Web exploitation: Ένα πραγματικό παράδειγμα

Επιμέρους θέματα/προβλήματα:

(1) Παράκαμψη προστασίας SQL Injection

(2) Advanced UPDATE SQL Injection how-to

By SolidSNK solidsnk@grhack.net

Εισαγωγικές έννοιες

- Input Validation
- Code Injection & SQL Injection

By SolidSNK solidsnk@grhack.net

Input Validation

Ίσως ο κρισιμότερος παράγοντας στη θωράκιση λογισμικού είναι το *input validation*. Πρόκειται για τη επεξεργασία των δεδομένων που εισάγει ο χρήστης προκειμένου να διασφαλιστεί ότι δεν είναι επιβλαβή. Αρχικά, οι στρατηγικές της εξακρίβωσης είναι δύο:

- *Whitelist*, όπου συγκροτείται λίστα με το επιτρεπτό input.
- *Blacklist*, όπου η λίστα περιλαμβάνει το “κακό” input.

Στην περίπτωση “κακού” input, υπάρχουν οι εξής επιλογές:

- Τερματισμός της εφάρμογής ή απόρριψη των δεδομένων.
- Φιλτράρισμα/αντικατάσταση των κρίσιμων χαρακτήρων ή ακολουθιών.

Ανεπαρκές input validation μπορεί να οδηγήσει σε χρησιμοποίηση της εφαρμογής με κακόβουλο τρόπο ή ακόμα και σε ολοκληρωτική κατάληψη της.

By SolidSNK solidsnk@grhack.net

Code Injection

- Ευρύ είδος επίθεσης κατά το οποίο ο κακόβουλος χρήστης τοποθετεί στην εφαρμογή κάποιου είδους κώδικα. Ο τρόπος που θα ερμηνευτεί και θα εκτελεστεί από την εφαρμογή εξαρτάται από τη φύση της εφαρμογής και του σφάλματος. Για παράδειγμα, ένα σφάλμα *php remote file inclusion* (*RFI*) μπορεί να οδηγήσει στην εκτέλεση κώδικα php στη μεριά του server, ενώ σε μια *XSS* επίθεση ο browser του client είναι αυτός που εκτελεί javascript κώδικα.
- Ένα είδος Code Injection επίθεσης είναι η *SQL Injection*. Συναντάται σε εφαρμογές του ιστού κυρίως, όπου η εφαρμογή αλληλεπιδρά με μια βάση δεδομένων. Σε αυτή την περίπτωση ο επιτιθέμενος , λόγω ελλειπούς input validation, τοποθετεί (“injects”) κακόβουλο κώδικα SQL σε μια αίτηση (query). Η παραποιημένη query θα εκτελεστεί από τη βάση δεδομένων, με ποικίλα αποτελέσματα. Η σωστή σύνταξη του SQL κώδικα που θα τοποθετηθεί καθώς και το αποτέλεσμα της επίθεσης εξαρτάται από τη μορφή της query που θα εκτελεστεί. Συνήθως η malicious query επιστρέφει δεδομένα από τη βάση, αλλά δε λείπουν επιθέσεις κατά τις οποίες μεταβάλλεται ένα ή περισσότερα records σε αυτή. Η SQL Injection ανήκει με τη σειρά της σε ένα σύνολο τεχνικών όπου μία γλώσσα προγραμματισμού εντίθεται μέσα σε μία άλλη και υπάρχει ελλειπής έλεγχος κατά την αλληλεπίδραση τους.

By SolidSNK solidsnk@grhack.net

Παράδειγμα μιας τυπικής SQL Injection

Το παραδειγμα που ακολουθεί είναι μια τυπική περίπτωση SQL Injection στο `WHERE` τμήμα ενός `SELECT` statement. Έστω ότι έχουμε τον παρακάτω κώδικα php στο /index.php του www.isuck.org (παρατηρείστε την απουσία input validation):

```
$de = mysql_query("SELECT de FROM news WHERE id={$_GET['id']}");
show_description_from_result($de);
```

Είναι προφανές πως το αναμενόμενο input είναι ένας θετικός ακέραιος. Αν υποθέσουμε πως στη βάση που χρησιμοποιεί η εφαρμογή υπάρχει ένα table “users” με μία column “email” , τότε ζητώντας το παρακάτω URI:

```
http://www.isuck.org/index.php?id=-1+UNION+ALL+SELECT+email+FROM
+users+LIMIT+1
```

Η MySQL τελικά θα εκτελέσει την query:

```
SELECT de FROM news WHERE id=-1 UNION ALL SELECT email FROM users
LIMIT 1
```

… άρα εκεί όπου συνήθως εμφανίζεται η περιγραφή (column de) ενός “news article” θα υπάρχει το πρώτο email στο table “users”. Με αυτό τον τρόπο ένας κακόβουλος χρήστης μπορεί να έχει πρόσβαση άμεσα σε ευαίσθητες πληροφορίες από τη βάση.

By SolidSNK solidsnk@grhack.net

Η πραγματικότητα

Facts:

(1) Πλέον τέτοια παραδείγματα σπανίζουν. Στα περισσότερα *Συστήματα Διαχείρισης Δεδομένων* (*Content Management System*) στον ιστό το input validation είναι εκτεταμένο με αποτέλεσματα το λογισμικό να είναι θωρακισμένο ενάντια σε SQL Injection στις περισσότερες των περιπτώσεων.

(2) Όμως, η αυξανόμενη πολυπλοκότητα των διάφορων CMS οδηγεί στο να εκτελούνται δεκάδες SQL queries ανα σελίδα, αυξάνοντας την πιθανότητα ο προγραμματιστής να έχει παραβλέψει το validation σε κάποια από αυτές. Επιπλέον, η ασφάλεια που προσφέρει μία γενική προστασία του λογισμικού καθώς και η φαινομενική αδυναμία εκμετάλλευσης ορισμένων περιπτώσεων (π.χ. σε ένα `INSERT` statement όπως θα δούμε) κάνει τον προγραμματιστή και κατ' επέκταση το λογισμικό πιο επιρρεπή σε input validation flaws ανοίγοντας το δρόμο σε πιο περίπλοκες SQL Injection.

By SolidSNK solidsnk@grhack.net

Η επίθεση

By SolidSNK solidsnk@grhack.net

Overview του λογισμικού-στόχου

- Είναι ένα Σύστημα Διαχείρησης Περιεχομένου του ιστού (Web Content Management System ή πιο σύντομα WCMS/CMS)
- Είναι γραμμένο σε php.
- Αλληλεπιδρά με βάση δεδομένων MySQL.
  - Η αλληλεπίδραση γίνεται μέσω της function

    `function db_query($qstring)`

    η οποία εκτός άλλων εκτελεί validation στο argument string. Αν αυτό κριθεί ασφαλές, καλεί τη `mysql_query()` με αυτούσιο το `$qstring` ως παράμετρο και κα επιστρέφει το αποτέλεσμα. Σε αντίθετη περίπτωση, για λόγους απλοποίησης, ας υποθέσουμε πως η php τερματίζεται καλώντας την `die()`.

(1) To Input validation flaw

By SolidSNK solidsnk@grhack.net

Εξετάζοντας το source code

```
foreach($_POST as $what => $value)
{
    ...
    if($what=='somevar' && $item>0)
       db_query("UPDATE data SET value=$value WHERE id=$member");
    ...
}
show_value();
```

Σημ: Η μεταβλητή `$member` παίρνει πάντα ακέραια τιμή. Η `show_value()` εν περιλήψη επιστρέφει με κάποιον τρόπο στη response την τωρινή τιμή της στήλης value του κάθε μέλους. Συνήθως αυτό γίνεται σε άλλο αρχείο, αλλά είναι εδώ για λόγους απλοποίησης. Αυτό που πρέπει κανείς να κρατήσει είναι πως το αποτέλεσμα της query είναι φανερό στο χρήστη.

Από τον κώδικα εξάγεται ότι:

- Η τιμή της POST μεταβλητής `somevar` χρησιμοποιείται αυτούσια στην παράμετρο της `db_query()` και ορίζει την τιμή της **ακέραιας** column '`value`'.

Είναι προφανές πως έχουμε ένα σφάλμα input validation, διότι η $value ,και μπορούμε να επηρεάσουμε τη query όπως θέλουμε. Υπάρχει όμως η περίπτωση να τη χρησιμοποιήσουμε για κακόβουλο σκοπό; Κοινώς, είναι άραγε exploitable ;!

By SolidSNK solidsnk@grhack.net

Exploitable ?

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
    SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

από mysql.com

Για ένα πίνακα `table_reference` , η τιμή που παίρνει η στήλη `co1_name1` είναι αποτέλεσμα της expression `expr1`. Expression στη mysql ορίζεται ως ένας συνδυασμός από functions, λογικές κ μαθηματικές εκφράσεις, literal values και ονομάτων στηλών, ο οποίος εν τέλει αντικαθιστάται (evaluates) με μία τιμή, την ισοδύναμη τιμή της expression.

Πρέπει να βρεθεί η κατάλληλη expression στο `UPDATE` statement ώστε η query στο post.php **να χρησιμοποιηθεί με σκοπό να προσπελάσουμε δεδομένα στη βάση**. Θυμηθείτε πως η expression αυτή πρέπει να κάνει evaluate αποκλείστικά σε μια ακέραια τιμή!

Η επιθυμητή expression απαιτεί τη χρήση:

(1) `IF` statement

(2) `SUBSTRING()`, `LENGTH()`, `ASCII()`, `CHAR()` functions

(3) *Subquery, ένα `SELECT` statement εν μέσω παρενθέσεων σε ένα εξωτερικό statement*

Όμως, η `db_query()` αποκλείει τις subqueries... ή μήπως όχι;

By SolidSNK solidsnk@grhack.net

(2) Παρακάμπτωντας την προστασία για τις subqueries

By SolidSNK solidsnk@grhack.net

Επεξήγηση της `db_query()`

Για τις ανάγκες της παρουσίασης θα χρησιμοποιηθεί η `validate()` η οποία αποτελείται μόνο από το validation τμήμα της `db_query()`

Η `db_query()` :

• Αναθέτει στη μεταβλητή `$teststr` το περιεχόμενο της `$qstring` σε lowercase και χωρίς whitespaces δεξιά και αριστερά.
• Ελέγχει το περιεχόμενο της `$teststr` για ύποπτες ακολουθίες χαρακτηρών όπως “union” , “sleep”, “benchmark”.
• Διαπιστώνει αν υπάρχουν ακολουθίες χαρακτηρών που αποτελούν σχόλια για τη MySQL.
• Ερευνεί την ύπαρξη subquery.

Κοιτώντας προσεκτικότερα τον κώδικα:
• Ο χαρακτήρας που σηματοδοτεί comment μέχρι το τέλος της γραμμής “#” παραλείφθηκε από τον έλεγχο.

Επεξήγηση της `db_query()`

Το τμήμα του κώδικα υπεύθυνο για τις subqueries είναι το παρακάτω

```
if (preg_match('~\([^)]*?select~s', $teststr) != 0)
```

Η regular expression, η παντέντα που επιχειρεί να περιγράψει επαρκώς την εμφάνιση subquery, είναι το πρώτο argument της `preg_match`. Συγκεκριμένα:

- Οι χαρακτήρες '`~`' στην αρχή της expression και πριν το '`s`' είναι οι delimiters που την οριοθετούν. Το '`s`' είναι ο modifier και σύμφωνα με αυτόν η τελεία (.) κάνει match και τους newline χαρακτήρες.
- Η ακολουθία χαρακτήρων '`\(`' κάνει match το χαρακτήρα '`(`'.
- Η επόμενη ακολουθία '`[^)]`' είναι μία negated character class και κάνει match ο,τιδήποτε δεν είναι κλειστή παρένθεση '`)`'.
- Ο ειδικός χαρακτήρας '`*`' κάνει match 0 ή παραπάνω φορές το προηγούμενο token (δηλαδή χαρακτήρα διαφορετικό από κλειστή παρένθεση). Το αγγλικό ερωτηματικό '`?`' περιορίζει την αναζήτηση στο πρώτο συμβατό match.
- Οι επόμενοι χαρακτήρες μέχρι την περισπωμένη έχουν κυριολεκτική σημασία, άρα αναζητείται αυτούσια η ακολουθία '`select`'.

By SolidSNK solidsnk@grhack.net

Παρακάμπτωντας τον έλεγχο

Σε απλά ελληνικά η regular epxression θα ικανοποιηθεί αν το argument string αποτελείται από μία ανοιχτή παρένθεση, μία οποιαδήποτε ακολουθία χαρακτήρων διάφορων της κλειστής παρένθεσης και τη λέξη “select”.

Βήμα-βήμα, σκοπός είναι:
(1) Το input να μην επαληθεύεσει την πατέντα.
(2) Η σύνταξη του να ακολουθεί τους κανόνες της MySQL για subqueries.

Πως θα γίνει αυτό;

By SolidSNK solidsnk@grhack.net

Παρακάμπτωντας τον έλεγχο

Μία λύση είναι η παρακάτω

```
"(#)\n
SELECT blah from bleh)"
```

Ή σε url encoded μορφή

```
"%28%23%29%0ASELECT+blah+from+bleh%29"
```

Η κλειστή παρένθεση διασφαλίζει πως η πατέντα δε θα βρει match, ενώ η newline κλείνει το σχόλιο που ξεκίνησε με το '#'. Η MySQL θα αγνοήσει το σχόλιο με αποτέλεσμα να υπάρχει μία έγκυρη subquery.

By SolidSNK solidsnk@grhack.net

(2+1) Η ολοκλήρωση της επίθεσης

By SolidSNK solidsnk@grhack.net

Η προσπέλαση των δεδομένων

Αφού λύθηκε το πρόβλημα με τις subqueries, μένει να προσδιοριστεί η τελική μορφή του input και κατ' επέκταση της query.

Αν υποθέσουμε πως μας ενδιαφέρει να προσπελάσουμε το πρώτο record της στήλης 'column' στον πίνακα 'table' , τότε σε ένα `UPDATE` statement με καθορισμένο το table και την **integer** column όπου γίνεται η αλλαγή, υπάρχει η εξής μέθοδος:

```
UPDATE tab SET col=(SELECT IF(SUBSTRING(SELECT column from table
LIMIT 1),1,1)='a',0,1)
```

Αν ο πρώτος χαρακτήρας του record είναι 'a' η εξωτερική subquery ισοδυναμεί με 0 , σε αντίθετη περίπτωση ισοδυναμεί με 1. Ο χρήστης κατόπιν εξετάζει την τιμή της col για να εξακριβώσει αν η condition είναι αληθής ή ψευδής. Συγκρίνοντας κάθε φορά το αποτέλεσμα του `SUBSTRING` με όλους τους πιθανούς χαρακτήρες και κάνοντας το ίδιο για κάθε γράμμα του record, αυξάνοντας κάθε φορά το offset στη `SUBSTRING()`, τότε μετά από ένα αρκετά μεγάλο αριθμό από queries το ζητούμενο record θα συλλεχθεί ολόκληρο.

Βέβαια, μια τέτοιου είδους σειριακή αναζήτηση δεν είναι καθόλου αποδοτική, ενώ στο πραγματικό παράδειγμα υπάρχουν ακόμα μερικές δυσκολίες και παράμετροι που πρέπει να ληφθούν υπόψιν.

By SolidSNK solidsnk@grhack.net

Η προσπέλαση των δεδομένων

Επιστρέφοντας στην πραγματικότητα

```
foreach($_POST as $what => $value)
{
    ...
    if($what=='somevar' && $item>0)
        db_query("UPDATE data SET value=$value WHERE id=$member");
    ...
}
show_data();
```

- Το τμήμα της query που επηρεάζεται είναι μετά το `SET`. Κανένα, πρόβλημα, απλά τα POST δεδομένα δε θα περιλαμβάνουν όλη τη query,
- Το CMS κάνει escape τα single και double quotes. Γι αυτό το λόγο χρησιμοποιείται η function `CHAR()` .
- Η σειριακή αναζήτηση , στη συγκεκριμένη περίπτωση η σύγκριση κάθε χαρακτήρα του record με όλους τους δυνατούς , είναι μη αποδοτική. Η απόδοση είναι ένας κρίσιμος παράγοντας και πρέπει να βρεθεί κάτι εναλλακτικό, ώστε να μειωθούν τα requests στο server στο μικρότερο δυνατό.
- Τα δεδομένα πρέπει να είναι url encoded.

By SolidSNK solidsnk@grhack.net

Η προσπέλαση των δεδομένων

Για το θέμα απόδοση, εναλλακτική της σειριακής αναζήτησης είναι η δυαδική, η οποία εφαρμόζεται σε ταξινομημένο πίνακα. Εδώ πρέπει να γίνει κατανοητό πως το σύνολο των δυνατών χαρακτηρών μπορεί να αναπαρασταθεί με τη βοήθεια των αντίστοιχων ASCII τιμών τους ως ένας ταξινομημένος πίνακας. Για παράδειγμα, αν το record είναι μια συμβολοσειρά δεκαεξαδικών χαρακτηρών, όπως ένας SHA-1 κρυπτογραφημένος κωδικός , τότε οι πιθανοί χαρακτήρες κάθε ψηφίου του κωδικού είναι οι αριθμοί 0-9 και τα γράμματα a-f. Από αυτό το σύνολο κατασκευάζεται ο πίνακας

“0123456789abcdef”
ή
{ 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 97 , 98 , 99 , 100 , 101 , 102 }

Έτσι για κάθε θέση της ζητούμενης συμβολοσειράς εφαρμόζεται δυαδική αναζήτηση στον παραπάνω πίνακα.

By SolidSNK solidsnk@grhack.net

Η προσπέλαση των δεδομένων

Συνοψίζοντας, το περιεχόμενο της POST μεταβλητής `wysiwyg` πρέπει να έχει την παρακάτω μορφή

```
“(#)\n
SELECT IF(ASCII(SUBSTRING((#)\n
SELECT password FROM users WHERE
admin=1),1,1))>55,0,1))”
```

Ή σε url-encoded μορφή

```
“%28%23%29%0ASELECT+IF%28ASCII%28SUBSTRING
%28%28%23%29SELECT+password+FROM+users+WHERE+admin=1%2
9,1,1%29%29>55,0,1%29%29”
```

Συγκεκριμένα, η `somevar` θα πάρει αυτή την τιμή στην πρώτη επανάληψη της binary search για τον πρώτο χαρακτήρα του κωδικού.

By SolidSNK solidsnk@grhack.net

Επίλογος

Αν και το παράδειγμα αναφερόταν σε συγκεκριμένο λογισμικό, υπάρχει η δυνατότητα γενίκευσης σε δύο επίπεδα, όσα ήταν και τα επιμέρους θέματα της παρουσίασης.

- Όσον αφορά το validation της `db_query()`, τέτοιο global validation είναι συχνό φαινόμενο στα μεγάλα web CMS. Μια αμέλεια του προγραμματιστή ή/και λίγο συνδιαστική σχέψη αρκούν πολλές φορές για την εξουδετέρωση του. Επίσης εκτέθηκε η αδυναμία της στρατηγικής Blacklist, καθώς ο κακόβουλος χαρακτήρας '#' ήταν άγνωστος στον προγραμματιστή και δεν αντιμετωπίστηκε όπως έπρεπε.

- Η μέθοδος exploitation ενός `UPDATE` statement είναι ακόμα άγνωστη στους developers. Αυτή τη στιγμή στον ιστό υπάρχουν πολλά λάθη τέτοιας μορφής, τα οποία είναι πλήρως εκμεταλλεύσιμα. Σημειωτέον πως η συγκεκριμένη μέθοδος μπορεί να χρησιμοποιηθεί με μικρές αλλαγές και σε `INSERT` statements.

By SolidSNK solidsnk@grhack.net