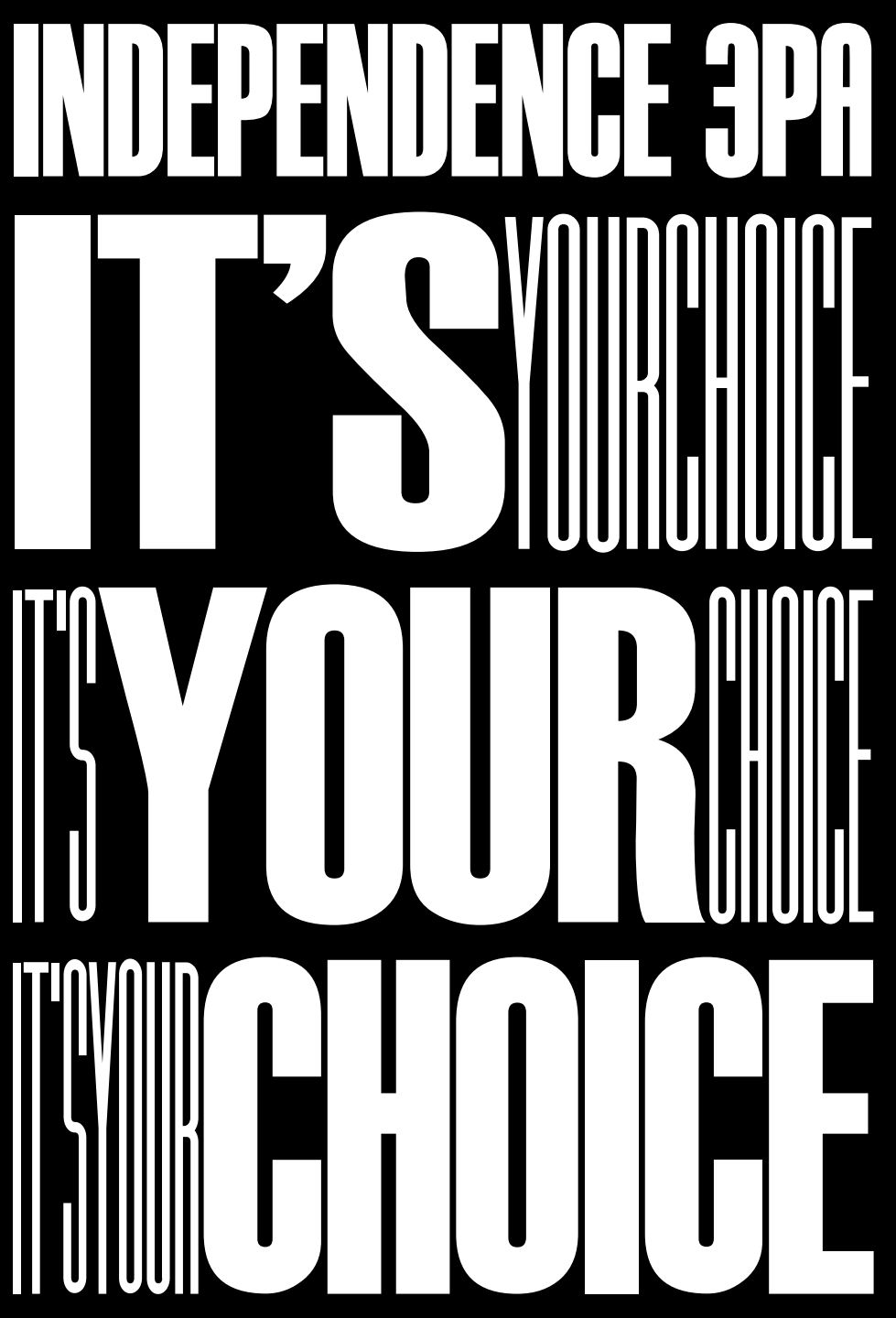
Positive Hack Days

Анализ клиентского JavaScript-кода для обнаружения HTTP-эндпоинтов

Даниил Сигалов

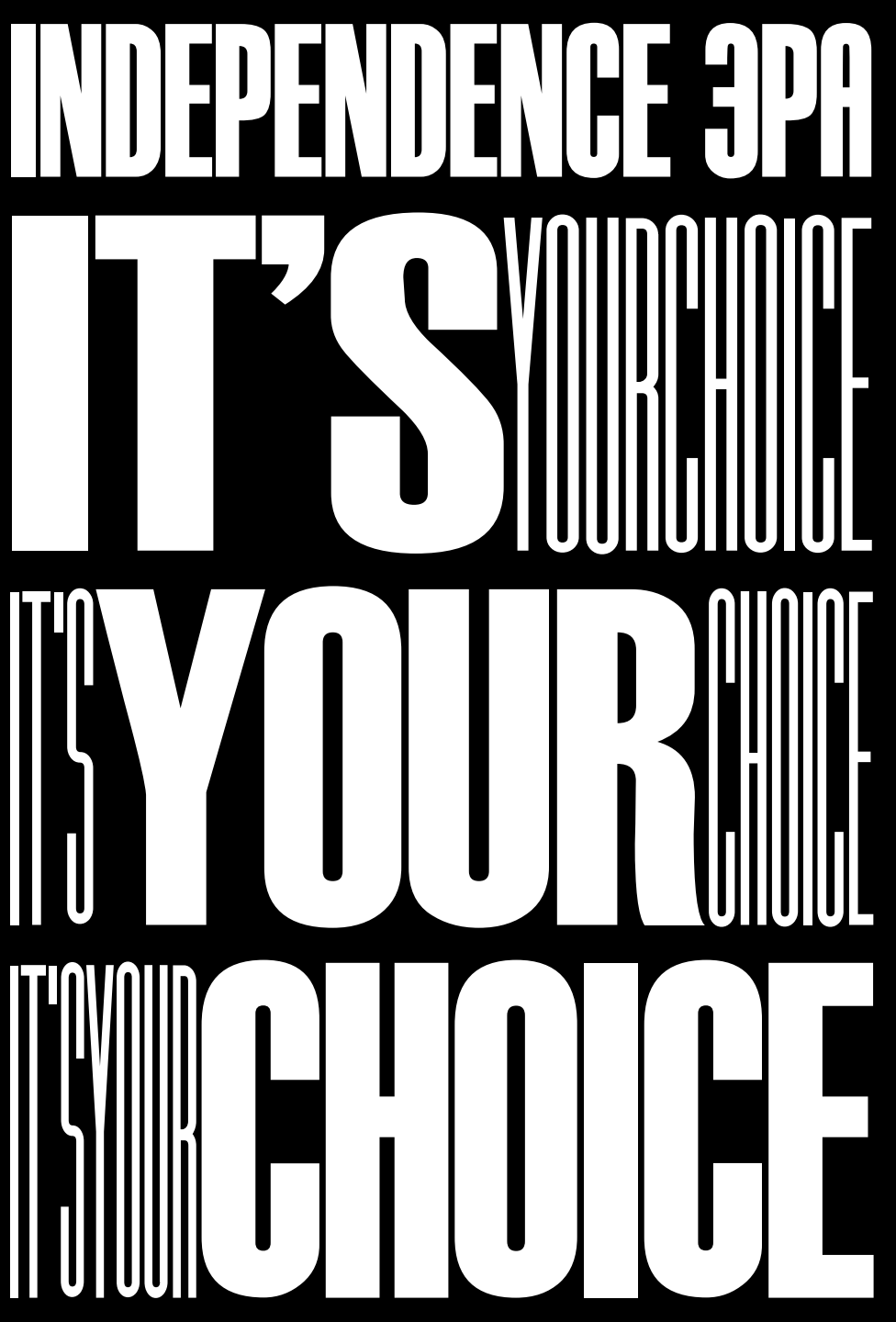
Даниил Сигалов

- Работаю в компании SolidSoft
- Научный сотрудник факультета ВМК МГУ
- Играю за CTF-команду Bushwhackers
- github & tg: @asterite3
- email: asterite@seclab.cs.msu.ru



Анализатор кода для обнаружения HTTP-эндпоинтов

- Компонент black-box web-сканера, который делаем в SolidSoft
- Научная работа
- Делается группой ресёрчеров



Задача: автоматический поиск HTTP-эндпоинтов

При поиске серверных уязвимостей отправляем атакующие вектора

```
"><script>alert(1);</script>
```

```
' or 1=1 --
```

Задача: автоматический поиск HTTP-эндпоинтов

При поиске серверных уязвимостей отправляем атакующие вектора

⟼ `"><script>alert(1);</script>`

⟼ `' or 1=1 --`

Но куда именно их подставлять? В какие запросы?

```
POST /api/user/info
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 14

id=' or 1=1 --
```

Задача: автоматический поиск HTTP-эндпоинтов

Выявление поверхности атаки

⟶ Компонент сканера безопасности

⟶ Также может быть полезно при пентесте вручную

Задача: автоматический поиск HTTP-эндпоинтов

Выявление поверхности атаки

Источник: **веб-страницы**

(есть и другие: дирбастинг, известные эндпоинты конкретного софта, …)

Майнинг HTTP-эндпоинтов с веб-страницы

```
<script type="text/javascript">
        function saveLike(articleID) {
            $.ajax({
                method: 'POST'
                url: '/api/like/add',
                data: {
                    id: articleID
                }
            });
        }
        function getNewsList(cb) {
            $.getJSON('/api/get-data?type=news_list', cb);
        }
  </script>
 </head>
<body>
  <a href="/user/profile?id=18">Article author</a>
  <a href="/article/text?id=84932">Read full text</a>
  <form method="POST" action="/news/search">
    <input type="text" name="query">
    <input type="text" name="sort_by">
    <input type="submit" name="Искать">
  </form>
```

Майнинг HTTP-эндпоинтов с веб-страницы

```
<script type="text/javascript">
        function saveLike(articleID) {
            $.ajax({
                method: 'POST'
                url: '/api/like/add',
                data: {
                    id: articleID
                }
            });
        }
        function getNewsList(cb) {
            $.getJSON('/api/get-data?type=news_list', cb);
        }
  </script>
 </head>
<body>
  <a href="/user/profile?id=18">Article author</a>
  <a href='/article/text?id=84932">Read full text</a>
  <form method="POST" action="/news/search">
    <input type="text" name="query">
    <input type="text" name="sort_by">
    <input type="submit" name="Искать">
  </form>
```

Просто для запросов,
отправляемых HTML-разметкой

Майнинг HTTP-эндпоинтов с веб-страницы

```
<script type="text/javascript">
        function saveLike(articleID) {
            $.ajax({
                method: 'POST'
                url: '/api/like/add',
                data: {
                    id: articleID
                }
            });
        }
        function getNewsList(cb) {
            $.getJSON('/api/get-data?type=news_list', cb);
        }
  </script>
 </head>
<body>
  <a href="/user/profile?id=18">Article author</a>
  <a href='/article/text?id=84932">Read full text</a>
  <form method="POST" action="/news/search">
    <input type="text" name="query">
    <input type="text" name="sort_by">
    <input type="submit" name="Искать">
  </form>
```

Майнинг HTTP-эндпоинтов с веб-страницы

⟶ Просто для запросов, отправляемых HTML-разметкой

⟶ Для запросов, отправляемых JS-кодом, задача намного сложнее
- требует анализа кода

Какие запросы может отправлять JS-код на странице?

“Эндпоинты” vs запросы

```
GET /api/user/info?id=151 HTTP/1.1
Host: example.com
X-Requested-With: XMLHttpRequest
```

```
GET /api/user/info?id=152 HTTP/1.1
Host: example.com
X-Requested-With: XMLHttpRequest
```

Это разные запросы, но они относятся к одной серверной “ручке”

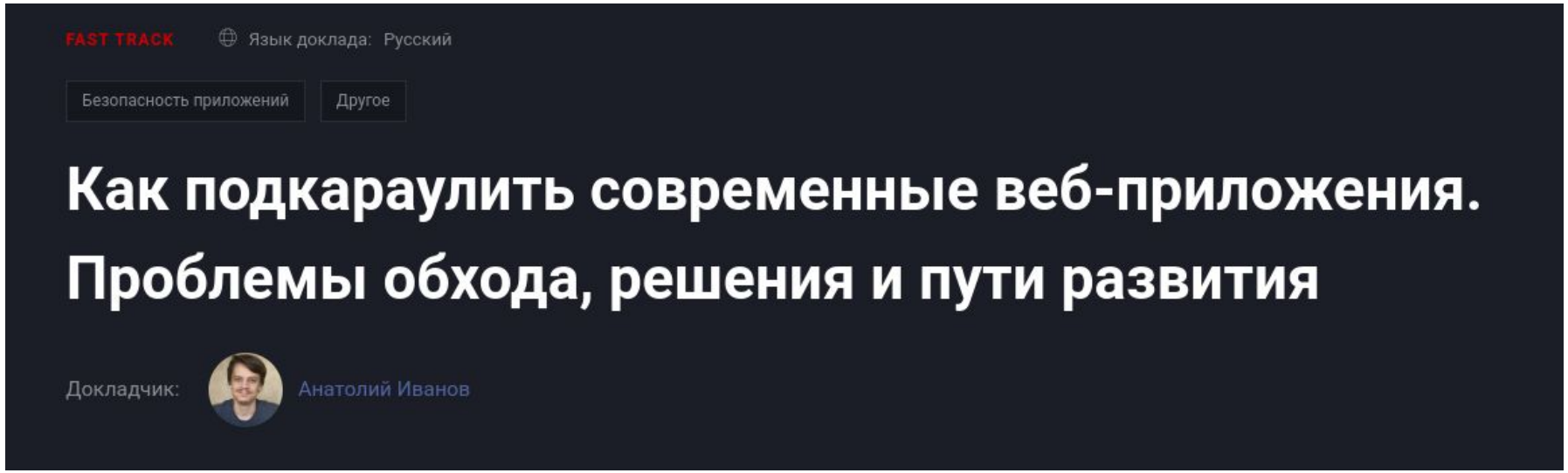
Какие запросы может отправлять JS-код?

В современных сканерах безопасности для этого чаще всего используют **динамический краулинг**

Динамический краулинг

Страницы открываются в headless браузере, иногда после этого с их элементами происходит какое-то взаимодействие

Смотрим какие запросы отправились со страницы, парсим её разметку после загрузки



Динамический краулинг

+ простой динамический краулинг дёшево реализовать
+ часть запросов легко найти таким способом

Динамический краулинг

+ простой динамический краулинг дёшево реализовать
+ часть запросов легко найти таким способом

- краулинг с полным покрытием непонятно как сделать, это активная область исследований
- “полноценный” динамический краулинг долгий
- он не находит запросы в JS, которые нельзя стриггерить

Динамический краулинг

+ простой динамический краулинг дёшево реализовать
+ часть запросов легко найти таким способом

- краулинг с полным покрытием непонятно как сделать, это активная область исследований
- “полноценный” динамический краулинг долгий
- он не находит запросы в JS, которые нельзя стриггерить

В сканерах, которые мы видели, динамический краулинг реализован в простейшем виде

```
var baseURL = "https://site.com/data";
function someFunctionWhoKnowsHowItIsCalled() {
    $.ajax({
        url: baseURL + "/search",
        data: {
            q: "abcd",
            limit: "100"
        }
    });
}
```

```
var baseURL = "https://site.com/data";
function someFunctionWhoKnowsHowItIsCalled() {
    $.ajax({
        url: baseURL + "/search",
        data: {
            q: "abcd",
            limit: "100"
        }
    });
}
```

Идея

```
var baseURL = "https://site.com/data";
function someFunctionWhoKnowsHowItIsCalled() {
    $.ajax({
        url: baseURL + "/search",
        data: {
            q: "abcd",
            limit: "100"
        }
    });
}
```

Идея

- иногда стриггерить запрос сложно или невозможно, при этом в JS запрос хорошо виден
- часто много JS кода бандлится вместе, включая неиспользуемый

Идея

- иногда стриггерить запрос сложно или невозможно, при этом в JS запрос хорошо виден
- часто много JS кода бандлится вместе, включая неиспользуемый
- мёртвый код неинтересен для клиентских уязвимостей, но интересен для поиска эндпоинтов

Существующие анализаторы JS

- ни один не решает нашу задачу из коробки
- state-of-the-art статические анализаторы JS-кода (TAJS, SAFE, WALA) на простейшей странице с jQuery будут работать очень долго, после чего завершаться с ошибкой
- часто полу-конкретные данные представлены слишком абстрактно

```
"/api/" + any_string + "/do.php" = any_string
```

- не анализируют мёртвый код

Статический анализ JS для майнинга HTTP-запросов

- ⟶ поэтому мы решили написать свой собственный анализатор
- ⟶ анализ должен быть достаточно лёгким, чтобы работать на реальных страницах

Статический анализ JS для майнинга HTTP-запросов

- наш алгоритм наивный, довольно простой, не sound
- не использует решёток и fixpoint солвер
- абстрактных значений всего два: *UNKNOWN* и *FROM_ARG*

Алгоритм

Algorithm 3 Algorithm of searching for AJAX calls and their arguments

```
1:  function ExtractDEPs(AST, MemModel, Chain, Queue)
2:      d ← 0
3:      for all vertex v ∈ AST in DFS order do
4:          if v is CallExpression then
5:              if ∃sign ∈ SignatureSet : MatchesSignature(v, sign) == true then
6:                  args ← v.arguments
7:                  vals ← [EvalExpr(arg, MemModel, ScopeModel) for arg in args]
8:                  vals, hadArgsDependency ← CheckAndRemoveFromArg(vals)
9:                  Results ← Results ∪ {(sign, vals)}
10:                 if hadArgsDependency == true then
11:                     Queue ← BuildCallChains(v, AST, Queue, ScopeModel, MemModel)
12:             if Chain is not ε and |Chain| > 0 v.callee is Identifier then
13:                 (binding, ast, args), rest ← cut first element from Chain
14:                 if binding == GetBinding(v.callee, ScopeModel) then
15:                     act, MemModel ← SetActualArgVals(func, args, MemModel)
16:                     ret ← ExtractDEPs(ast, MemModel, rest, Queue)
17:                     newResults, newQueue ← ret
18:                     Results ← Results ∪ newResults
19:                     Queue ← Queue concatenated with newQueue
20:                     MemModel ← MemModel \ act
21:         else if v is Function then
22:             d ← d + 1
23:             args ← v'.arguments
24:             formal ← {(GetBinding(a, ScopeModel) ↦ FromArg) for a in args}
25:             MemModel ← MemModel ∪ formal
26:         else if d > 0 and v.type ∈ {VariableDeclarator, AssignmentExpression} then
27:             MemModel ← Assign(v, MemModel)
28:         if ∃ exited vertex v' and v' is Function then
29:             d ← d − 1
30:             args ← v'.arguments
31:             formal ← {(GetBinding(a, ScopeModel) ↦ FromArg) for a in args}
32:             MemModel ← MemModel \ formal
33:     return Results, Queue
34: function SetActualArgVals(func, args, MemModel)
35:     vals ← [EvalExpr(arg, MemModel, ScopeModel) for arg in args]
36:     act ← {(GetBinding(a_i, ScopeModel) ↦ vals_i) for 0 <= i < |func.arguments|}
37:     MemModel ← MemModel ∪ act
38:     return act, MemModel
```

Алгоритм

Algorithm 3 Algorithm of searching for AJAX calls and their arguments

```
1:  function ExtractDEPs(AST, MemModel, Chain, Queue)
2:      d ← 0
3:      for all vertex v ∈ AST in DFS order do
4:          if v is CallExpression then
5:              if ∃sign ∈ SignatureSet : MatchesSignature(v, sign) == true then
6:                  args ← v.arguments
7:                  vals ← [EvalExpr(arg, MemModel, ScopeModel) for arg in args]
8:                  vals, hadArgsDependency ← CheckAndRemoveFromArg(vals)
9:                  Results ← Results ∪ {(sign, vals)}
10:                 if hadArgsDependency == true then
11:                     Queue ← BuildCallChains(v, AST, Queue, ScopeModel, MemModel)
12:             if Chain is not ∅ and |Chain| > 0 v.callee is Identifier then
13:                 (binding, ast, args), rest ← cut first element from Chain
14:                 if binding == GetBinding(v.callee, ScopeModel) then
15:                     act, MemModel ← SetActualArgVals(func, args, MemModel)
16:                     ret ← ExtractDEPs(ast, MemModel, rest, Queue)
17:                     newResults, newQueue ← ret
18:                     Results ← Results ∪ newResults
19:                     Queue ← Queue concatenated with newQueue
20:                     MemModel ← MemModel \ act
21:         else if v is Function then
22:             d ← d + 1
23:             args ← v'.arguments
24:             formal ← {(GetBinding(a, ScopeModel) ↦ FromArg) for a in args}
25:             MemModel ← MemModel ∪ formal
26:         else if d > 0 and v.type ∈ {VariableDeclarator, AssignmentExpression} then
27:             MemModel ← Assign(v, MemModel)
28:         if ∃ existed vertex v' and v' is Function then
29:             d ← d − 1
30:             args ← v'.arguments
31:             formal ← {(GetBinding(a, ScopeModel) ↦ FromArg) for a in args}
32:             MemModel ← MemModel \ formal
33:     return Results, Queue
34: function SetActualArgVals(func, args, MemModel)
35:     vals ← [EvalExpr(arg, MemModel, ScopeModel) for arg in args]
36:     act ← {(GetBinding(a_i, ScopeModel) ↦ vals_i) for 0 <= i < |func.arguments|}
37:     MemModel ← MemModel ∪ act
38:     return act, MemModel
```

Алгоритм

Анализатор ищет в JS-коде вызовы, отправляющие запрос на сервер, и их аргументы

- встроенные: `fetch`, `XMLHttpRequest`
- вызовы популярных библиотек (jQuery, Axios, AngularJS, …)
- вызовы находятся просто по синтаксическим сигнатурам, включающим имена функций и объектов
  - `$.ajax(...)`
  - `$http.post(...)`
  - `axios.put(...)`
  - `fetch(...)`

Алгоритм

- анализ работает над AST-деревом (CFG не строится)
- AST-дерево обходится в глубину
  - присваивания переменных, встреченные при обходе, **запоминаются**

Алгоритм

- анализ работает над AST-деревом (CFG не строится)
- AST-дерево обходится в глубину
  - присваивания переменных, встреченные при обходе, **запоминаются**
  - для значений, которые вычислить не можем, получается *UNKNOWN*

Алгоритм

- анализ работает над AST-деревом (CFG не строится)
- AST-дерево обходится в глубину
  - присваивания переменных, встреченные при обходе, **запоминаются**
  - для значений, которые вычислить не можем, получается *UNKNOWN*
  - встречая AJAX-вызов, анализатор вычисляет его аргументы, используя записанные значения переменных

Моделирование данных

- значения: конкретные (`5, "abc", null, ["x", 1]`) и два абстрактных (*`UNKNOWN`* и *`FROM_ARG`*)
- в объектах и массивах могут быть абстрактные значения (`{x: [`*`UNKNOWN`*`]}`)
- для каждой переменной анализ хранит только **одно** значение
- переменные идентифицируются с точностью до лексического скоупинга

Моделирование данных

Переменные идентифицируются с точностью до лексического скоупинга

```
var x = 5;
function f(y) {
    var x = "abc";
    if (y > 2) {
        let x = null;
    }
}
```

Для этого примера анализатор различает разные переменные “x”

Современные парсеры предоставляют эту информацию (например, Babel)

Алгоритм

- поддерживается ряд встроенных операций JS (+, присваивание полей объектов, …)
- вызовы пользовательских функций игнорируются, их возвращаемым значением считается *UNKNOWN*
- для поиска значений глобальных переменных делается ещё один предварительный обход AST
- условия ветвлений игнорируются, все ветви всегда обходятся один раз
- фактически, анализ состоит из нескольких обходов AST

Алгоритм - пример

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2
3  function fetchData(recordID, cache, resultCB) {
4      if (cache.haveRecord(recordID)) {
5          resultCB(cache.get(recordID));
6      } else {
7          $.ajax({
8              url: apiBaseURL + '/data/fetch',
9              data: {
10                 record: recordID,
11                 limit: 100,
12                 format: "json"
13             },
14             success: function(/*...*/){/*...*/}
15         });
16     }
17 }
```

Алгоритм - пример

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2
3  function fetchData(recordID, cache, resultCB) {
4      if (cache.haveRecord(recordID)) {
5          resultCB(cache.get(recordID));
6      } else {
7          $.ajax({
8              url: apiBaseURL + '/data/fetch',
9              data: {
10                 record: recordID,
11                 limit: 100,
12                 format: "json"
13             },
14             success: function(/*...*/){/*...*/}
15         });
16     }
17 }
```

```
location.origin = "http://tst.com"
apiBaseURL = "http://tst.com/api/v2.0/"
```

Алгоритм - пример

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2
3  function fetchData(recordID, cache, resultCB) {
4      if (cache.haveRecord(recordID)) {
5          resultCB(cache.get(recordID));
6      } else {
7          $.ajax({
8              url: apiBaseURL + '/data/fetch',
9              data: {
10                 record: recordID,
11                 limit: 100,
12                 format: "json"
13             },
14             success: function(/*...*/){/*...*/}
15         });
16     }
17 }
```

```
location.origin = "http://tst.com"
apiBaseURL = "http://tst.com/api/v2.0/"
recordID = FROM_ARG
cache = FROM_ARG
resultCB = FROM_ARG
```

Алгоритм - пример

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2
3  function fetchData(recordID, cache, resultCB) {
4      if (cache.haveRecord(recordID)) {
5          resultCB(cache.get(recordID));
6      } else {
7          $.ajax({
8              url: apiBaseURL + '/data/fetch',
9              data: {
10                 record: recordID,
11                 limit: 100,
12                 format: "json"
13             },
14             success: function(/*...*/){/*...*/}
15         });
16     }
17 }
```

```
location.origin = "http://tst.com"
apiBaseURL = "http://tst.com/api/v2.0/"
recordID = FROM_ARG
cache = FROM_ARG
resultCB = FROM_ARG
```

Алгоритм - пример

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2
3  function fetchData(recordID, cache, resultCB) {
4      if (cache.haveRecord(recordID)) {
5          resultCB(cache.get(recordID));
6      } else {
7          $.ajax({
8              url: apiBaseURL + '/data/fetch',
9              data: {
10                 record: recordID,
11                 limit: 100,
12                 format: "json"
13             },
14             success: function(/*...*/){/*...*/}
15         });
16     }
17 }
```

```
location.origin = "http://tst.com"
apiBaseURL = "http://tst.com/api/v2.0/"
recordID = FROM_ARG
cache = FROM_ARG
resultCB = FROM_ARG
```

Алгоритм - пример

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2
3  function fetchData(recordID, cache, resultCB) {
4      if (cache.haveRecord(recordID)) {
5          resultCB(cache.get(recordID));
6      } else {
7          $.ajax({
8              url: apiBaseURL + '/data/fetch',
9              data: {
10                 record: recordID,
11                 limit: 100,
12                 format: "json"
13             },
14             success: function(/*...*/){/*...*/}
15         });
16     }
17 }
```

```
apiBaseURL + '/data/fetch' =
   "http://tst.com/api/v2.0/data/fetch"
```

Алгоритм - пример

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2
3  function fetchData(recordID, cache, resultCB) {
4      if (cache.haveRecord(recordID)) {
5          resultCB(cache.get(recordID));
6      } else {
7          $.ajax({
8              url: apiBaseURL + '/data/fetch',
9              data: {
10                 record: recordID,
11                 limit: 100,
12                 format: "json"
13             },
14             success: function(
15         });
16     }
17 }
```

```
$.ajax call
args[0] = {
    url: "http://tst.com/api/v2.0/data/fetch",
    data: {
        record: FROM_ARG,
        limit: 100,
        format: "json"
    }, …
}
```

Алгоритм - пример

```
$.ajax call
args[0] = {
    url: "http://tst.com/api/v2.0/data/fetch",
    data: {
        record: FROM_ARG,
        limit: 100,
        format: "json"
    }, …
}
```

```
GET /api/v2.0/data/fetch?record=UNKNOWN&limit=100&format=json
HTTP/1.1
Host: tst.com
X-Requested-With: XMLHttpRequest
```

Алгоритм

Межпроцедурный анализ - поиск значений аргументов вызывающей функции

- для формальных аргументов функции используется специальное значение: `FROM_ARG`
- если при обработке AJAX-вызова в функции f мы встречаем в аргументах `FROM_ARG`, то ищем места вызова f
  - ищем по значению среди переменных. Не работает для полей объектов и колбеков, переданных в аргументах вызова

Алгоритм - поиск значений аргументов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g() {
 9     var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

data = *FROM_ARG*

Алгоритм - поиск значений аргументов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g() {
 9     var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

```
$.ajax call
args[0] = {
    url: ...,
    data: FROM_ARG,
     ...
}
```

Алгоритм - поиск значений аргументов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g() {
9      var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

```
$.ajax call
args[0] = {
    url: ...,
    data: FROM_ARG,
     ...
}
```

Алгоритм - поиск значений аргументов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g() {
9      var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

```
$.ajax call
args[0] = {
    url: ...,
    data: FROM_ARG,
     ...
}
```

Алгоритм

Межпроцедурный анализ - поиск значений аргументов вызывающей функции

- для каждого из найденных мест вызова анализ повторится от начала тела функции, содержащей искомое место вызова
- однако, в этот раз, дойдя до искомого вызова f, анализ перейдёт на начало f - но теперь с вычисленными фактическими значениями аргументов

Алгоритм - поиск значений аргументов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g() {
9      var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

Алгоритм - поиск значений аргументов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g() {
 9     var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

Алгоритм - поиск значений аргументов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g() {
9      var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

```
options = {
    recordID: 12,
    limit: 100
}
```

Алгоритм - поиск значений аргументов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g() {
9      var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

```
options = {
    recordID: 12,
    limit: 100
}
```

Алгоритм - поиск значений аргументов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g() {
9      var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

```
data = {
    recordID: 12,
    limit: 100
}
```

Алгоритм - поиск значений аргументов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g() {
 9     var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

```
$.ajax call
args[0] = {
    url: ...,
    data: {
        recordID: 12,
        limit: 100
    }
      ...
}
```

Алгоритм - поиск значений аргументов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g() {
9      var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

Алгоритм - поиск значений аргументов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g() {
 9     var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

Алгоритм - поиск значений аргументов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g() {
 9     var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

rid = 34

Алгоритм - поиск значений аргументов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g() {
 9     var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

rid = 34

Алгоритм - поиск значений аргументов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g() {
9      var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

```
data = {
    recordID: 34,
    sort: 'desc'
}
```

Алгоритм - поиск значений аргументов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g() {
 9     var options = { recordID: 12, limit: 100 };
10     f(options);
11 }
12 function k() {
13     const rid = 34;
14     f({ recordID: rid, sort: 'desc' });
15 }
```

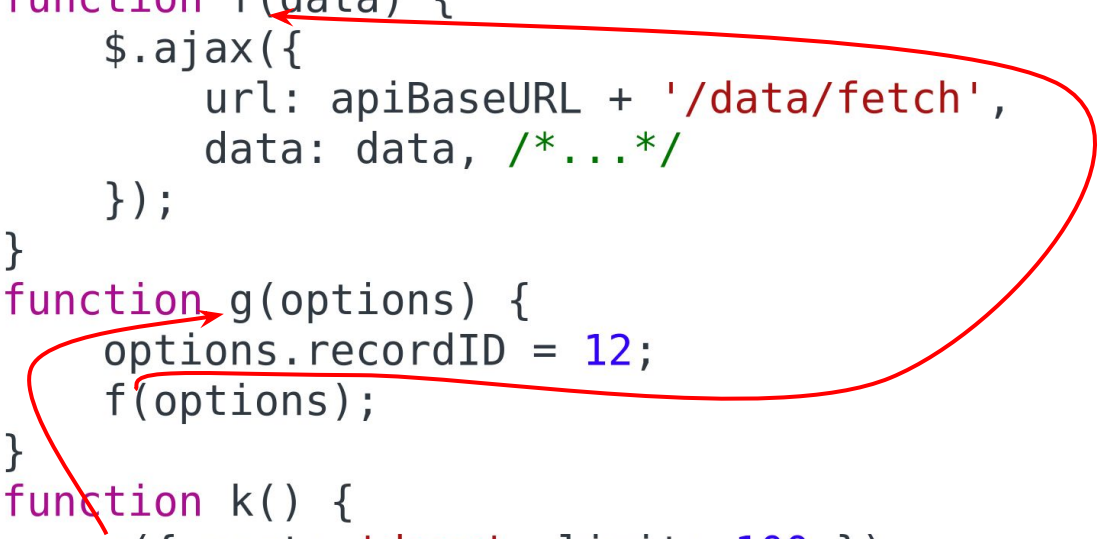
```
$.ajax call
args[0] = {
    url: ...,
    data: {
       recordID: 34,
       sort: 'desc'
    }
    ...
}
```

Цепочка из нескольких вызовов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g(options) {
9      options.recordID = 12;
10     f(options);
11 }
12 function k() {
13     g({ sort: 'desc', limit: 100 });
14 }
```

Цепочка из нескольких вызовов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g(options) {
9      options.recordID = 12;
10     f(options);
11 }
12 function k() {
13     g({ sort: 'desc', limit: 100 });
14 }
```



Цепочка из нескольких вызовов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g(options) {
 9     options.recordID = 12;
10     f(options);
11 }
12 function k() {
13     g({ sort: 'desc', limit: 100 });
14 }
```

```
$.ajax call
args[0] = {
    url: ...,
    data: FROM_ARG,
     ...
}
```

Цепочка из нескольких вызовов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g(options) {
 9     options.recordID = 12;
10     f(options);
11 }
12 function k() {
13     g({ sort: 'desc', limit: 100 });
14 }
```

options = *FROM_ARG*

Цепочка из нескольких вызовов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g(options) {
 9     options.recordID = 12;
10     f(options);
11 }
12 function k() {
13     g({ sort: 'desc', limit: 100 });
14 }
```

```
data = FROM_ARG
```

Цепочка из нескольких вызовов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g(options) {
9      options.recordID = 12;
10     f(options);
11 }
12 function k() {
13     g({ sort: 'desc', limit: 100 });
14 }
```

```
data = FROM_ARG
```

для этого анализатор поддерживает очередь цепочек вызова и текущую анализируемую цепочку

Цепочка из нескольких вызовов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g(options) {
 9     options.recordID = 12;
10     f(options);
11 }
12 function k() {
13     g({ sort: 'desc', limit: 100 });
14 }
```

для этого анализатор поддерживает очередь цепочек вызова и текущую анализируемую цепочку

Цепочка из нескольких вызовов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g(options) {
 9     options.recordID = 12;
10     f(options);
11 }
12 function k() {
13     g({ sort: 'desc', limit: 100 });
14 }
```

```
options = {
    sort: "desc",
    limit: 100
}
```

для этого анализатор поддерживает очередь цепочек вызова и текущую анализируемую цепочку

Цепочка из нескольких вызовов

```
1  var apiBaseURL = location.origin + '/api/v2.0/';
2  function f(data) {
3      $.ajax({
4          url: apiBaseURL + '/data/fetch',
5          data: data, /*...*/
6      });
7  }
8  function g(options) {
9      options.recordID = 12;
10     f(options);
11 }
12 function k() {
13     g({ sort: 'desc', limit: 100 });
14 }
```

```
options = {
    sort: "desc",
    limit: 100,
    recordID: 12
}
```

для этого анализатор поддерживает очередь цепочек вызова и текущую анализируемую цепочку

Цепочка из нескольких вызовов

```
 1 var apiBaseURL = location.origin + '/api/v2.0/';
 2 function f(data) {
 3     $.ajax({
 4         url: apiBaseURL + '/data/fetch',
 5         data: data, /*...*/
 6     });
 7 }
 8 function g(options) {
 9     options.recordID = 12;
10     f(options);
11 }
12 function k() {
13     g({ sort: 'desc', limit: 100 });
14 }
```

```
$.ajax call
args[0] = {
    url: ...,
    data: {
        sort: "desc",
        limit: 100,
        recordID: 12
    },
      ...
}
```

для этого анализатор поддерживает очередь цепочек вызова и текущую анализируемую цепочку

Алгоритм - сложности

- JS очень сложно анализировать статически
  проблема для всех, наш анализатор - не исключение
- динамическая загрузка кода и `eval()`
- бандлеры, ООП (`this`), DOM и его квирки
- не найдётся:
  - `this.makeRequest("endpoint", {x: 1});`
  - `a = require("jquery"); a.get("endpoint", {x: 1});`

Интеграция с динамическим анализом (сделано)

Используем управляемый браузер для загрузки страницы, получаем код через дебаггер. В результате:

- получаем весь JS-код на странице, включая загруженный динамически другим JS
- получаем самые простые динамические запросы из JS - которые сделаются при загрузке страницы
- получаем доступ к DOM: анализ может прочесть данные, которые код берёт оттуда
  (`var url = $('.main-container').attr('data-serverurl')`)

Интеграция с динамическим анализом (идеи)

- можно “подсматривать” значения переменных в динамике
  - данных (скажем, `baseURL`)
  - функций, модулей, объектов с функциями/данными
- можно пробовать вызывать отдельные функции на странице
  - в надежде что они отправят запрос
  - или что при их вызове мы в динамике увидим значение интересующей переменной

Эксперименты

⟶ какие уязвимости можно найти в JS-эндпоинтах в реальности?

⟶ какие из уязвимых эндпоинтов можно найти нашим анализом? Для каких он требуется?

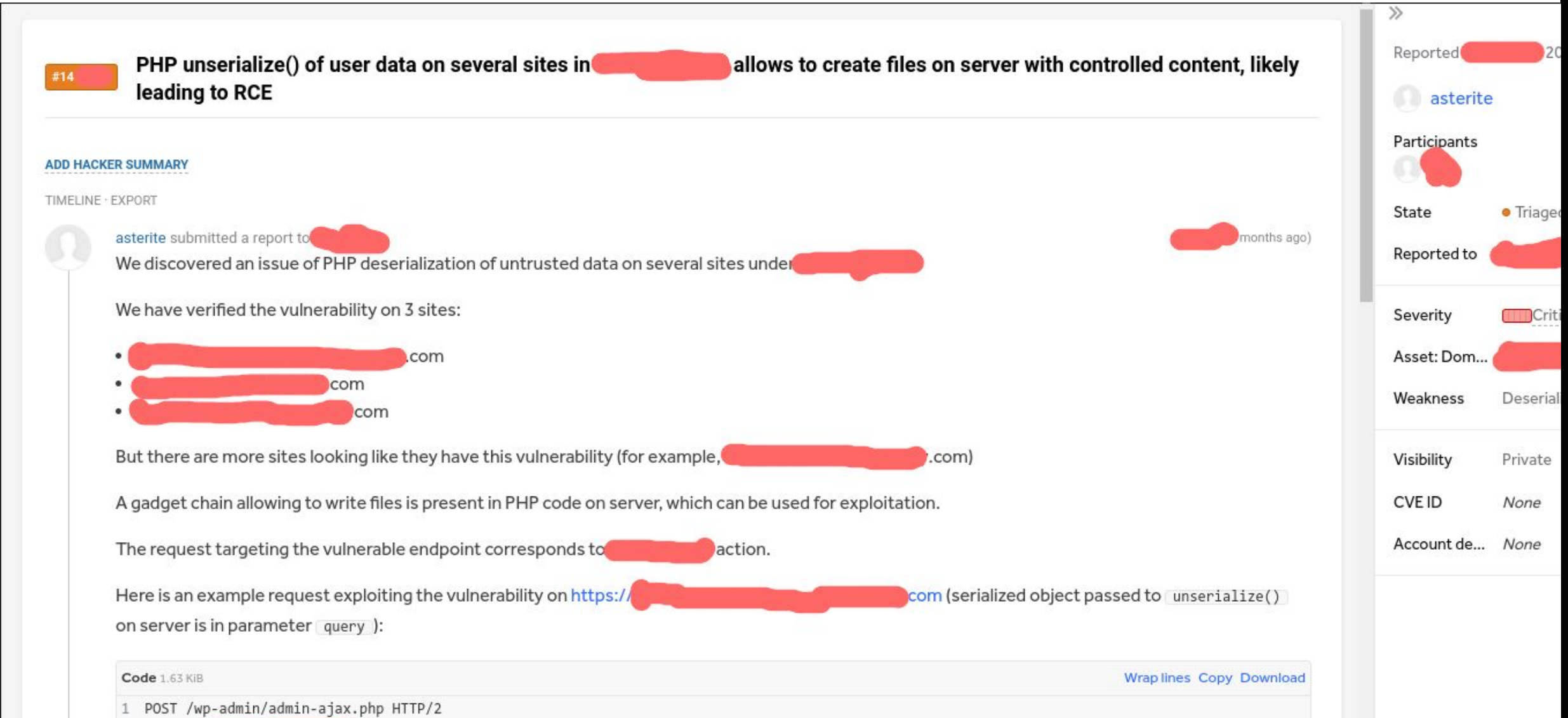
Эксперименты - реальные уязвимости

- CVE-2020-13640
  SQL injection in wpDiscuz WordPress plugin

- CVE-2022-24108 0day
  PHP unserialize() in OpenCart CMS plugin leads to RCE

Эксперименты - реальные уязвимости

Bug bounty: PHP unserialize



#14 **PHP unserialize() of user data on several sites in [redacted] allows to create files on server with controlled content, likely leading to RCE**

ADD HACKER SUMMARY

TIMELINE · EXPORT

asterite submitted a report to [redacted] [redacted] months ago)

We discovered an issue of PHP deserialization of untrusted data on several sites under [redacted]

We have verified the vulnerability on 3 sites:

- [redacted].com
- [redacted]com
- [redacted]com

But there are more sites looking like they have this vulnerability (for example, [redacted].com)

A gadget chain allowing to write files is present in PHP code on server, which can be used for exploitation.

The request targeting the vulnerable endpoint corresponds to [redacted] action.

Here is an example request exploiting the vulnerability on https://[redacted]com (serialized object passed to `unserialize()` on server is in parameter `query`):

Code 1.63 KiB — Wrap lines Copy Download

```
1 POST /wp-admin/admin-ajax.php HTTP/2
```

Reported [redacted] 20

asterite

Participants [redacted]

State ● Triaged

Reported to [redacted]

Severity [redacted]Criti

Asset: Dom... [redacted]

Weakness Deserial

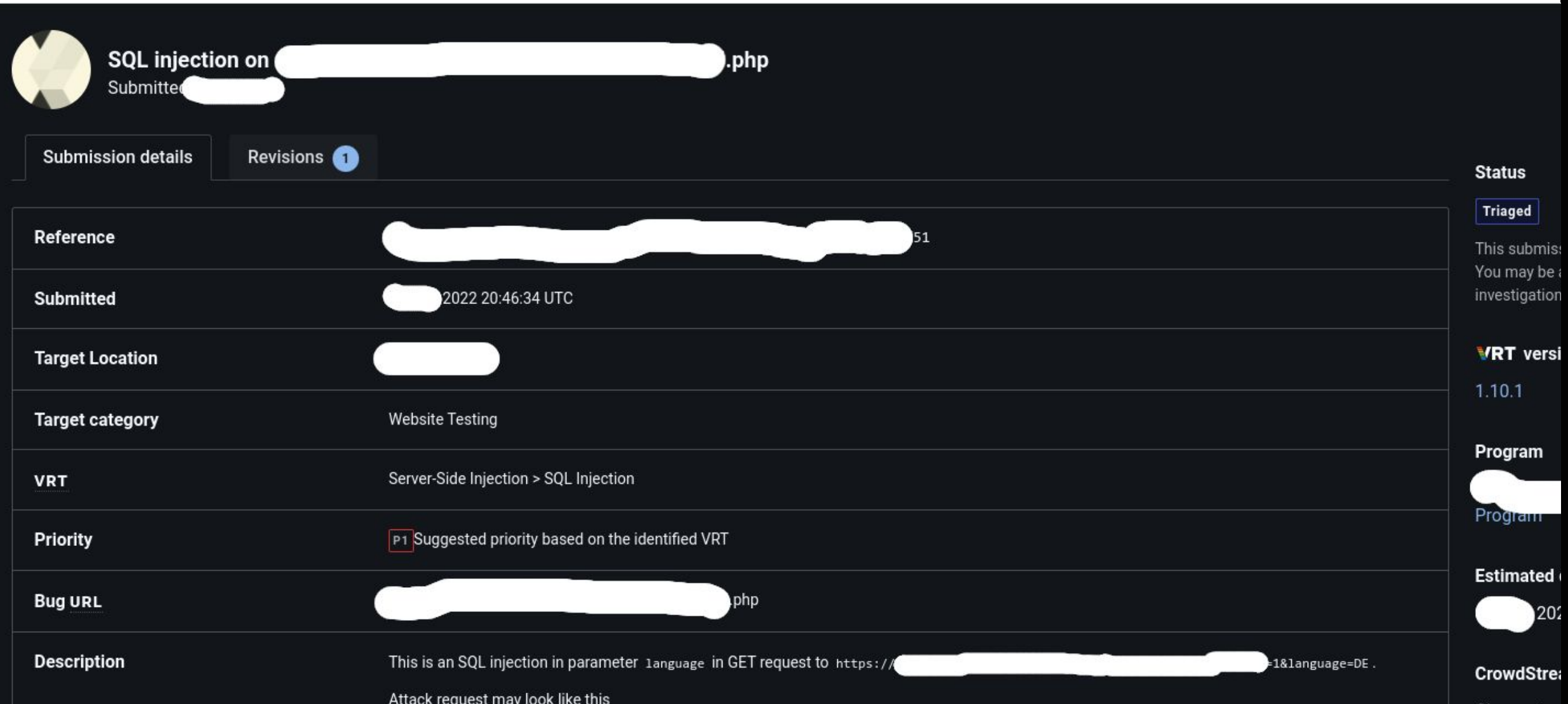
Visibility Private

CVE ID *None*

Account de... *None*

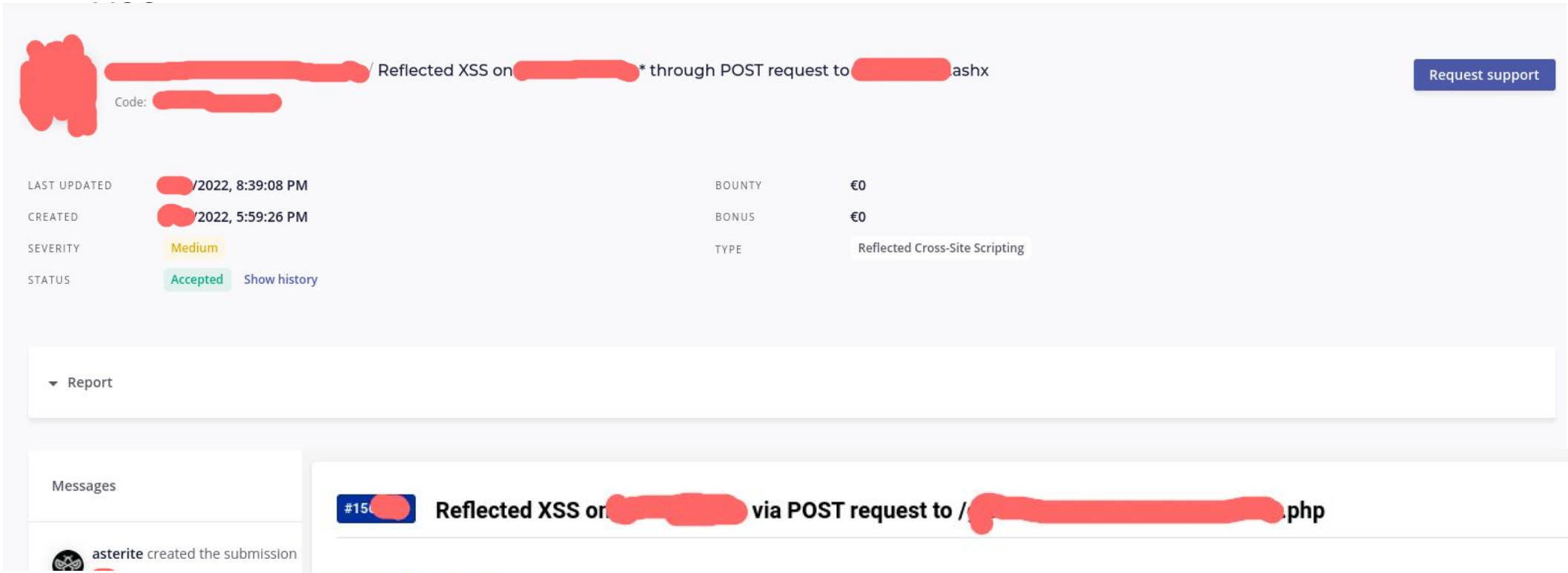
Эксперименты - реальные уязвимости

Bug bounty: SQL injection



Эксперименты - реальные уязвимости

Bug bounty: несколько Reflected



Эксперименты - реальные уязвимости

- 2 CVE
  - анализатор находит эти эндпоинты
  - запросы не делаются при загрузке страницы (чтобы найти в динамике нужно взаимодействие с элементами)

- BugBounty
  - 1 PHP unserialize, 1 SQLi, 6 Reflected XSS
  - из них PHP unserialize, SQLi и 1 Reflected XSS не нашлись в динамике (т. е. нашлись только анализатором)

Что анализ найдёт - пример 1 (bug bounty)

```
$('.ajax-car').each(function (x, y) { // ...
    $('#search_car').on('keyup', function () {
        runAjaxCarSearch(y);
    });
}); // ...

var runAjaxCarSearch = function (y) {
    if ($('#search_car').val().length > 3) {
        url = "/public/api/car-search.php";
        params += "model=" + $('#search_car').val()/*...*/
        $.ajax({ url: url + '?' + params }); // ...
    }
}
```

Что анализ найдёт - пример 1 (bug bounty)

```
$('.ajax-car').each(function (x, y) { // ...
    $('#search_car').on('keyup', function () {
        runAjaxCarSearch(y);
    });
}); // ...

var runAjaxCarSearch = function (y) {
    if ($('#search_car').val().length > 3) {
        url = "/public/api/car-search.php";
        params += "model=" + $('#search_car').val()/*...*/
        $.ajax({ url: url + '?' + params }); // ...
    }
}
```

Что анализ найдёт - пример 2 (bug bounty)

```
$('#submit_form').click(function(e) { //...
  $firstNameValid = $('#formFirstName').data('valid');
  // ...

  if ($emailValid && $companyValid && $firstNameValid && $lastNameValid && $clientTypeValid && $phoneValid && $optInValid && $clientType && $clientsValid && $employeesValid && $endpointsValid ) {
    // ...
    $firstName = $('#formFirstName').val();
    $lastName = $('#formLastName').val();
    $companyName = $('#formCompanyName').val();
    $phone = $('#formPhone').val();
    $email = $('#formMail').val();
    $clientType = $('#formClientType').val();
    $settingTypes = $('#formSettingTypes').val();
    $clientsPosition = $('#formClientsPosition').val();
    $employeesPosition = $('#formEmployeePosition').val();
    $firstTime = $('.formHolder .enableBtn:checked').val();
    $App_Metric_Campaign = $('.App_Metric_Campaign').val();
    $App_Metric_Content = $('.App_Metric_Content').val();
    $App_Metric_Medium = $('.App_Metric_Medium').val();
    $App_Metric_Source = $('.App_Metric_Source').val();
    $App_Metric_Region = $('.App_Metric_Region').val();
    $App_Metric_Category = $('.App_Metric_Category').val();
    $App_Metric_Convert_URL = $('.App_Metric_Convert_URL').val();
    $AuxInfo = $('.AuxInfo').val();
    $message = $('.message').val();

    $formID = $('#formID').val();
    $ga_cid = $('.$ga_cid').val();

    $.post("/scripts/main/submitHandle.php",
    {
      formID : $formID,
      firstName : $firstName,
      lastName : $lastName,
      phone : $phone,
      companyName : $companyName,
      email : $email,
      ga_cid : $ga_cid,
      clientType : $clientType,
      settingTypes : $settingTypes,
      clientsPosition : $clientsPosition,
      employeePosition : $employeesPosition,
      firstTime : $firstTime,
      App_Metric_Campaign : $App_Metric_Campaign,
      App_Metric_Content : $App_Metric_Content,
      App_Metric_Medium : $App_Metric_Medium,
      App_Metric_Source : $App_Metric_Source,
      App_Metric_Region : $App_Metric_Region,
      App_Metric_Convert_URL : $App_Metric_Convert_URL,
      App_Metric_Category : $App_Metric_Category,
      AuxInfo : $AuxInfo,
      message : $message,
```

Что анализ найдёт - пример 2 (bug bounty)

```
$('#submit_form').click(function(e) { //...
  $firstNameValid = $('#formFirstName').data('valid');
  // ...

  if ($emailValid && $companyValid && $firstNameValid && $lastNameValid && $clientTypeValid && $phoneValid && $optInValid && $clientType && $clientsValid && $employeesValid && $endpointsValid ) {
    // ...
    $firstName = $('#formFirstName').val();
    $lastName = $('#formLastName').val();
    $companyName = $('#formCompanyName').val();
    $phone = $('#formPhone').val();
    $email = $('#formMail').val();
    $clientType = $('#formClientType').val();
    $settingTypes = $('#formSettingTypes').val();
    $clientsPosition = $('#formClientsPosition').val();
    $employeesPosition = $('#formEmployeePosition').val();
    $firstTime = $('.formHolder .enableBtn:checked').val();
    $App_Metric_Campaign = $('.App_Metric_Campaign').val();
    $App_Metric_Content = $('.App_Metric_Content').val();
    $App_Metric_Medium = $('.App_Metric_Medium').val();
    $App_Metric_Source = $('.App_Metric_Source').val();
    $App_Metric_Region = $('.App_Metric_Region').val();
    $App_Metric_Category = $('.App_Metric_Category').val();
    $App_Metric_Convert_URL = $('.App_Metric_Convert_URL').val();
    $AuxInfo = $('.AuxInfo').val();
    $message = $('.message').val();

    $formID = $('#formID').val();
    $ga_cid = $('.$ga_cid').val();

    $.post("/scripts/main/submitHandle.php",
    {
      formID : $formID,
      firstName : $firstName,
      lastName : $lastName,
      phone : $phone,
      companyName : $companyName,
      email : $email,
      ga_cid : $ga_cid,
      clientType : $clientType,
      settingTypes : $settingTypes,
      clientsPosition : $clientsPosition,
      employeePosition : $employeesPosition,
      firstTime : $firstTime,
      App_Metric_Campaign : $App_Metric_Campaign,
      App_Metric_Content : $App_Metric_Content,
      App_Metric_Medium : $App_Metric_Medium,
      App_Metric_Source : $App_Metric_Source,
      App_Metric_Region : $App_Metric_Region,
      App_Metric_Convert_URL : $App_Metric_Convert_URL,
      App_Metric_Category : $App_Metric_Category,
      AuxInfo : $AuxInfo,
      message : $message,
```

Что анализ найдёт - пример 2 (bug bounty)

```
$('#submit_form').click(function(e) { //...
  $firstNameValid = $('#formFirstName').data('valid');
  // ...

  if ($emailValid && $companyValid && $firstNameValid &&
      $lastNameValid &&
      $clientTypeValid && $phoneValid
      && $optInValid && $clientType &&
      $clientsValid && $employeesValid && $endpointsValid ) {
    // ...
    $firstName = $('#formFirstName').val();
    $lastName = $('#formLastName').val();
    $companyName = $('#formCompanyName').val();
    $phone = $('#formPhone').val();
    $email = $('#formMail').val();
    $clientType = $('#formClientType').val();
    $settingTypes = $('#formSettingTypes').val();
    $clientsPosition = $('#formClientsPosition').val();
    $employeesPosition = $('#formEmployeePosition').val();
    $firstTime = $('.formHolder .enableBtn:checked').val();
    $App_Metric_Campaign = $('.App_Metric_Campaign').val();
    $App_Metric_Content = $('.App_Metric_Content').val();
```

Что анализ найдёт - пример 2 (bug bounty)

```
$('#submit_form').click(function(e) { //...
  $firstNameValid = $('#formFirstName').data('valid');
  // ...

  if ($emailValid && $companyValid && $firstNameValid &&
      $lastNameValid &&
      $clientTypeValid && $phoneValid
      && $optInValid && $clientType &&
      $clientsValid && $employeesValid && $endpointsValid ) {
    // ...
    $firstName = $('#formFirstName').val();
    $lastName = $('#formLastName').val();
    $companyName = $('#formCompanyName').val();
    $phone = $('#formPhone').val();
    $email = $('#formMail').val();
    $clientType = $('#formClientType').val();
    $settingTypes = $('#formSettingTypes').val();
    $clientsPosition = $('#formClientsPosition').val();
    $employeesPosition = $('#formEmployeePosition').val();
    $firstTime = $('.formHolder .enableBtn:checked').val();
    $App_Metric_Campaign = $('.App_Metric_Campaign').val();
    $App_Metric_Content = $('.App_Metric_Content').val();
```

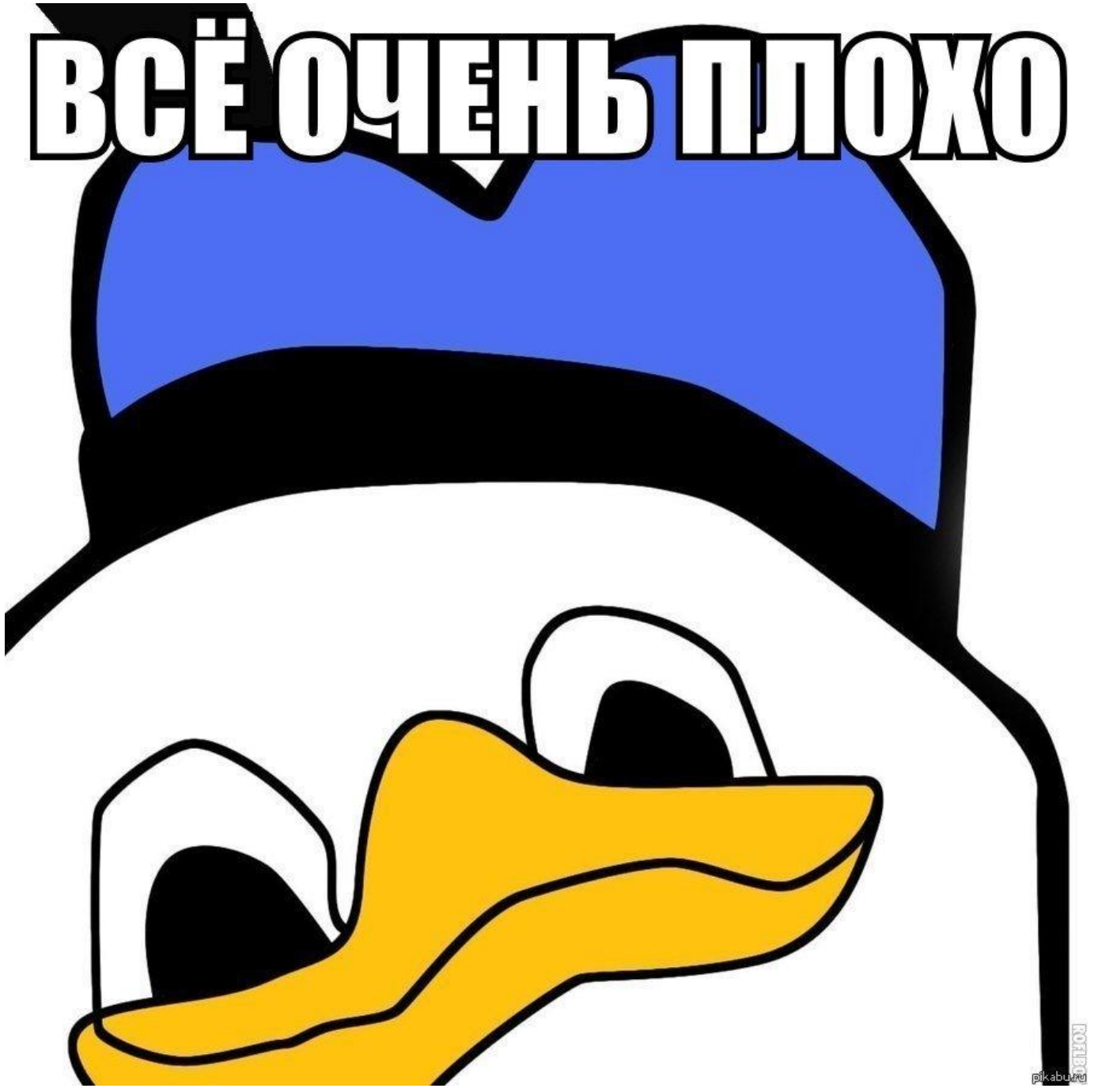
Что анализ найдёт - пример 2 (bug bounty)

```
$('#submit_form').click(function(e) { //...
  $firstNameValid = $('#formFirstName').data('valid');
  // ...
  // 11 conditions
  if ($emailValid && $companyValid && $firstNameValid &&
      $lastNameValid &&
      $clientTypeValid && $phoneValid
      && $optInValid && $clientType &&
      $clientsValid && $employeesValid && $endpointsValid ) {
    // ...
    $firstName = $('#formFirstName').val();
    // total 20 lines
    $ga_cid = $('.$ga_cid').val();

    $.post("/scripts/main/submitHandle.php",
    {
      formID : $formID,
      firstName : $firstName,
      // total 21 parameter

    },
```

Пример 2 - что на такое скажет краулер?



Что анализ найдёт - пример 3 (CVE)

```
function wpdiscuzLoadComments(loadButton, loaded, loading) {
    // ...
    var data = new FormData();
    data.append('action', 'wpdLoadMoreComments');
    data.append('lastParentId', getLastParentID());
    // ...
    var ajax = getAjaxObj(true, data);
    // ...
}
function getAjaxObj(isShowTopLoading, data) {
    //...
    data.append('postId', wpdiscuzPostId);
    return $.ajax({
        type: 'POST',
        url: wpdiscuzAjaxObj.url,
        data: data, // ...
    });
}
```

Найти это краулингом можно не всегда - это подгрузка доп. комментариев, комментариев может быть просто слишком мало

Что анализ найдёт

- Несложный код отправки запросов, old-school код в “стиле Си” (“jQuery-стиле”)
- Такое нередко встречается в плагинах WordPress
- Может быть и у современного сайта (код сайта может быть в целом сложным, но отправка AJAX-запросов может быть простой)

Что анализ найдёт

При этом запросы, которые находит анализ, могут:

- требовать сложной последовательности пользовательских действий
- быть такими что их вообще нельзя отправить из пользовательского интерфейса (“*скрытые функции*”). Мёртвый код

“Скрытые” функции веб-приложения

- Функции, доступные только в **авторизованной зоне** (админка)
  - такие эндпоинты могут быть всё равно доступны из-за уязвимости авторизации
  - пример - рендеринг markdown в Jira (CVE-2020-14181)

- Функции, которые забыли удалить (например дебаг-функции)

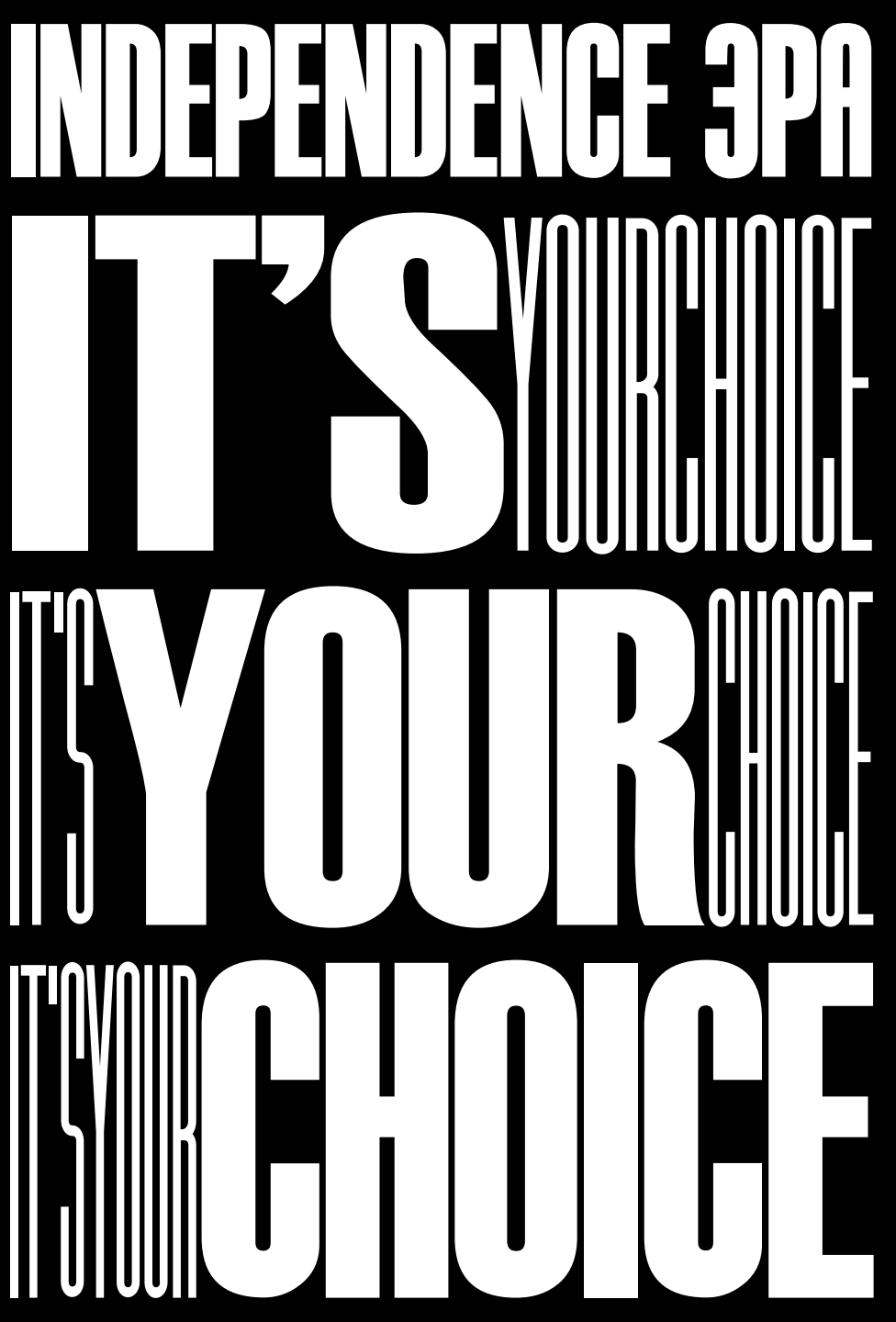
Когда краулинг сработает лучше

Запросы, которые можно стриггерить простым действием, но в отправке которых вовлечено много сложного JS-кода

- часть запросов отправится просто при загрузке страницы
- современные фреймворки и библиотеки бывают большими и сложными
- бандлеры, ООП, …

Демо

https://phd2022.solidwall.io



Заключение

- Статический анализ может работать точнее, интегрируясь с динамическим

- Наше видение: лучше сначала сделать простой и наивный механизм, который работает, а потом сложный, который его превзойдёт, чем сразу сложный

Заключение

- Ни один метод не является панацеей. Анализ дополняет краулинг, а не заменяет его
- Современным сканерам стоит использовать анализ клиентского JS-кода для майнинга эндпоинтов
- На текущий момент наибольшее покрытие даст использование обоих методов

phd Positive Hack Days

Спасибо

https://phd2022.solidwall.io