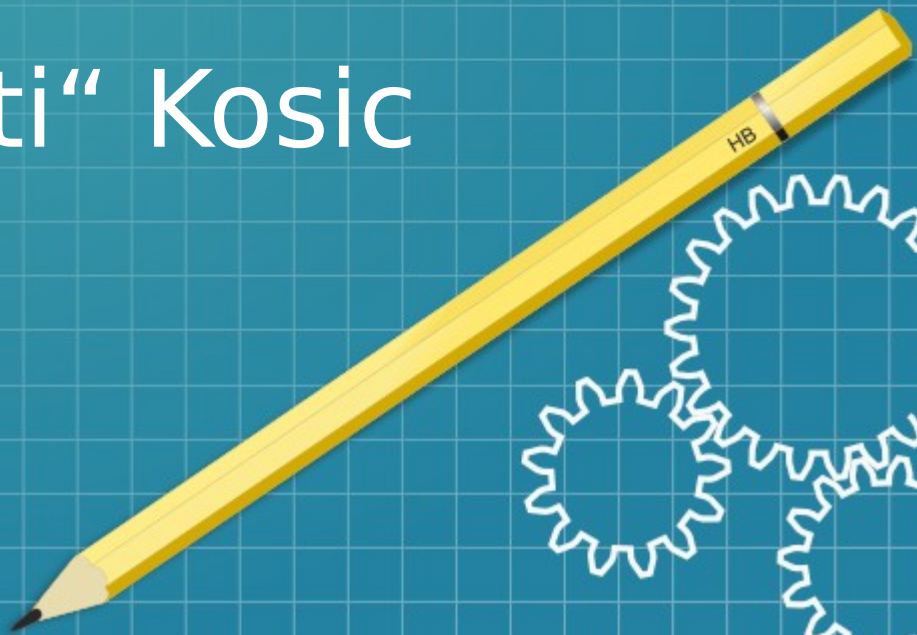
Shittr

ENOWARS 3

by Petar „Hetti“ Kosic

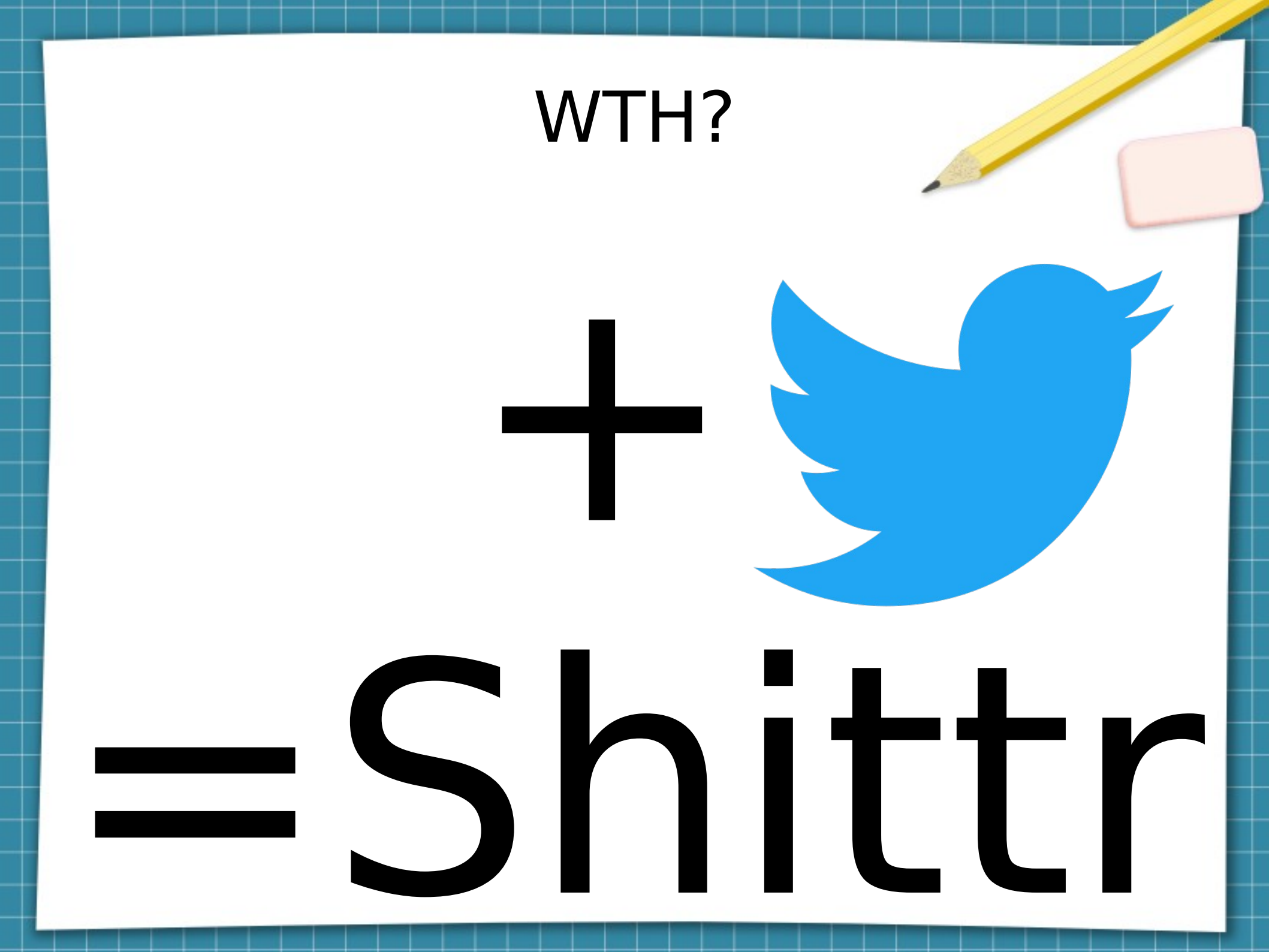




Enowars 3

- Attack-Defense CTF
- IP(v6) only Network
- Was set up during a course at university
- 05.07.2019 – 10 hours CTF
- Organized by our Friends ENOFLAG from Berlin (TU Berlin)

WTH?



=Shittr

Webserver (Reallife)

nginx

Apache

IIS

Webserver (CTF-Life)

Bash

Bash all the way

- Completly in bash written
- Custom HTTP Return codes → later
- Used sed, cat and other tools
- selfwritten openssl library for encryption and HTTPS

Lets check

the interface

Vulns that we found

Our Approach

- Analyse log of application
- Play around with the web interface
- Investigate Source Code

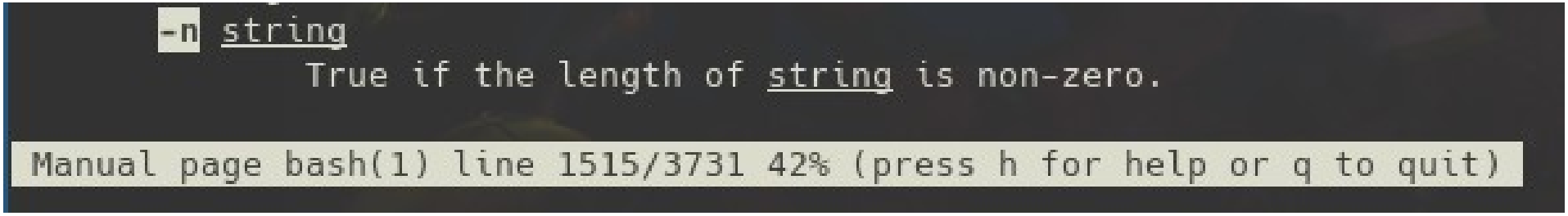
- First write the exploits to attack other teams
- Then fix the bugs we found

DEBUG Flag :D

middlewares.sh

```
17  is_admin() {
18      if [[ "$USER" =~ "admin" && -n "$DEBUG" ]]
19      then
20          ADMIN=1
21      else
22          ADMIN=0
23      fi
24  }
25
```

-n in Bash



So DEBUG = 1 & DEBUG = 0 evaluate both to true

Admin Account Regex

middlewares.sh

```
17  is_admin() {
18      if [[ "$USER" =~ "admin" && -n "$DEBUG" ]]
19      then
20          ADMIN=1
21      else
22          ADMIN=0
23      fi
24  }
25
```

=~ in Bash

- Operator for regex
- $USER =~ „admin“ matches every Username that contains the word „admin“

Admin: /log

```
Tue Oct 29 12:11:06 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] /static/js/vendor/popper.min.js
Tue Oct 29 12:11:06 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] GET, /static/js/vendor/popper.min.js, HTTP/1.1
Tue Oct 29 12:11:06 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] GET /static/js/vendor/popper.min.js HTTP/1.1
Tue Oct 29 12:11:06 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff]
Tue Oct 29 12:11:06 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] /home
Tue Oct 29 12:11:06 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] GET, /home, HTTP/1.1
Tue Oct 29 12:11:06 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] GET /home HTTP/1.1
Tue Oct 29 12:11:06 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] Session is 060558009637782212400000000
Tue Oct 29 12:11:06 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff]
Tue Oct 29 12:11:05 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] POST, /login, HTTP/1.1
Tue Oct 29 12:11:05 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] POST /login HTTP/1.1
Tue Oct 29 12:11:02 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff]
Tue Oct 29 12:11:02 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] /favicon.ico
Tue Oct 29 12:11:01 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] GET, /favicon.ico, HTTP/1.1
Tue Oct 29 12:11:01 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] GET /favicon.ico HTTP/1.1
Tue Oct 29 12:10:58 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff]
Tue Oct 29 12:10:58 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] /favicon.ico
Tue Oct 29 12:10:58 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] GET, /favicon.ico, HTTP/1.1
Tue Oct 29 12:10:58 UTC 2019, [fd00:1337:0001:8454:0000:0000:0000:ffff] GET /favicon.ico HTTP/1.1
```

What did we patch?

- DEBUG „Flag“
- Admin Regex

Our approach to get flags

- Use python and initially requests
  - Request run away
  - Implemented direct call of curl as system command instead
- Scraped all user names
- Visited their site and scraped the flag

Custom HTTP Codes

```
17    declare -A STATUS=(
18        [1337]="WORKS FOR ME"
19        [302]="TRY AGAIN"
20        [403]="GTFO"
21        [404]="NOPE"
22        [4242]="IT BURNS!!!"
23    )
```

Custom HTTP Codes
+
requests in Python

```
During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "./shittr.py", line 80, in <module>
    cookies = register_login()
  File "./shittr.py", line 51, in register_login
    r = requests.post(url+"register", data=data, verify=False, allow_redirects=False, stream=True)
  File "/usr/lib/python3.7/site-packages/requests/api.py", line 116, in post
    return request('post', url, data=data, json=json, **kwargs)
  File "/usr/lib/python3.7/site-packages/requests/api.py", line 60, in request
    return session.request(method=method, url=url, **kwargs)
  File "/usr/lib/python3.7/site-packages/requests/sessions.py", line 533, in request
    resp = self.send(prep, **send_kwargs)
  File "/usr/lib/python3.7/site-packages/requests/sessions.py", line 646, in send
    r = adapter.send(request, **kwargs)
  File "/usr/lib/python3.7/site-packages/requests/adapters.py", line 498, in send
    raise ConnectionError(err, request=request)
requests.exceptions.ConnectionError: ('Connection aborted.', BadStatusLine('HTTP/1.0 4242 IT BURNS!!!\n'))
✘ hetti@sternenregen ~/Desktop/University/CTF
```

Even more bugs

- Couple of Infoleaks
- Visibility Bypass
- RCE
- Crypto Flaws
- Admin Bypass

- See Github repo for more details

Lessons learned

- There is a curl library for python
- Requests library shits its pants with Custom HTTP Return Codes
- We didn‘t found all the bugs
- Accounts registered with „admin“ in the middle of account name ARE suspicious!

impact of this security threat in a realistic scenario

- Completly broken application

possible countermeasures

- Don‘t write your own crypto
- Don‘t write your own webserver in Bash

- Do(n‘t) write Custom HTTP Return Codes
- Implement proper registration and authentification
- Use Mastodon

Links

- https://github.com/enowars/enowars3-service-shittr
- https://enowars.com/
- https://moseskonto.tu-berlin.de/moses/modultransfersystem/bolognamodule/beschreibung/anzeigen.html?number=40933&version=1&sprache=2
- https://ctftime.org/team/1438/
- https://twitter.com/gehaxelt
- https://www.internetwache.org/

TX
Questions?