Често срещани уязвимости в web приложенията

представени от Георги Чорбаджийски
georgi@unixsol.org
http://georgi.unixsol.org/

Асоциация за Информационна Сигурност
http://iseca.org/

Сигурност?

- Сигурността не е нещо, което се купува
- Сигурността е процес
- Сигурността е компромис
- Сигурността трябва да е основна част от дизайна на приложението
- „Тестването може да открие присъствието на грешки, но не и отсъствието им“ - Dijkstra
- Security through obscurity – НЕ РАБОТИ!
- ПРОВЕРЯВАЙТЕ ВСИЧКО!
- Мислете като „лошите“
- Сигурността на една система е сигурността на най-слабото й място

Общи проблеми в web приложенията

- Топ 10 според Open Web Application Security Project
  1. Невалидирани входни данни
  2. Грешки при контрола на достъпа
  3. Грешки при управлението на сесиите и идентификацията
  4. Cross-Site Scripting (XSS)
  5. Препълване на буфери
  6. Злонамерено вмъкване на код (code injection)
  7. Проблеми при обработка на грешки
  8. Несигурно използване на криптография
  9. Denial of Services
  10. Грешки при конфигурация
- За повече информация
  - http://www.owasp.org/documentation/topten.html

Невалидирани входни данни

- Обработката на входни данни е важна част от всяко приложение
- Често допускани грешки:
  - Предполагане, че постъпилите данни са коректни
  - Проверяване за „грешни данни“ вместо за коректни
  - Не проверяване за валидност на формата на данните
  - Реализиране на ненадеждни проверки
- Частни случаи на проблема са уязвимостите от тип:
  - Cross-Site Scripting (XSS)
  - Препълване на буфери
  - Злонамерено вмъкване на код (code injection)

Невалидирани входни данни

- Как да разберем, че сме уязвими?
  - Ако не проверяваме входните данни или не го правим коректно със сигурност сме уязвими
- Как да се защитим?
  - Проверявайте за ВАЛИДНИ, а не за невалидни данни!
  - Не трябва да се разчита на проверки при клиента!
  - Получаваните данни трябва внимателно да бъдат проверявани за:
    - Тип на данните (string, integer, real...)
    - Разрешена кодова таблица
    - Минимална и максимална дължина, позволен интервал, дефиниционно множество
    - Разрешена ли е null стойност
    - Задължителен ли е даден параметър
    - Позволени ли са дублиращи се данни

Грешки при контрола за достъп

- Контролът за достъп е нещото, което дава различни права на потребителите на едно уеб приложение
- Често допускани грешки при реализацията
  - Подценяване на сложността на задачата
  - „Разхвърляне“ на контрола на много места
  - Пропускане на контрола за достъп в дизайна на приложението
- Как да разберем, че сме уязвими?
  - Нямаме яснота как точно работи контролът за достъп
  - Реализацията не е ясно написана и документирана
  - В идеята на приложението не е залегнал контролът за достъп

Грешки при контрола за достъп

- Как да се защитим?
  - Ясна дефиниция на контрола за достъп
  - Централизация
  - Специфични проблеми, за които трябва да се внимава:
    - Несигурно ползване на идентификатори
    - Forced browsing past access controls
    - Path traversal
    - File Permissions
    - Кеширане от страна на клиента

Сесии и идентификация

- Тъй като HTTP протоколът не поддържа сесии, разработчиците се налага да реализират собствени механизми и това води до грешки
- Често допускани грешки
  - Използване на несигурни методи за идентификация
  - „Lost password“ изисква твърде малко данни
  - Пазене на повече от нужното при клиента
  - Използване на предвидими сесийни идентификатори
  - Съхраняване на нехеширани пароли
- Как да разберем, че сме уязвими?
  - Като проследим всички процеси, при които се използва идентификация и сесия и проверим дали са коректни

Сесии и идентификация

- Как да се защитим?
  - Налагане ползването на силни пароли
  - Механизъм за защита от brute force атаки на паролата
  - Сигурни механизми при смяната на парола и свързаната с нея информация
  - Съхранение на паролите
  - Защита на предаваните данни чрез криптиране
  - Обвързване на сесийният идентификатор с допълнителни данни на потребителя
  - Избягвайте да давате информация за други потребители
  - Не ползвайте GET при идентификация и предаване на сесийна информация
  - Всеки компонент на даден сайт трябва да проверява идентификацията, а не да вярва на другите компоненти

Cross site scripting (XSS)

- Какъв е проблемът?
  - Атака върху потребителя
  - Експлоатира се доверието на потребител към даден сайт
  - Използва се за крадене информация от потребителска сесия или самата сесия
  - Реализира се чрез вмъкване на HTML тагове и JavaScript
- Как да разберем, че сме уязвими?
  - Ако показваме данни, идващи от потребителите без подходящо филтриране, сме уязвими
- Как да се защитим?
  - Не показвайте директно данни, идващи от потребителите на други потребители
  - Филтрирайте и/или нормализирайте показваните данни
- За повече информация
  - http://www.cgisecurity.com/articles/xss-faq.html

Препълване на буфери

- Това са проблеми в компилируемите езици, за щастие такива не се ползват често при уеб приложенията
- Как да разберем, че сме уязвими?
  - Трябва да следим новините, свързани със сигурността на софтуера, който ползваме
  - Трябва да знаем отговора на въпросите:
    - Ползваме ли несигурни или трудни за работа функции за обработка на низове и заделяне на памет?
    - Предвиждаме ли достатъчно големи буфери за входните данни?
- Как да се защитим?
  - Използвайте функции, които по надежден начин проверяват големината на буферите и входа (`strlcpy`, `snprintf`)
  - Избягвайте функции без начин за подаване на големина на буфера (`gets`, `strcpy`)

Злонамерено вмъкване на код Command Injection

- Какъв е проблемът?
  - Възможно е изпълнение на нежелани команди на сървъра
- Примерна програма

```
<?php
  // $_SESSION['userid'] = ';cat /etc/passwd';
  system("ls /home/{$_SESSION['userid']}");
?>
```

- Как да разберем, че сме уязвими?
  - Всяко извикване на външна команда с нефилтрирани параметри подадени от потребителя е потенциална уязвимост
- Как да се защитим?
  - При извикване на външни команди не ползвайте параметри, контролирани от потребителя
  - Филтрирайте командите и параметрите

Злонамерено вмъкване на код SQL Injection

- Какъв е проблемът?
  - Възможно е пълно компрометиране на базата данни
  - При някои бази данни може да доведе до изпълнение на команди на сървъра
  - При някой бази може да доведе до четене и писане на файлове, за които DB сървъра има права
- Примерна програма:

```
<?php
  $user == "' OR user = 'admin'";
  $query = "UPDATE usertable SET pwd='xxxx' WHERE user='$user'";
# $query = "UPDATE usertable SET pwd='xxxx' WHERE user='' OR user='admin'";
?>
```

Злонамерено вмъкване на код SQL Injection

- Как да разберем, че сме уязвими?
  - Всяко изграждане на заявка с нефилтрирани параметри, подадени от потребителя е потенциална уязвимост
- Как да се защитим?
  - Никога не ползвайте в SQL заявка данни, които не сте "почистили"
  - Prepared queries
  - Ограничавайте правата на потребителя, които ползвате за достъп до базата
- За повече информация
  - http://www.unixwiz.net/techtips/sql-injection.html

Проблеми при обработка на грешки

- Какъв е проблемът?
  - При обработка на грешки на потребителя се дава твърде много информация
  - При грешка по подразбиране действието се разрешава
  - Не се запазва информацията за появили се грешки
- Как да разберем, че сме уязвими?
  - Подложете приложението си на crash testing
  - Подавайте комбинации от валидни и невалидни данни
- Как да се защитим?
  - Тествайте всички възможни пътища за появяване на грешки
  - Създайте си правила, коя част от информацията за възникнала грешка се подава към потребителя и коя се запазва

Несигурно използване на криптография

- Повечето web приложения трябва да пазят някои важни данни в БД или във файлове и въпреки че има добри библиотеки с функции за криптиране се допускат основни грешки при реализацията
- Често допускани грешки
  - Важни данни не се криптират
  - Несигурно съхранение на ключове, сертификати и пароли
  - Неправилно съхранение на данни в паметта
  - Лоши източници на случайни числа
  - Лош избор на алгоритъм за криптиране/хеширане
  - Опити за създаване на собствен алгоритъм за криптиране/хеширане

Несигурно използване на криптография

- Как да разберем, че сме уязвими?
  - Хората пишещи код, занимаващ се с криптиране трябва много добре да разбират какво правят
  - При разглеждане на кода трябва да се обърне специално внимание на ползваните алгоритми, тяхната реализация, как защитените данни се записват, четат и обработват
- Как да се защитим?
  - Използвайте готови и тествани библиотеки
  - Подсигурете сигурното съхранение на паролите, ключовете и сертификатите

Denial of Services

- Web приложенията са особено уязвими на DoS атаки поради това, че не е лесно да отличат нормалният трафик от атаката
- Често допускани грешки
  - В конфигурацията на сървъра няма ограничения върху броя на обслужваните клиенти
  - Не се прави разграничаване на заявките по адреси
  - Позволява се „изключването“ на потребители чрез атака с грешни пароли

Denial of Services

- Как да разберем, че сме уязвими?
  - Използвайте програма за тестване на натоварването
  - Тествайте от няколко адреса едновременно
- Как да се защитим?
  - Основно правило е да имате лимит на заявките
  - Оптимизирайте приложенията си
  - Кеширайте резултатите от заявките
  - За идентифицирани потребители
    - Обслужвайте само X брой заявки от идентифициран потребител
    - При постъпване на няколко заявки, обработете ги последователно, използвайки сесийната информация за синхронизация
  - За не идентифицирани потребители
    - Избягвайте използването на функционалност, изискваща много ресурси
    - Обслужвайте само определен брой паралелни заявки

Грешки при конфигурацията

- Конфигурацията на сървъра и на web приложението играят основна роля за сигурността
- Често допускани грешки
  - Незакърпени проблеми в сигурността на сървърите
  - Грешки в конфигурацията, позволяващи да се виждат директории и достъп до тях
  - Забравени достъпни примерни файлове, архивни копия, конфигурации и т.н.
  - Неправилно настроени права за достъп до файлове и директории
  - Разрешени ненужни услуги, включително такива за отдалечена администрация
  - Използване на стандартни акаунти с непроменени пароли
  - Твърде информативни съобщения за грешки
  - Неправилно конфигурирани SSL сертификати и криптографски настройки
  - Използване на самоподписани сертификати

Грешки при конфигурацията

- Как да разберем, че сме уязвими?
  - Използвайте скенери за уязвимости
  - При избиране на даден сървърен софтуер се постарайте да научите какъв е препоръчителният начин сървъра да работи сигурно
  - Избягвайте ползването на сървъри с лош track record
- Как да се защитим?
  - Следете за последните проблеми в сигурността
  - Инсталирайте винаги най-новите версии на софтуера и слагайте необходимите кръпки към него
  - Винаги изключвайте услугите, които не ползвате
  - Редовно проверявайте за проблеми в конфигурацията, следвайки съветите за сигурно конфигуриране

Заръки за програмиста, пишещ сигурен софтуер

- Проверявай винаги и всичко!
- Прави проверки, така че да не могат да бъдат избегнати
- Не вярвай на данни, идващи от външен източник (потребители, db, файлове и т.н.)
- Внимавай когато даваш достъп до данни, предоставени от потребителя
- Използвай правилно криптография
- Не слагай на публично достъпни места конфиденциални данни

Заръки за администратора

- На работещ сайт записвай грешките, но не ги показвай на гледащия страницата
- Всеки сайт трябва да работи с различен потребител
- Всеки потребител трябва да вижда само собствените си файлове
- Забрани функционалността, която е лесно да бъде използвана неправилно
- Следи логовете си!

Благодаря за вниманието!

Имате ли въпроси?