

Pokročilé metody XSS

XanthiX

Štruktúra prednášky

- 1) Teoretická časť – princíp XSS
- 2) Praktická časť – ukážka XSS keyloggeru prostredníctvom reklamného banneru

Motivácia: prečo XSS?

- Téma diplomovej práce
- Penetračný testing aplikačnej vrstvy
- XSS v súčasnosti značne vedie [cca 70-80% testovaných webových portálov zraniteľných voči XSS]
- XSS zraniteľnosti dávajú priestor kreativite útočníka => široké spektrum možností
- XSS zraniteľnosti sú úzko prepojené s ďalšími zraniteľnosťami aplikačnej vrstvy [HTTP RESPONSE SPLITTING => XSS => CSRF]

Čo robí XSS nebezpečným?

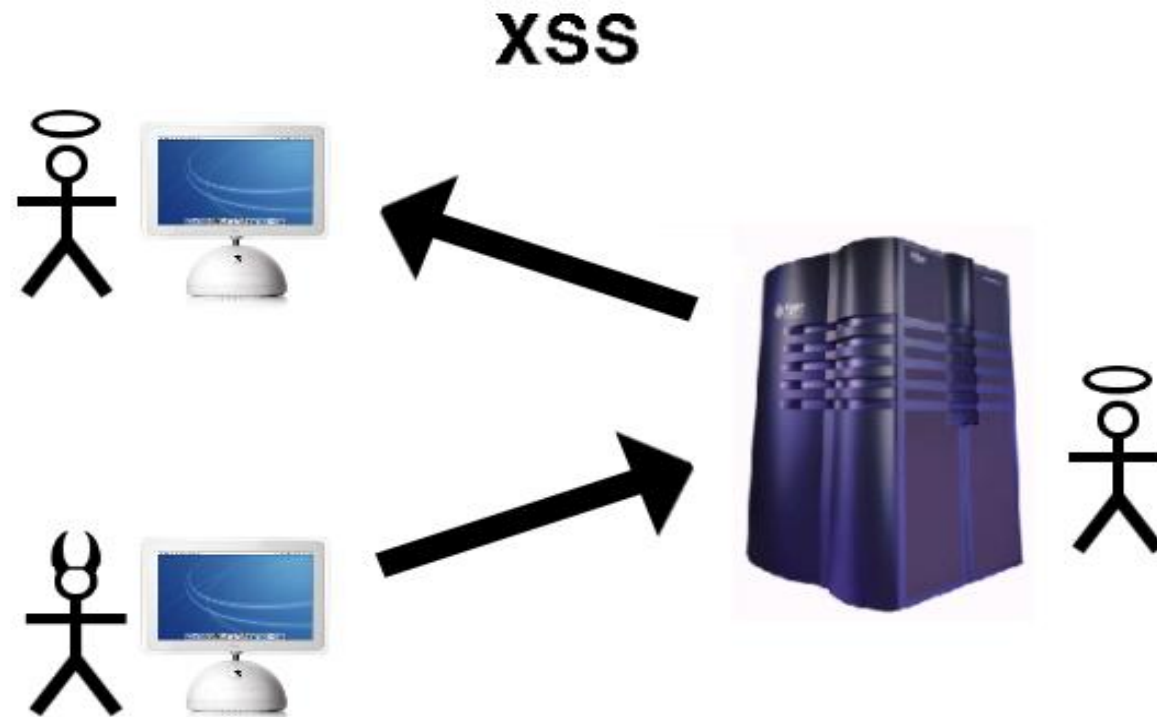
- Aktívne zneužitie XSS zraniteľnosti prechádza nepozorovane skrze spodné vrstvy ISO OSI modelu [L1-L6]
- Zraniteľnosť a jej aktívne zneužitie nie je vo všeobecnosti detekovateľné bežnými detekčnými mechanizmami [FW/IDS/IPS] [potreba použiť špeciálne FW detekujúce špecifické vzory typu XSS]
- Vytvorením šifrovaného spojenia prostredníctvom SSL/TLS nie je možné zamedziť útokom na báze XSS
- Zraniteľnosti typu XSS bývajú stále značne podceňované vývojármi webových aplikácií a neprístupuje sa k nim často s dostatočnou vážnosťou. [Nedostatočné bezpečnostné povedomie]. Na druhej strane čím ďalej, tým viac webových aplikácií využíva vývojové nástroje, ktoré dávajú priestor pre existenciu týchto zraniteľností [Web 2.0]
- Zraniteľnosť nie je často možné lokalizovať ku konkrétnemu miestu, ale je kombináciou viacerých vlastností, ktoré sa samo sebou nejavia ako bezpečnostná hrozba [príklad XSS prostredníctvom PDF, XSS prostredníctvom MOV]

Prostriedky útočníka

(multimediálne webové objekty a nástroje pre možnosť dynamickej interakcie vo webovom prostredí)

- JavaScript (at most)
- VBS Script
- Macromedia Flash Objects [using JS]
- ActiveX [in MS IE]
- PDF Files [using JS – partially fixed in Adobe Acrobat]
- GIF [already fixed in MS IE]
- XML
- CSS
- Video files [already fixed in Apple Quicktime]

Základný koncept XSS



XSS – Príklady možností útočníka

- Odcudzenie relácie autorizovaného užívateľa [napr. webmail]
- Odpočúvanie stlačených kláves v rámci kontextu napadnutej stránky
- Zneužitie dôveryhodnosti zraniteľného portálu a prezentácia nepravdivých informácií [iDnes.cz]
- Použitie prehliadača obeť ako proxy server útočníka [ovládanie botnetov]
- Infikovanie obeť škodlivým kódom prostredníctvom XSS [využitie dôveryhodnosti portálu]
- Skenovanie portov vnútorného segmentu siete
- Zmena konfigurácie WiFi access pointov
- Čítanie obsahu adresárovej štruktúry disku [skrze PDF XSS vektor]
- Vytvorenie skrytého komunikačného kanála medzi obeťou a útočníkom

Ako môže útočník preniesť svoj kód k obeti

- Útočník umiestni svoj kód na stránku, ktorú má pod vlastnou kontrolou (útočník = majiteľ záškodníckej stránky)
- Útočník vloží záškodný kód zneužitím bežných zraniteľností nižších vrstiev systému [SQL injection, buffer overflow, získanie rootovského hesla]
- Útočník je schopný vložiť (uložiť) a následne interpretovať svoj kód prostredníctvom dôveryhodnej tretej strany [fórum, reakcie v blogu, webmail, sociálne siete] (perzistentná XSS zraniteľnosť)
- Útočník využíva dôveryhodnú tretiu stranu len ako transportné médium, pričom jeho kód neostáva uložený na strane dôveryhodnej tretej strany (non-persistentná XSS zraniteľnosť)

Formy XSS zraniteľností

- Perzistentné XSS (Stored)
 - javascript je injektovaný permanentne do kódu webovej stránky
- Non-persistentné XSS (Reflected)
 - javascript je prenesený prostredníctvom XSS zraniteľnosti webu tretej strany, pričom záškodný kód neostáva uložený na tomto webe
- DOM-Based XSS (Reflected)
 - Veľmi podobné ako predchádzajúci prípad s tým, že útočníkov kód pri prenose vôbec nie je zasielaný na stranu zraniteľného serveru

Persistent XSS

- Most dangerous – global impact (practically anyone who visits the infected site can be potentially influenced)
- Attacker's javascript code is permanently inserted into the content of the webpage
- Attacker's code propagation:
 - Web-based e-mail
 - Web-based chat
 - Web forum
 - Blog
 - Barcode [Cross-Zeitung scripting]
- Example:
`check this url`

Non-persistent XSS

- the malicious (javascript) payload is echoed by the server in an immediate response to an HTTP request from the victim
- Poorly filtered parameter of the URL can be modified in such a way that the javascript is executed

Attacker's code propagation:

- E-mail
- ICQ
- IRC
- Blog
- Web forum

Example:

`http://www.vulnerable_website.com?x="><script>alert(1)</script>`

DOM-Based XSS (Cross-Site Scripting)

- It is not a result of a vulnerability within a server side script, but an improper handling of user supplied data in the client side JavaScript
- In general not possible to detect from the server
- Example:

```
<script>
  //client site vulnerable script
  var url = window.location.href;
  var pos = url.indexOf("title=") + 6;
  var len = url.length;
  var title_string = url.substring(pos,len);
  document.write(unescape(title_string));
</script>
```

Malicious URL:

```
http://victim/promo?product_id=100&title=Foo#<SCRIPT>alert('DOM-Based XSS')</SCRIPT>
```

Hiding the XSS (self-contained XSS)

- Original script:

```
<script>alert("Self-contained XSS");</script>
```

- Base64-encoded modification of the previous script

```
data:text/html;base64,PHNjcmlwdD5hbGVydCgiU2VsZi1jb250YWluZWQgWFNTIik7PC9zY3JpcHQ+
```

Hiding the XSS (URL encoded XSS)

- Original script:

```
javascript:alert("URL encoded XSS")
```

- URL-encoded modification of the previous script

```
javascript:%61%6C%65%72%74%28%22%55%52%4C%20%65%6E%63%6F%64%65%64%20%58%53%53%22%29
```

XSS Defense

(web developer's point of view)

Control the input as much as possible and do not allow for the attacker's JavaScript to be injected in any possible way. [e.g. using **htmlspecialchars** - PHP procedure which converts dangerous characters to HTML entities

Example:

< à <

> à >

XSS Defense

(potential victim's point of view)

- Effective defense without radical restriction of active components is hardly possible
- When a suspicious URL is received don't think just about the trustworthiness of the webserver but think about trustworthiness of the source as well.
- Restrict the javascript execution [e.g. NoScript extension for Firefox]
- Restrict the execution of the flash objects
- Be paranoid

References

- XSS Attacks – Syngress [book]
- Owasp.org
- Ha.ckers.org
- Gnucitizen.org
- [lecture given at 24C3 conference]
- http://chaosradio.ccc.de/24c3_m4v_2212.html
- [practical examples of XSS vulnerabilities]
- <http://xssed.org>
- <http://events.ccc.de/congress/2007/Hacks-XSS>