularza do zmiany działania zapytania.
3. **Ochrona danych przed wstrzyknięciem kodu SQL:**
   - Funkcja **`trim()`** usuwa początkowe i końcowe odstępy z danych z formularza.
   - Funkcja **`mysqli_real_escape_string()`**, która dodaje znaki ucieczki, aby potencjalnie niebezpieczne symbole nie zmieniły działania zapytania.
4. **Walidacja formularzy:**
   - Funkcja **`isset()`** sprawdza, czy zmienna istnieje (a dokładniej — czy przypisano jej wartość). Funkcja isset() zwraca true tylko wtedy, jeśli kod przypisał zmiennej wartość.
   - Funkcja **`empty()`** określa, czy zmienna zawiera wartość pustą. W PHP odpowiadają jej: 0, pusty łańcuch znaków (‘’ lub ””) oraz wartości false i NULL. Funkcja empty() zwraca prawdę jeżeli zmienna jest pustym łańcuchem lub ma wartość 0, false albo NULL.
   - funkcja **`is_numeric()`** – sprawdza czy podana wartość jest liczbą.
5. „SHA” to akronim od angielskiego Secure Hash Algorithm, czyli „bezpieczny algorytm generowania skrótów”. „Skrót” w słownictwie technicznym oznacza niepowtarzalny łańcuch znaków o stałej długości, który stanowi jednoznaczną reprezentację danego tekstu. Funkcja SHA() tworzy skrót w postaci 40-znakowego zaszyfrowanego łańcucha w systemie szesnastkowym, który jednoznacznie reprezentuje pierwotne hasło.
6. Funkcja SHA() obsługuje szyfrowanie jednostronne. Zakodowanych w ten sposób danych nie można odszyfrować.

Źródło: Head First PHP & MySQL Lynn Beighlley, Michael Morrison