MetaXSSploit

Bringing XSS in Pentesting

A journey in building a security tool

Claudio Criscione

@paradoxengine

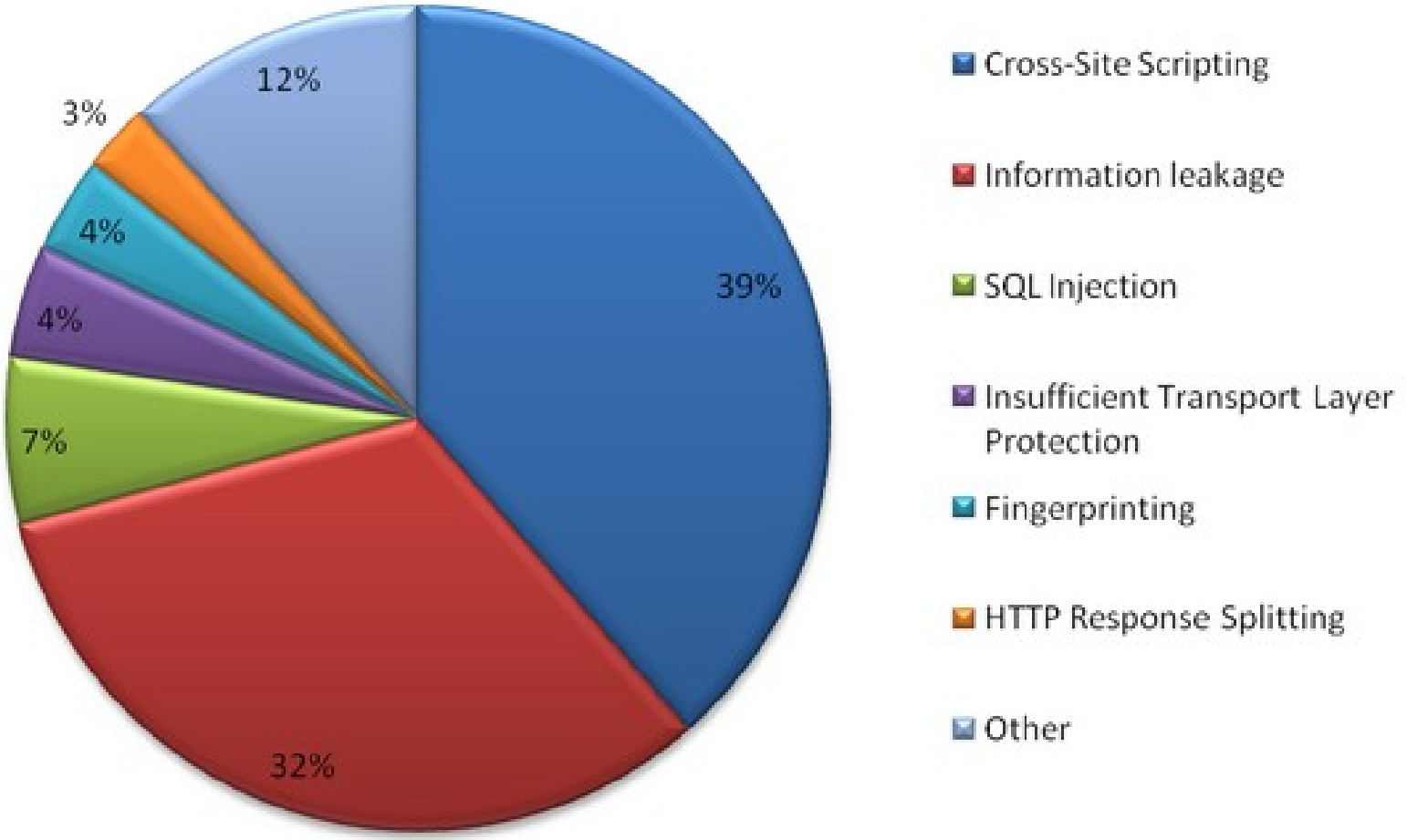
/me

```
blackfire@Thor:~/me$ ls -al
totale 72
drwxr-xr-x  2 blackfire blackfire  4096 2011-05-01 16:43 .
drwx------ 60 blackfire blackfire 20480 2011-05-01 16:41 ..
-rw-r--r--  1 blackfire blackfire     0 2011-05-01 16:43 Claudio Criscione
-rw-r--r--  1 blackfire blackfire     0 2011-05-01 16:41 ex-Pentester [10 years]
-rw-r--r--  1 blackfire blackfire     0 2011-05-01 16:41  No Aff
-rw-r--r--  1 blackfire blackfire     0 2011-05-01 16:43 Italian
blackfire@Thor:~/me$
```

XSS

And how a security tool is born!

Relevant?



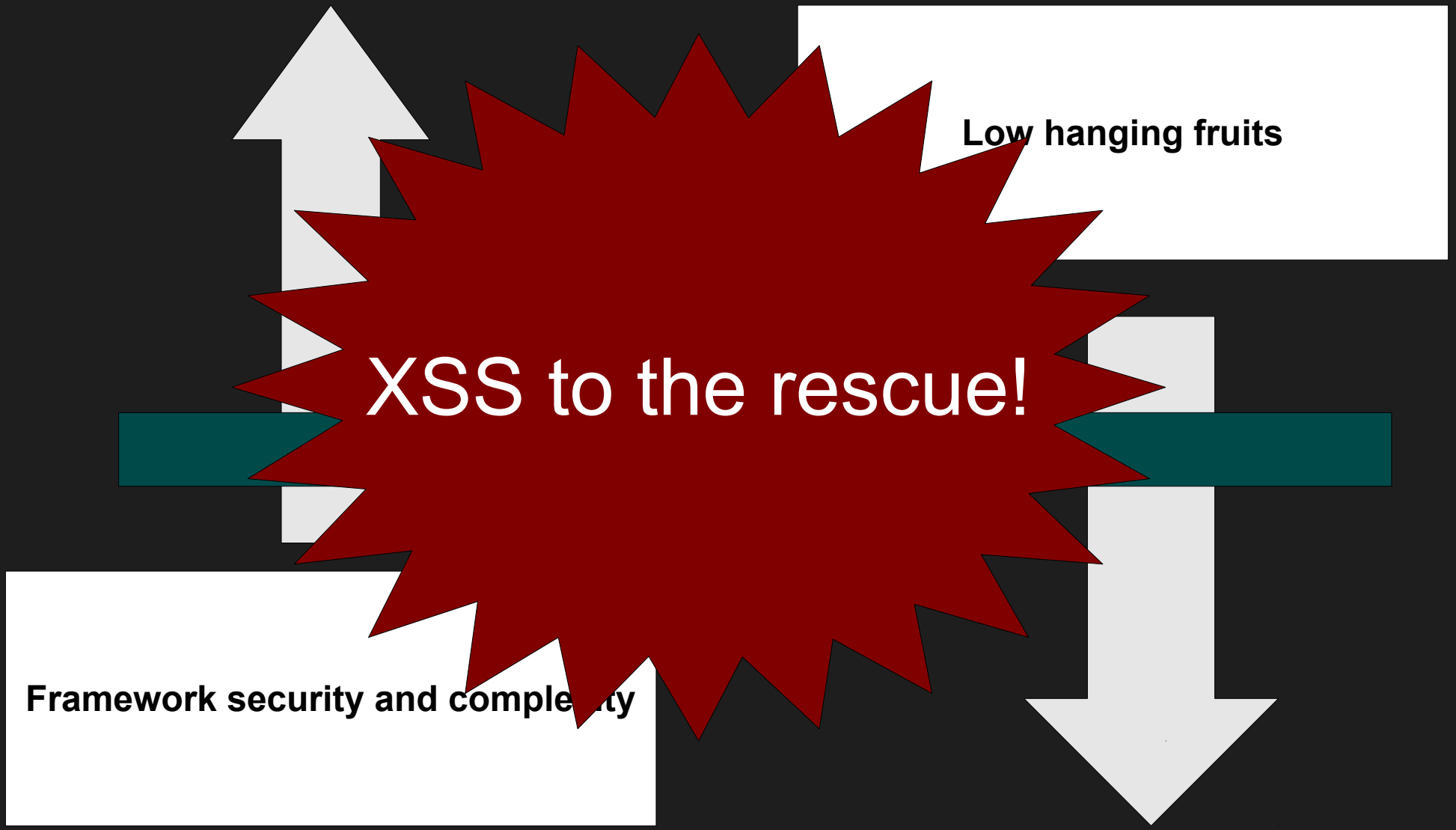
Web Application Security Statistics 2010 – Web Application Security Consortium
http://projects.webappsec.org/w/page/13246989/Web-Application-Security-Statistics

Relevant in the real world?


Apache
SOFTWARE FOUNDATION

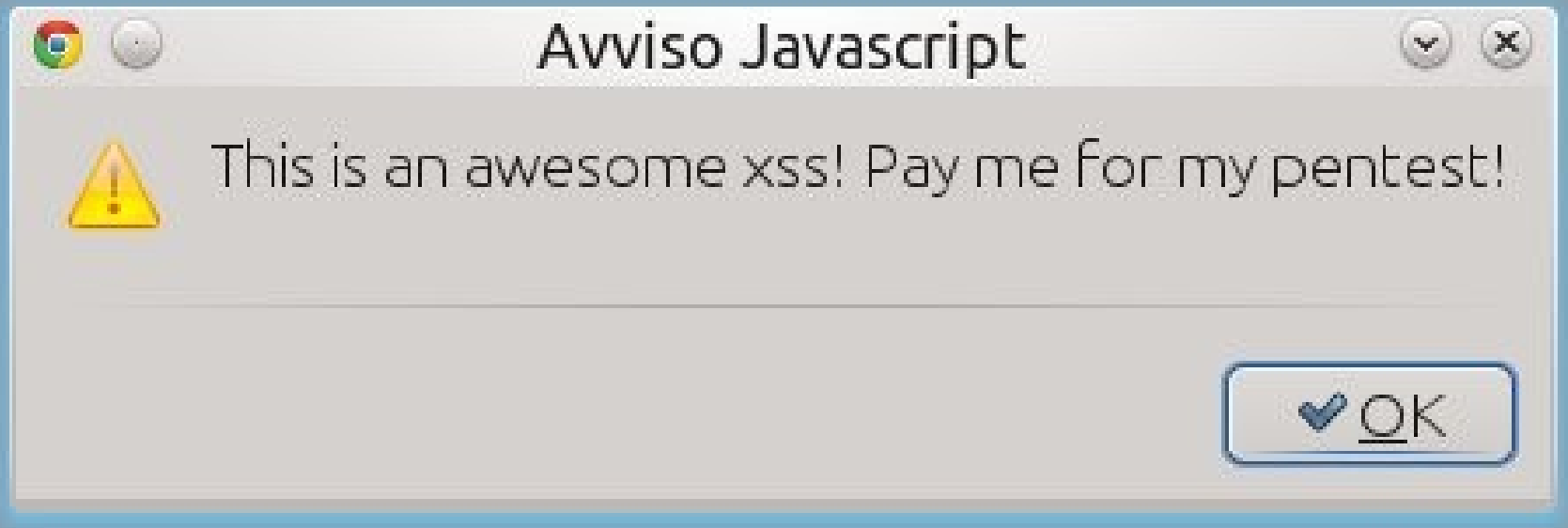
APACHE

A gift for pentesters




Ubiquitus
Easy to find
Hard to patch

In the beginning was the Alert()



Where does this approach bring us?

Standard customer answer
"There is no risk, it's just client side"

It is **our*** fault

***pentesters**

Seeing is believing

Since user interaction is required we can avoid **full real-world exploitation of the issue in the PT**

WRONG!

Using an Alert today vs actively exploiting the XSS is closer to running Nessus than using Metasploit

Why is this even worse?

"Oh, I've just found XSSes during this PT. This is lame :("

- The little hacker inside you


A perception mismatch

We **(all of us)** KNOW it is bad

We **(most of us)** don't feel it is "**that**" bad

Those **(most of them)** outside the community thinks it's more or less **meaningless**

How do we fix this?

We build a tool!

MetaXSSploit to the rescue

Bridging the gap between Metasploit and XSS

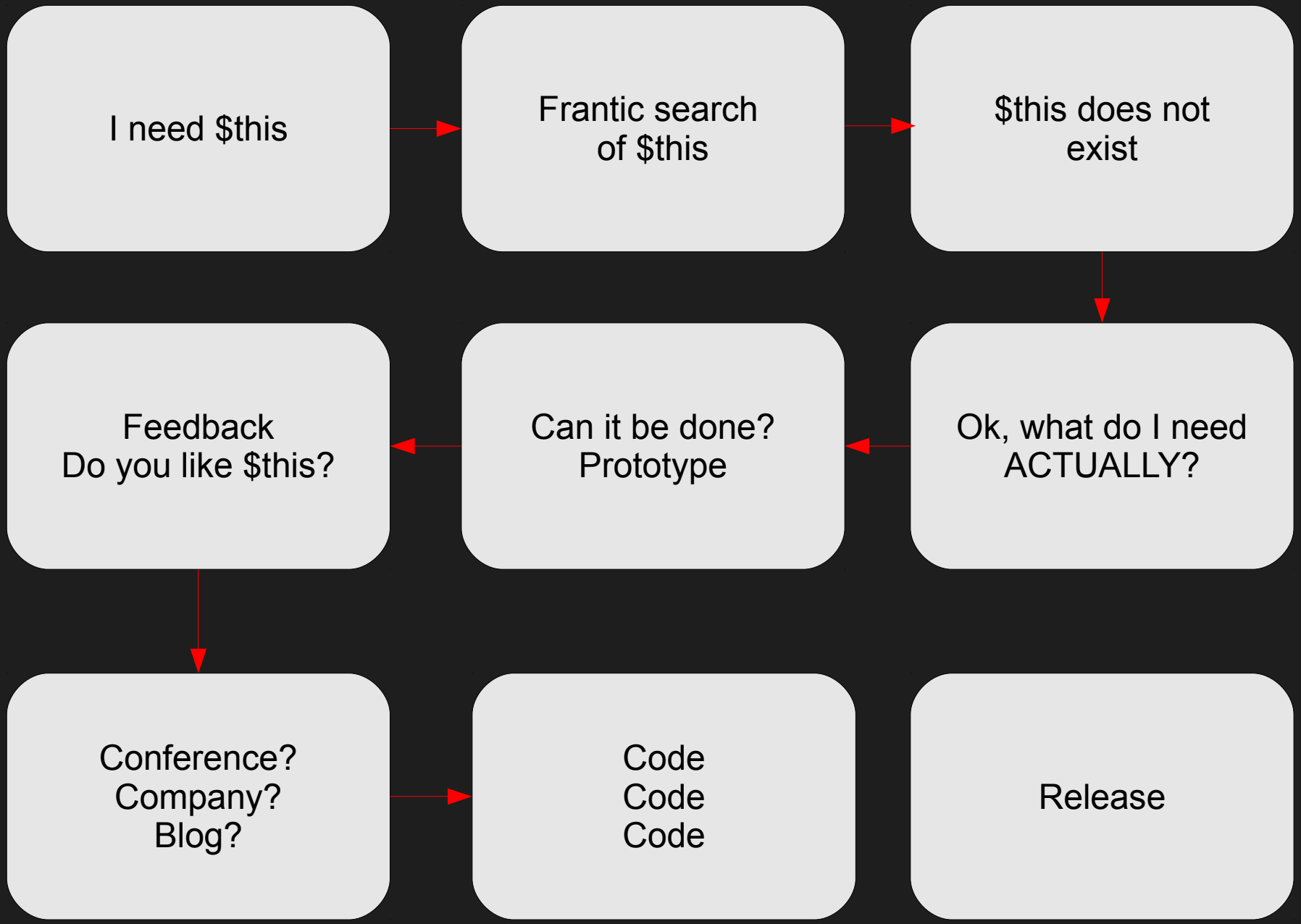
Weaponizing XSS

The story behind MetaXSSploit


It all began one stormy night
STONERIVER

Behind the scene [1]

- I had to code a Metasploit module for an XSS vulnerability in VMware to be included in VASTO (another project)
- "Darn, I have to write everything from scratch!"
- Lack of reusability of code, but the potential was out there
- Ok, let's do "something!"



YES, testing happens **AFTER** the release!

Timeline

November 2010 Had the idea

Set the Goals

I will make a tool

- That I'll want to use
- That other people will want to use
- That is Open Source
- That can be easily extended by other people
- That will speed up pentesting
- That will finally allow us to store the kowledge about XSS
- That is cool enough to be presented

Whenever you write new code...

Am I reinventing the wheel? Let's check!

Yes, there are relevant projects

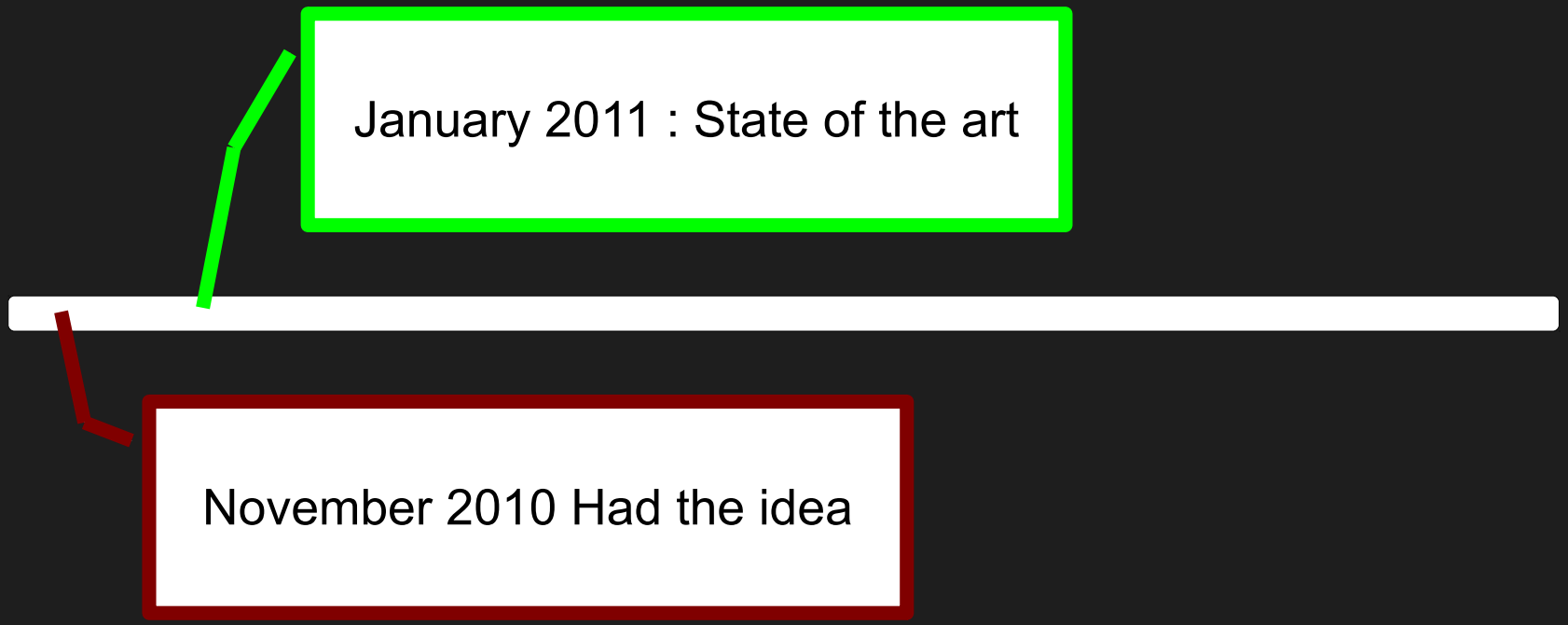
- XSSploiter
- XSSer
- Attack API and more...

Do they fulfill my goals?

- Not easy to extend (unless you learn ALL the tool!)
- It will take forever* to learn how they work

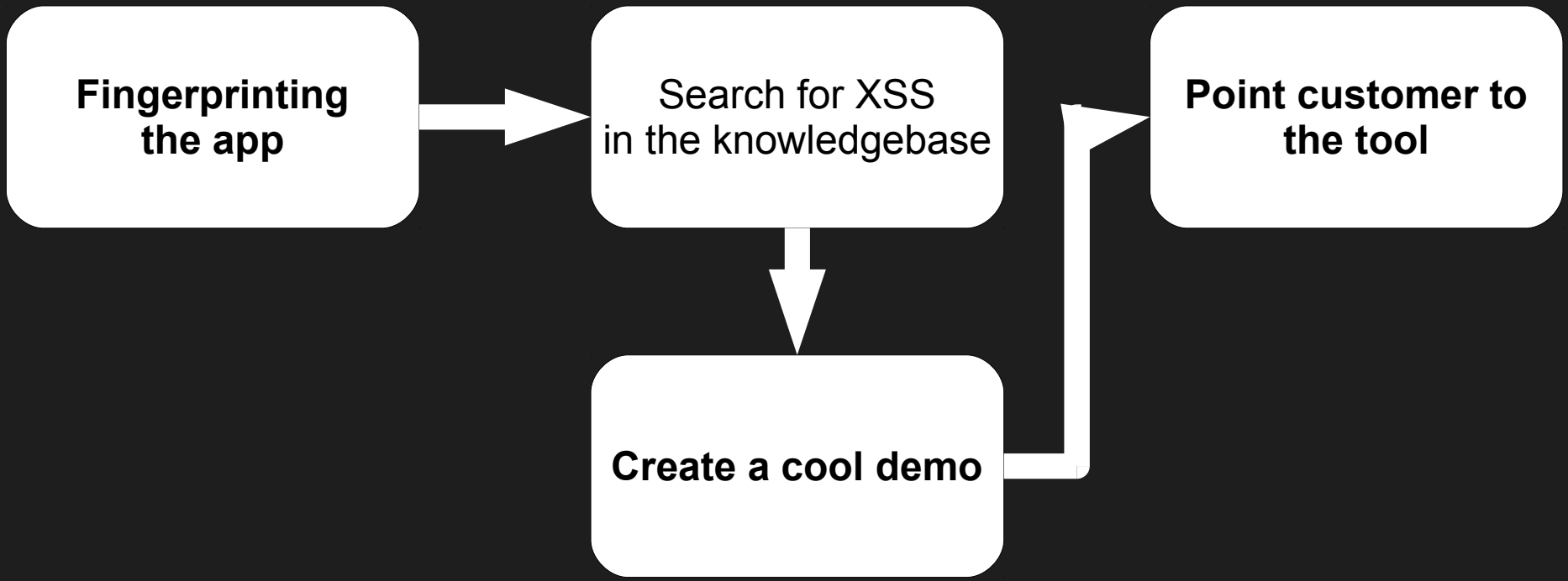
*more than 4 hours

Timeline

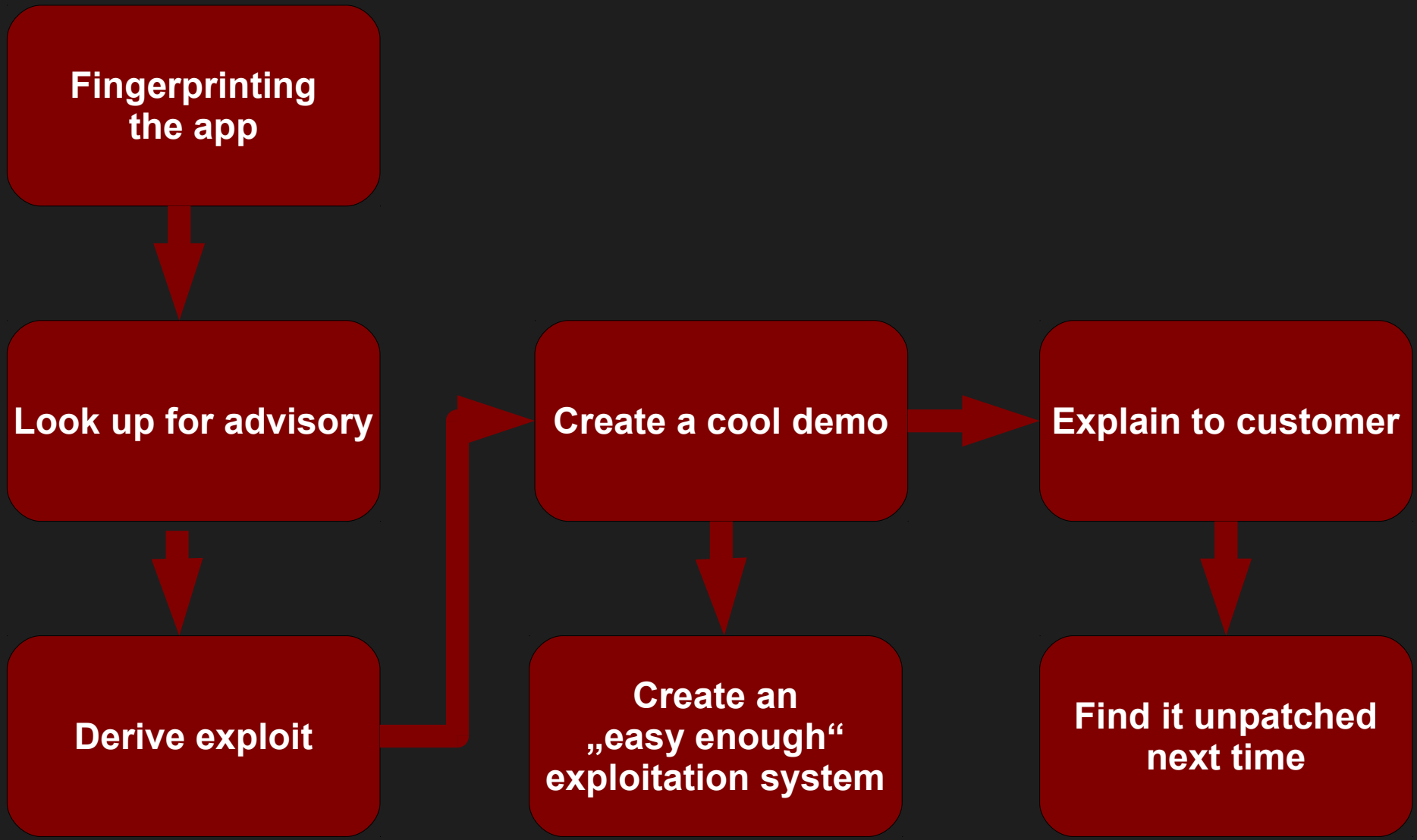


Draw use cases

Use case – Knowledge Base



Use case – Knowledge Base



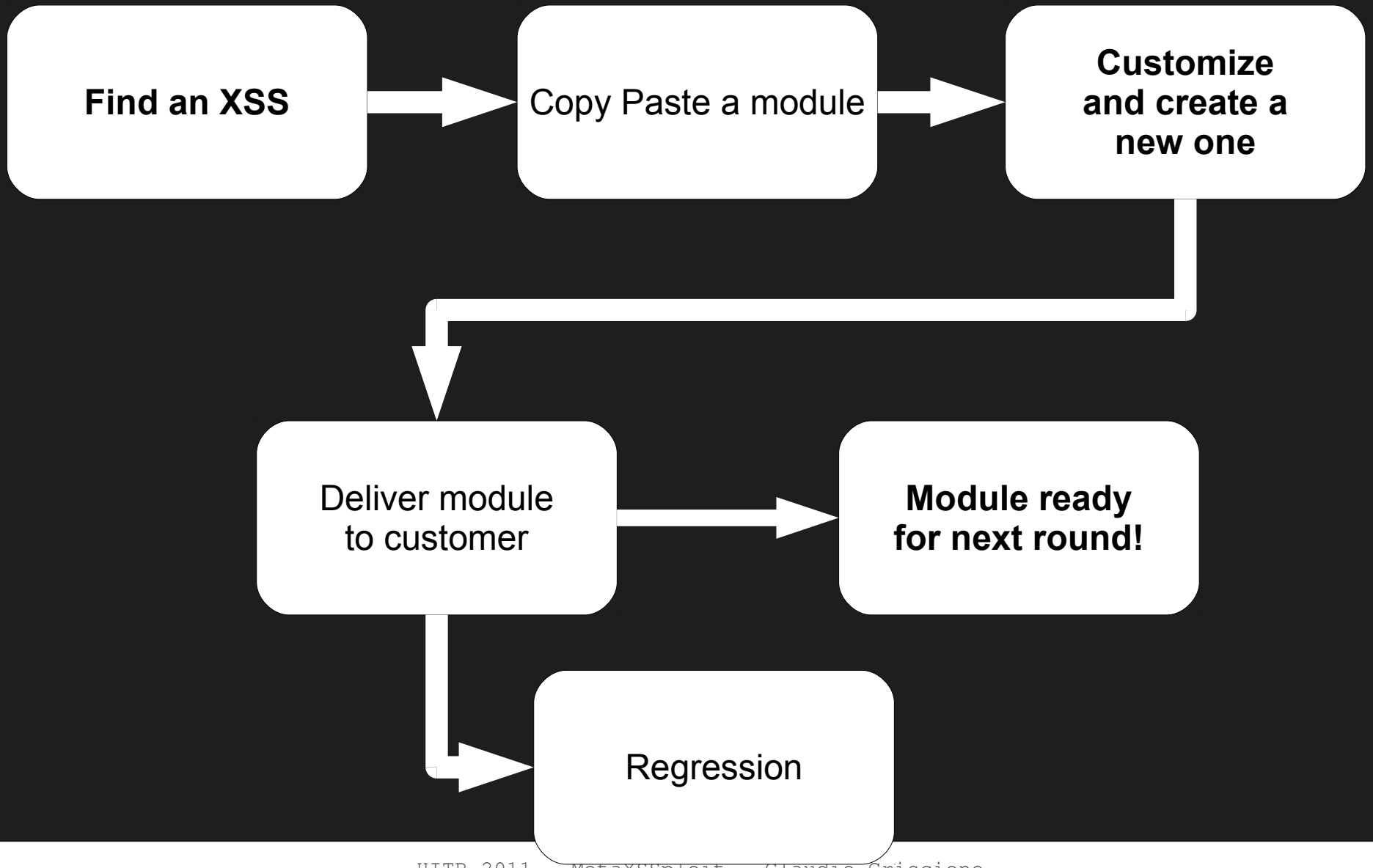
What about fingerprinting?

We already have some interesting project for that, like Blind Elephant

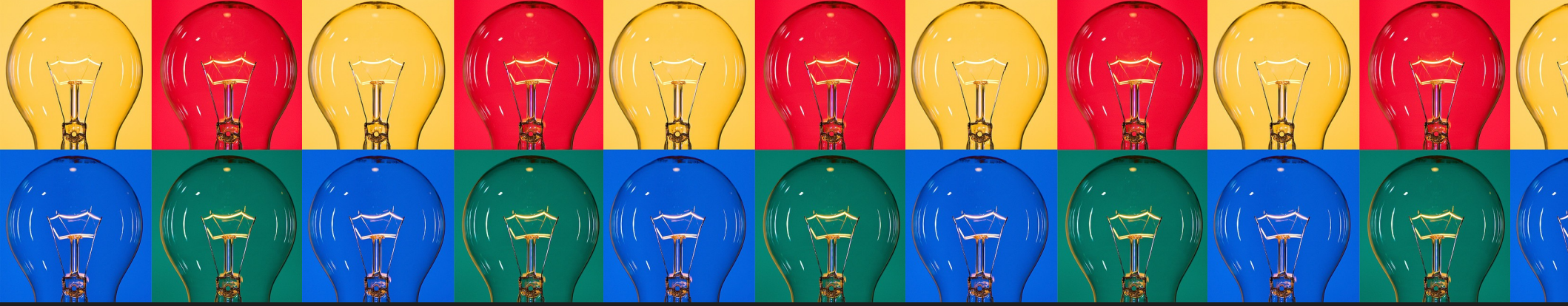
The general rule for tool building is

Solve only one problem

Use case – Custom App

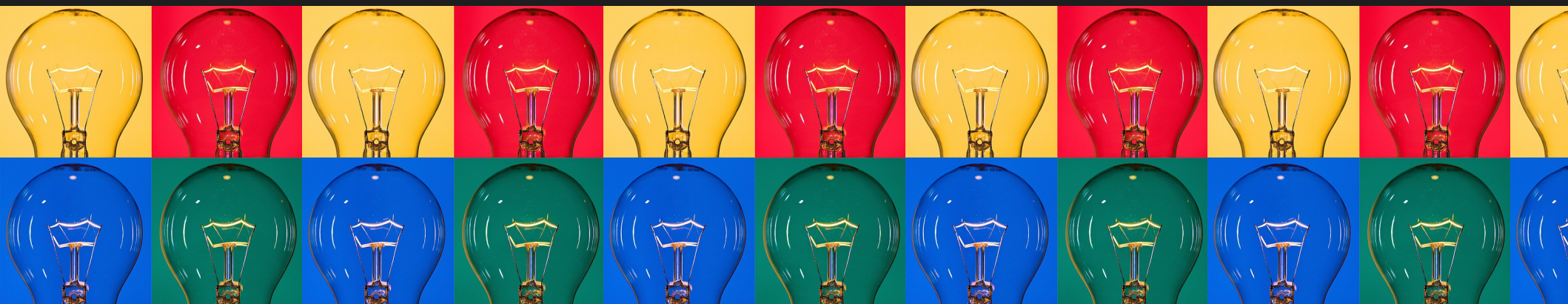



Adding some coolness



Idea 1

Leveraging those 100000000000 XSSes in Bugtraq so they at least serve a purpose



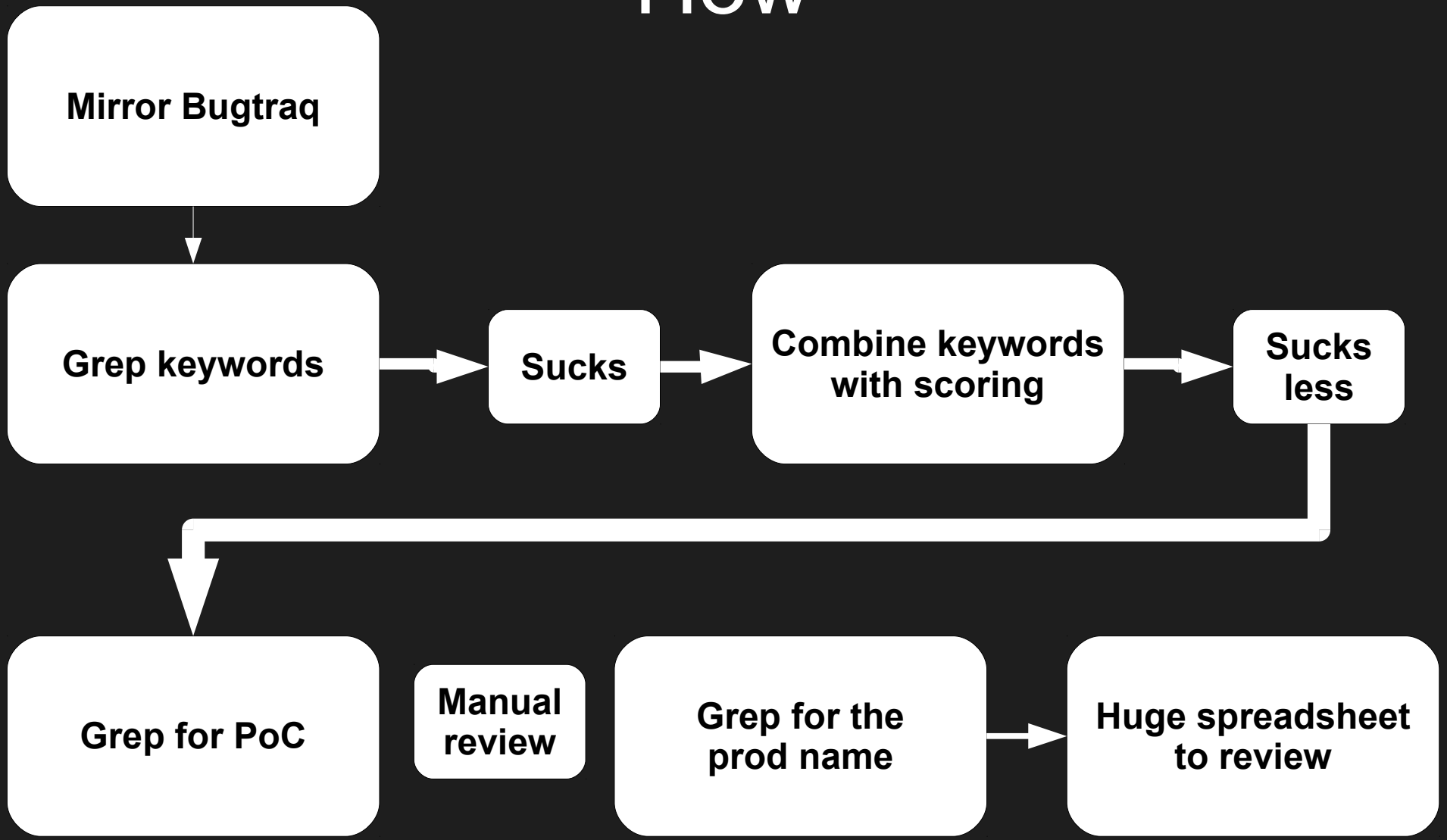
Automatic bugtraq to MetaXSSploit

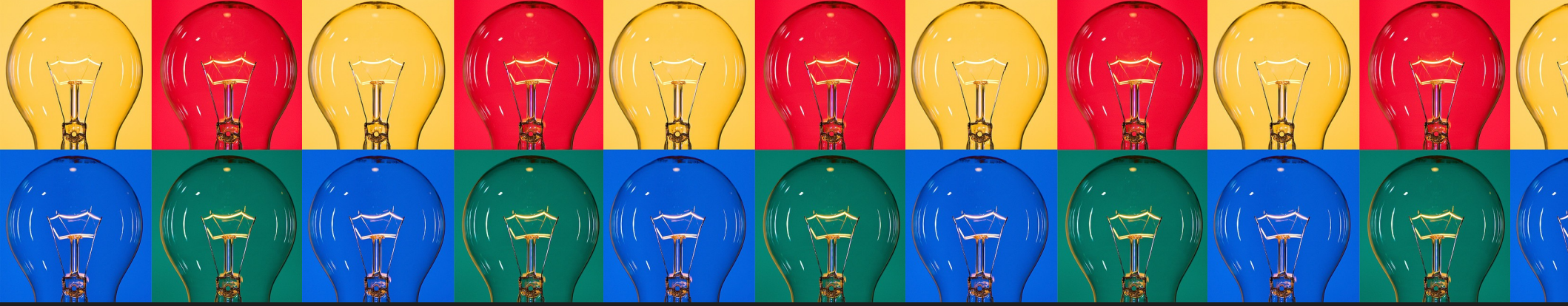
Providing XSS sample for exploitation is straightforward and most advisories will do it

Even if our method sucks badly, there are so many XSSes we are bound to have a huge library anyway

Some of these exploits end up buried and nowhere to be found after some months!

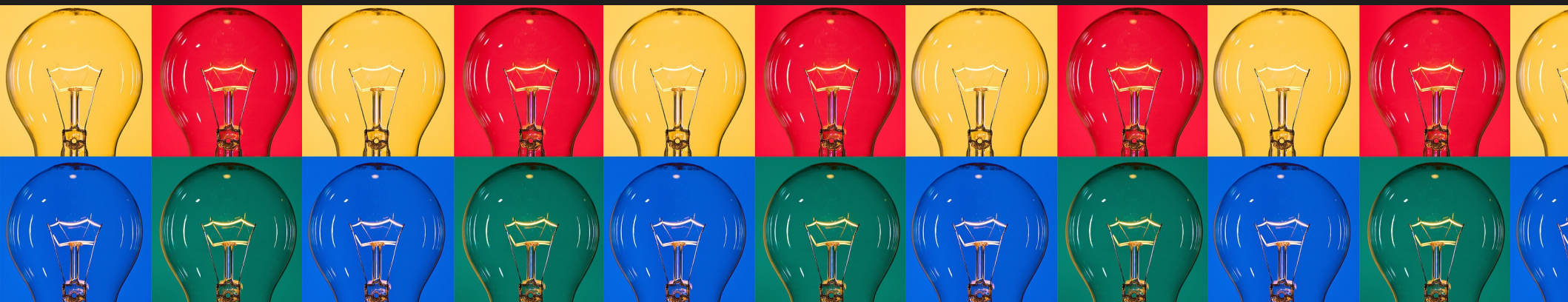
How


Mirror Bugtraq
Grep keywords
Sucks
Combine keywords with scoring
Sucks less
Grep for PoC
Manual review
Grep for the prod name
Huge spreadsheet to review



Idea 2

Automate the creation of exploits with a web interface



How do we do it?

- "Look, i found an xss"
- Go to the page
- Fill up the fields, give us the vector
- Download the resulting code
  - Request review on our side for inclusion

Looks interesting

Code is quickly reviewed and added to the MetaXSSploit database

Bragging factor for newbies

Easy to code

Did it work?

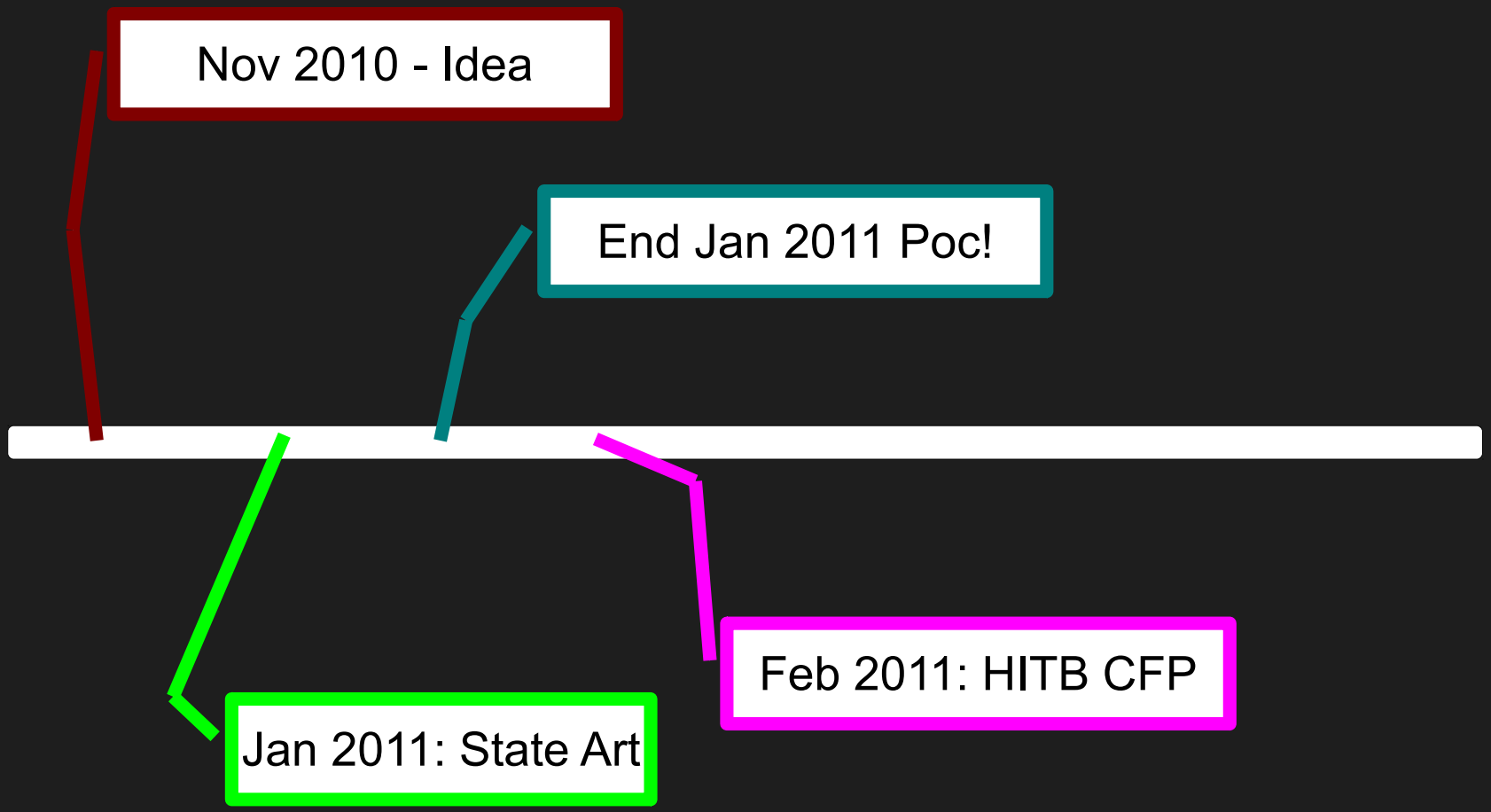
Yes, my POC is "more or less" working!

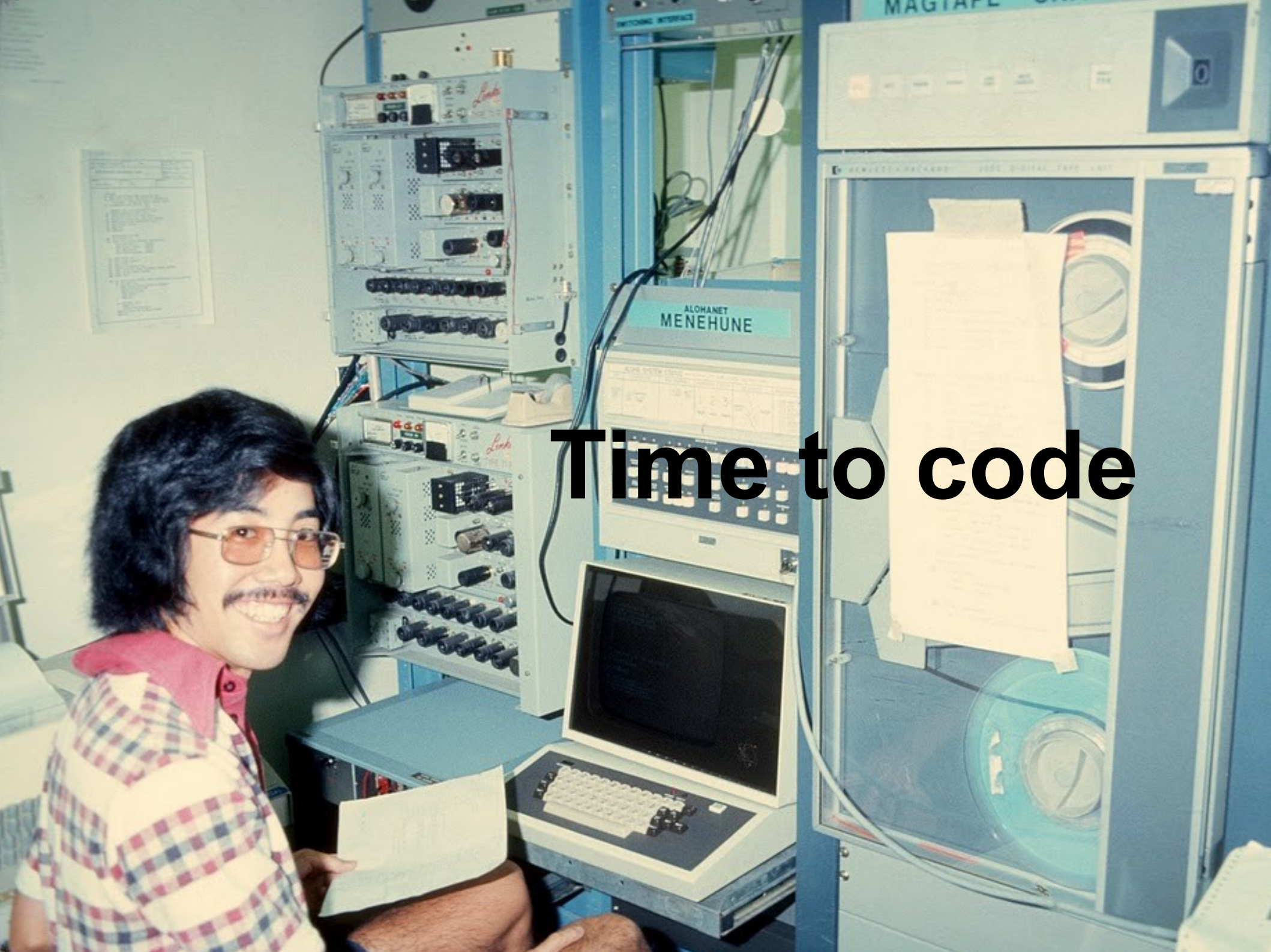
Cool!

So what do we do? Well, let's submit this one and wait for approval!

Approved? Ok now you'd better start coding!

Timeline


Nov 2010 - Idea
End Jan 2011 Poc!
Feb 2011: HITB CFP
Jan 2011: State Art


ALOHANET
MENEHUNE
Time to code

Why Metasploit?

Well known, so it cuts time to learn

Big framework, so it cuts development time

Big name, instead of the 100000th tiny tool

But wait!

Andres Riancho : W3AF is now sponsored by Rapid7, think about it as the Metasploit of webapps. We are even going to have payloads for web applications!

Joshua J Drake: Yes, there is space for Improvements in Metasploit XSS support*

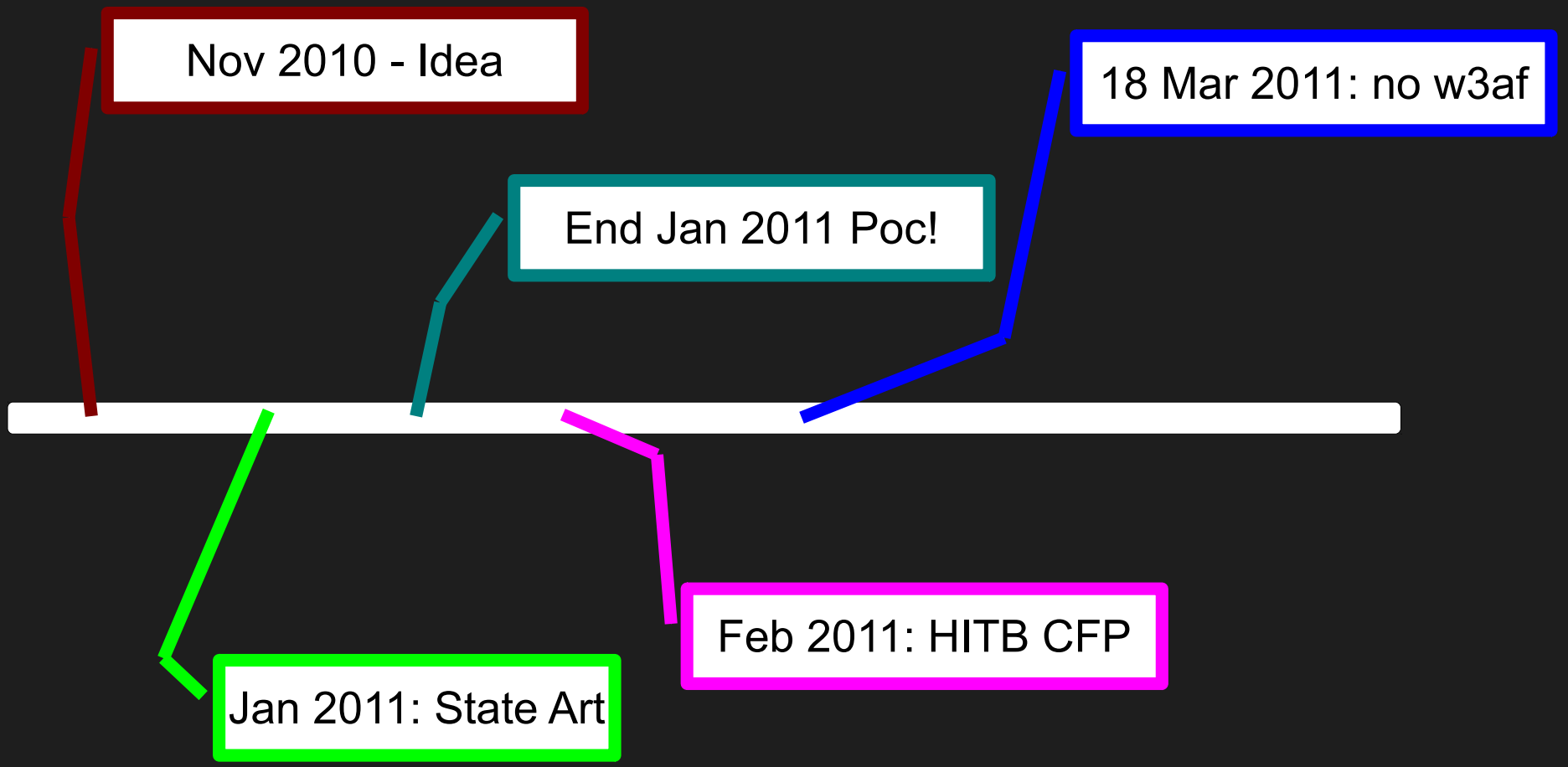
Why not w3af?

It is meant to be used against server-side targets, not client side

Let's speak with the guy in charge and see if we can integrate!

no.

Timeline


Nov 2010 - Idea
18 Mar 2011: no w3af
End Jan 2011 Poc!
Feb 2011: HITB CFP
Jan 2011: State Art

Reusing components

Metasploit has some interesting things in the Web / HTTP department, maybe?

```
blackfire@Thor:~/Tools/Metasploit$ find ./ -type f  | xargs grep 'Exploit::Remote::HttpServer' | grep -v svn | grep -v http | grep -v browser
./modules/exploits/osx/armle/safari_libtiff.rb: include Msf::Exploit::Remote::HttpServer::HTML
./modules/exploits/unix/webapp/mambo_cache_lite.rb:     include Msf::Exploit::Remote::HttpServer::PHPInclude
./modules/exploits/unix/webapp/php_include.rb:  include Msf::Exploit::Remote::HttpServer::PHPInclude
./modules/exploits/unix/webapp/base_qry_common.rb:      include Msf::Exploit::Remote::HttpServer::PHPInclude
./modules/exploits/unix/webapp/google_proxystylesheet_exec.rb:  include Msf::Exploit::Remote::HttpServer
./modules/exploits/windows/email/ms10_045_outlook_ref_resolve.rb:       include Msf::Exploit::Remote::HttpServer::HTML
./modules/exploits/windows/email/ms10_045_outlook_ref_only.rb:  include Msf::Exploit::Remote::HttpServer::HTML
./modules/exploits/windows/misc/realtek_playlist.rb:    include Msf::Exploit::Remote::HttpServer::HTML
./modules/auxiliary/server/file_autopwn.rb:     include Msf::Exploit::Remote::HttpServer::HTML
./modules/auxiliary/gather/android_htmlfileprovider.rb: include Msf::Exploit::Remote::HttpServer::HTML
```

Looks interesting, but...

Webapp attack support

In metasploit is **less than rudimentary.**

The tool has not been thought with XSS in mind.

Time to expand it, how do we do that?

What do we need

Deliver	Get the victim to execute the attack
Payloads	Do “something“ - the current payloads are simply not suited for us.
Vector Modules	Specifics for the actual exploit
Obfuscator Adaptor	Adapt the exploit to various browsers and make it harder to detect

Delivering the exploit

To deliver our exploit we can leverage Metasploit HTML server feature

Built to exploit browsers, of course!

For this reason, let's create a new class of exploits, XSSEXPLOIT

```
module Exploit::Remote::HttpServer::XSSExploit
include Msf::Exploit::Remote::HttpServer::HTML
```

XSSEXPLOIT

- XSSExploit provides the basic functionalities
  - Internal HTTP Server
  - Static content serving (for fake / phishing pages)
  - Invocation of obfuscators and encryptors
  - Helper methods for exploit check

  All MetaXSSploit exploits include this module

Basic form of delivery
User hitting our web server

A new kind of payload

- We need to think about our new payload: how are we going to deliver?
- (most) standard metasploit payloads = execute something
- We have a number of different use cases
  - Redirects, POSTs, JS inclusions...

XSS Payload

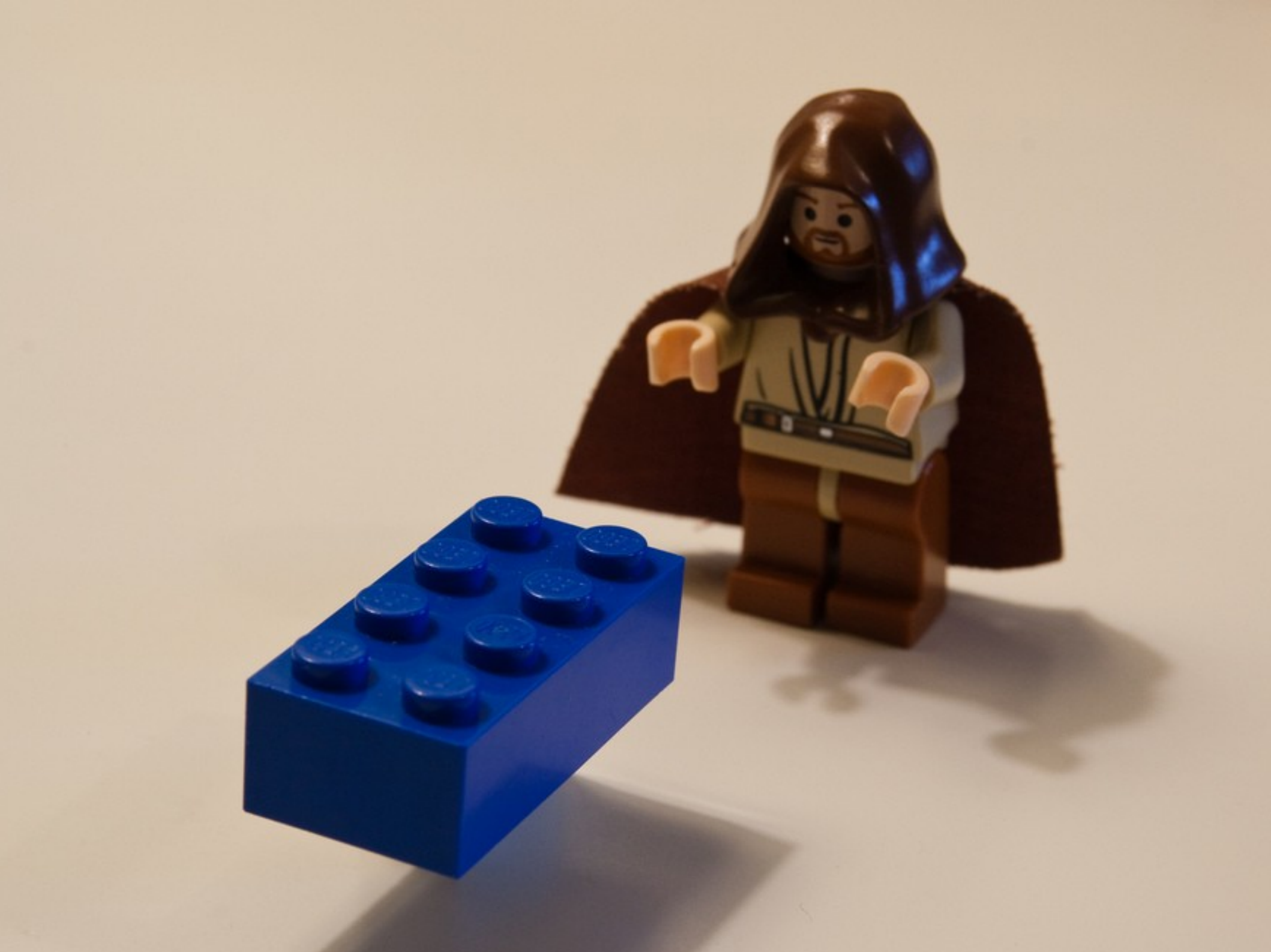
- **Always** pure javascript code
- The shorter the payload, and the smaller the charset the better
- Can be more or less everything
  - Easy integration with all standard XSS-exploiting tools
- Want to integrate a new tool? Just create an XSS payload!

The modules : XSSExploits

- Have to override the *vector_encapsulate* method
- We want them to produce a complete string pointing to the target by leveraging the payload
- Sample:

```
def vector_encapsulate(payload)
weaponized =
"http://#{datastore['RHOST']}:#{datastore['RPORT']}#{datastore['BASEPATH'
]}mkportal/modules/rss/handler_image.php?i=<script>#{payload}</script>"
return weaponized
```

Let us assemble everything like LEGO (tm)



Putting all together

- In order to actually deliver the exploit, the encapsulated_payload is not enough
- We also need an header, a footer and a delivery vector
- Enter the **wrapper**


The wrapper

The wrapper

- No real match in the Metasploit architecture
  - Maybe encoders, but then we also need encoders for all the actual encoding :{
- Implemented as a *case* switch in the XSS Payload main class
- In the end the wrapper will encapsulate the payload and produce a full HTML page to be served, ready to hit!

Supporting POST

- Wrappers' behaviour has to mutate to accomodate posts
  - Payload encapsulation is different
- One of the handy speed-ups of MetaXSSploit
- Some of the wrappers will not work with POST

Demo Time
Wake up NOW() please!

Challenges and cool stuff [If we have time]

Avoiding module sprawl

- We generated 500 different exploits: that means 500 files?!?
- Surely not! Most of them are just a line of POC, so we created a generic exploit module able to read and process that line... and just store it!
- In the end, just a text file as a database

BadChars and Encoding

- Each XSS exploit has to specify the set of chars it cannot use
- Payloads can be encoded, to a certain extend, but each payload requires a given set of chars
- Maximum size also considered
  - Less is better

Browser Adapters

- Different browsers need slightly different payloads, and some payloads will not work with some browsers
- Browser Adapters can convert payloads
- TBD :)

Future development

Integrating email sending

- We generate a fake page and code the email-sending on the server side
- We keep the wrapper logic, but of course we visit the page instead of the attacker
- We need a different wrapper for the final payload delivery

More! More! More!

- **More** modules, **more** encoders, **more** obfuscators
- The web app makes it easy to create some of them
- It is easy to enstablish a standard, turning **all advisories** directly into exploit modules

Complex scenarios

- Most complex logic can be supported with specific modules, but not everthing is supported (e.g. Parallel execution of calls in the same exploit)
- Even if you have to custom code it, it's still MUCH clearer than any description and you can test for it again in the future quickly

Summing Up

- We have built XSS support into Metasploit
- You can use it to
  - Leverage an existing Knowledge Base
  - Speed up actualy exploitation of XSSes
  - Build regression tests for web app
- Modules can be generated automatically via a webapp or by script
- Not avail NOW(), will be shortly (review process permettendo)

Questions

Thank you!

This was

MetaXSSploit

Bringing XSS in Pentesting

And I am : Claudio Criscione - @paradoxengine

claudio.criscione DALLE_PARTI_DI gmail.com

Tnx for the pictures!

http://www.flickr.com/photos/alkelda/4950272224/
http://www.flickr.com/photos/john/42644763/sizes/o/
http://www.flickr.com/photos/danielray/4803043617/
http://www.flickr.com/photos/zetson/3036254720/
http://www.flickr.com/photos/drift-words
Roger Bissonnette
http://www.flickr.com/photos/bri-bri