Browser Exploitation Framework

BeEF

www.beefproject.com

WTFlip is Browser Hacking?

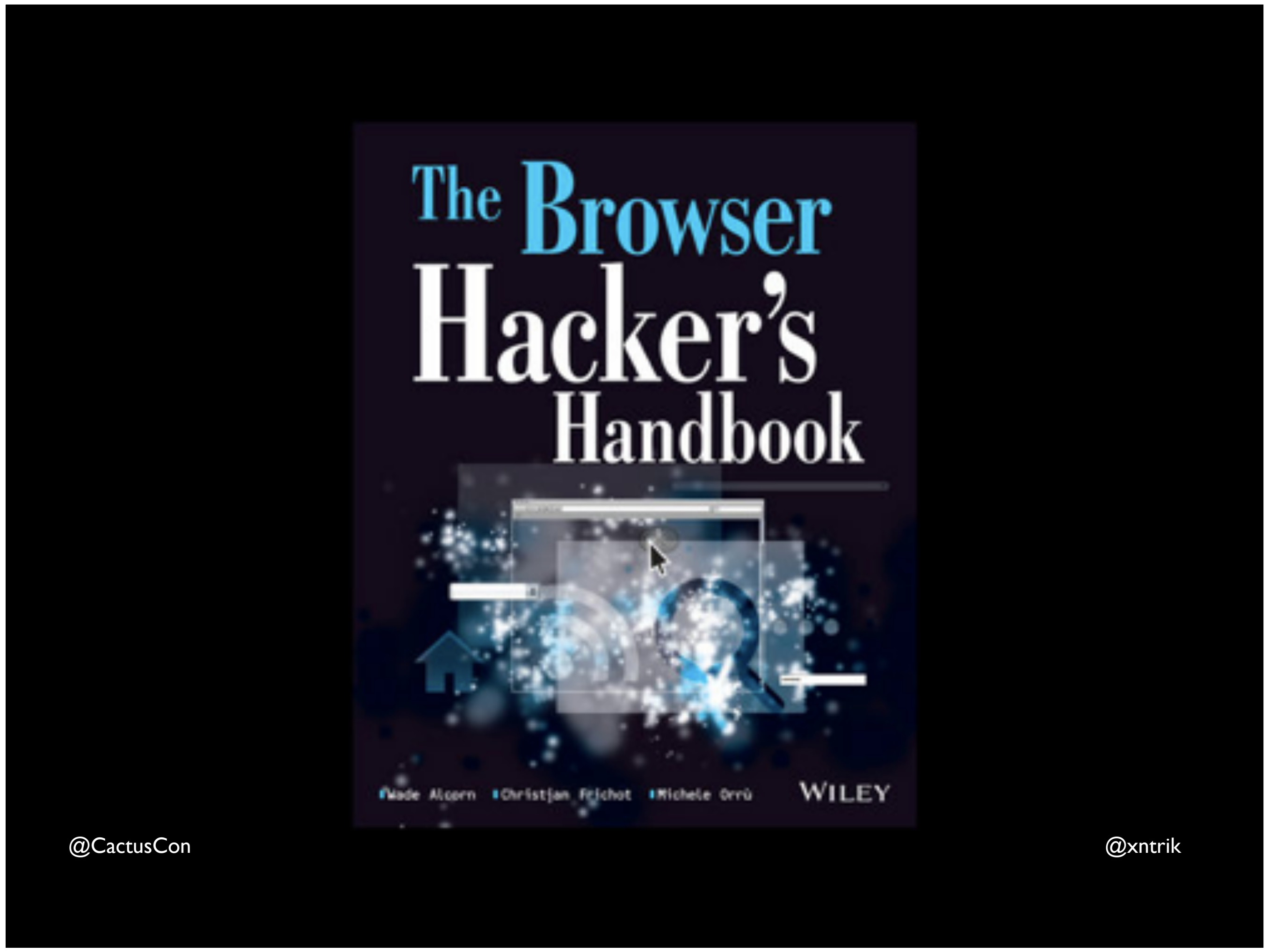
..and the importance of client-side testing

An intimate look at JavaScript with Christian Frichot

Who

- App Sec Nerd @ LinkedIn
- BeEF Developer
- Security Fun Guy
- Drummer
- Nunchuck skills

The opinions of this presentation are my own and don’t reflect my employer.



And a co-author of the Browser Hacker's Handbook. Released in 2014, a number of the concepts discussed here are from our book.

The Browser Hacker's Handbook focuses on various attacks that can occur once an attacker has control of the DOM, or other contexts within the browser, such as plugins.

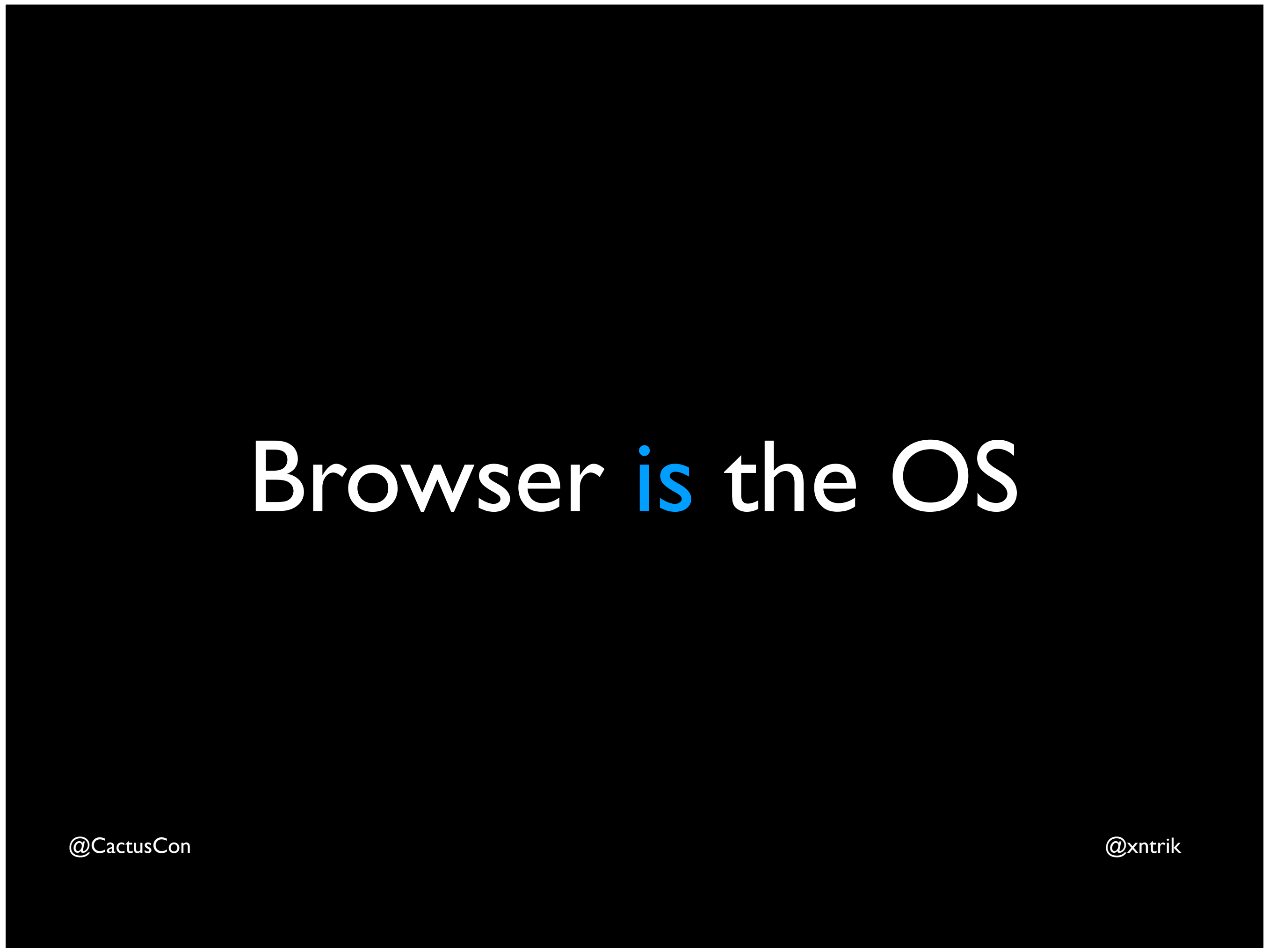
You can grab the book from Amazon here: http://a.co/jbruw7E

@antisnatchor
(hates pants)
@wadealcorn
(likes pants)

The other co-authors ;)

What

This talk is mainly about Browsers, JavaScript and the Browser Exploitation Framework. Focused on the following themes:

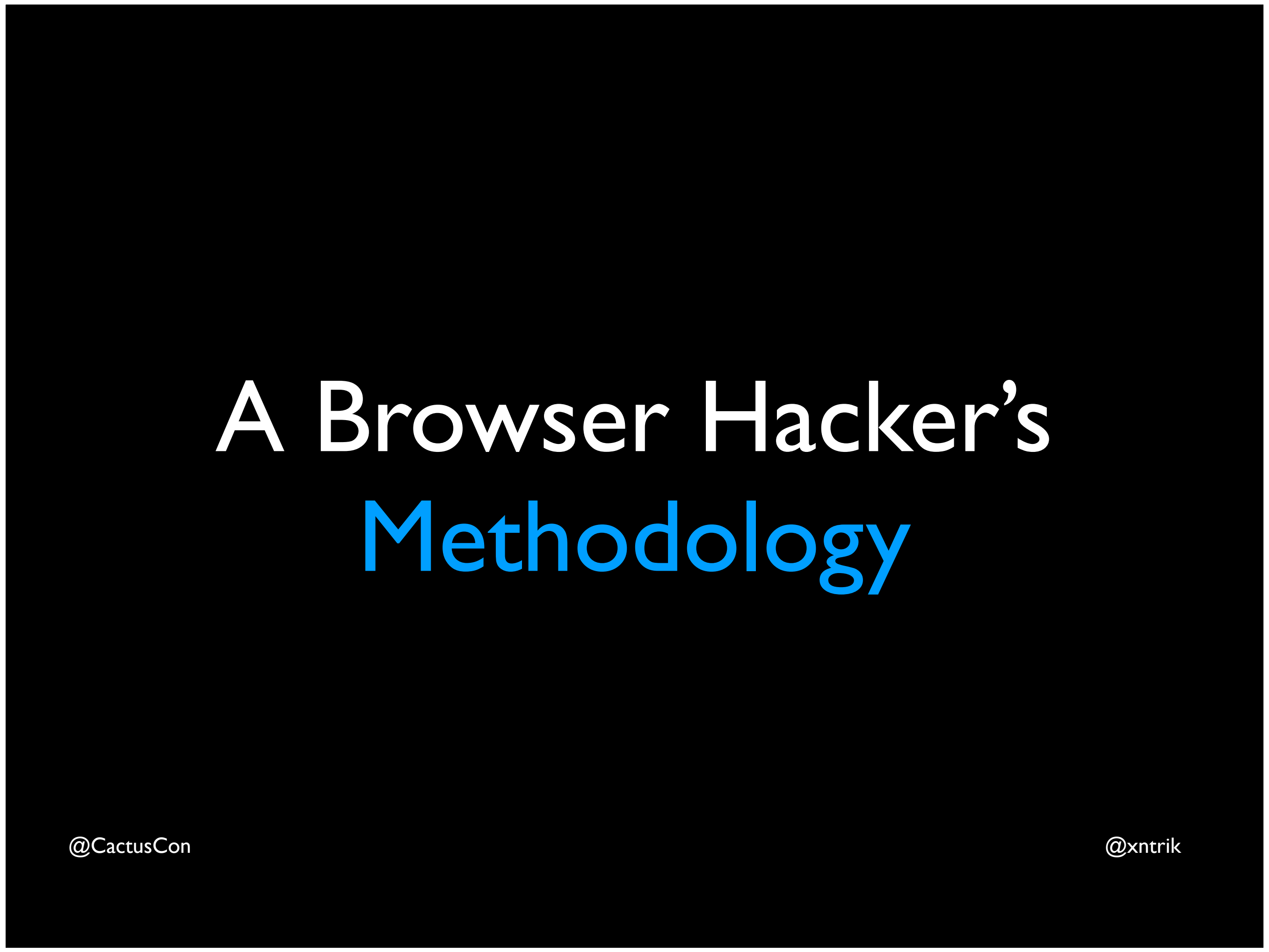


Browser is the OS

@CactusCon

@xntrik

especially of our current generations.



A Browser Hacker's Methodology

@CactusCon

@xntrik

A browser-based client-side security testing methodology

Browser Hacks

@CactusCon

@xntrik

A summary of attacks, and just how bad it can get if malicious logic gets inside your browsers.. I'm really keen to move beyond an alert popup box if you've discovered an XSS - if you ever need to demonstrate how bad client-side injection is, you should definitely be looking at tools like BeEF.

How we live

The Internet, and browser's are EVERYWHERE. You've got n+1 on your phone, tablet and computer. Let alone your work computers etc etc. Each browser establishes a context with each and every site, offering effectively infinite combinations of interactions.

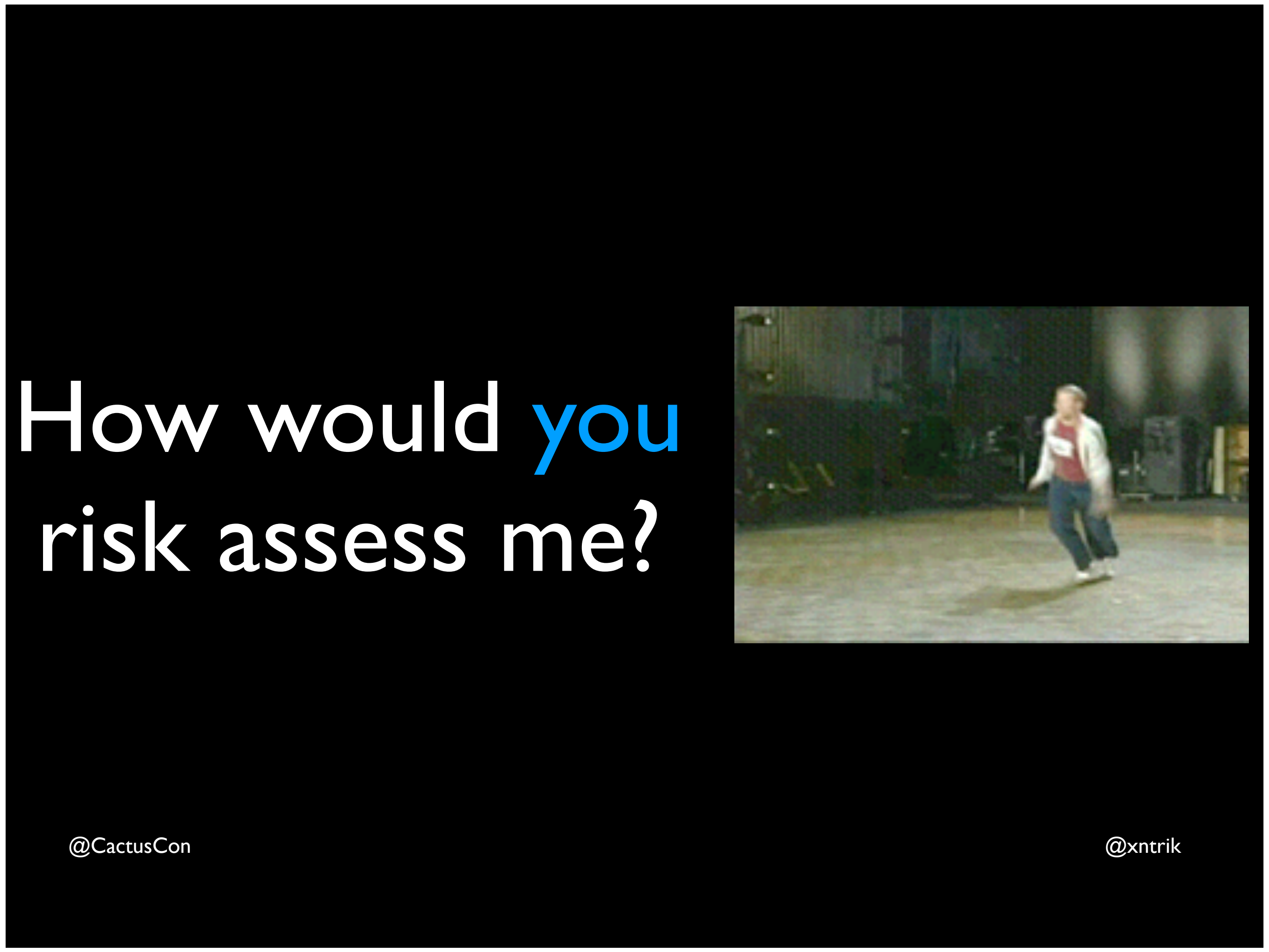
HTML, JavaScript and other browser-tech is deeply embedded in how we live, and is NOT disappearing.

Subtle Complexity

@CactusCon

@xntrik

Take Chrome for example. Looks very simple, but is exceptionally complex.

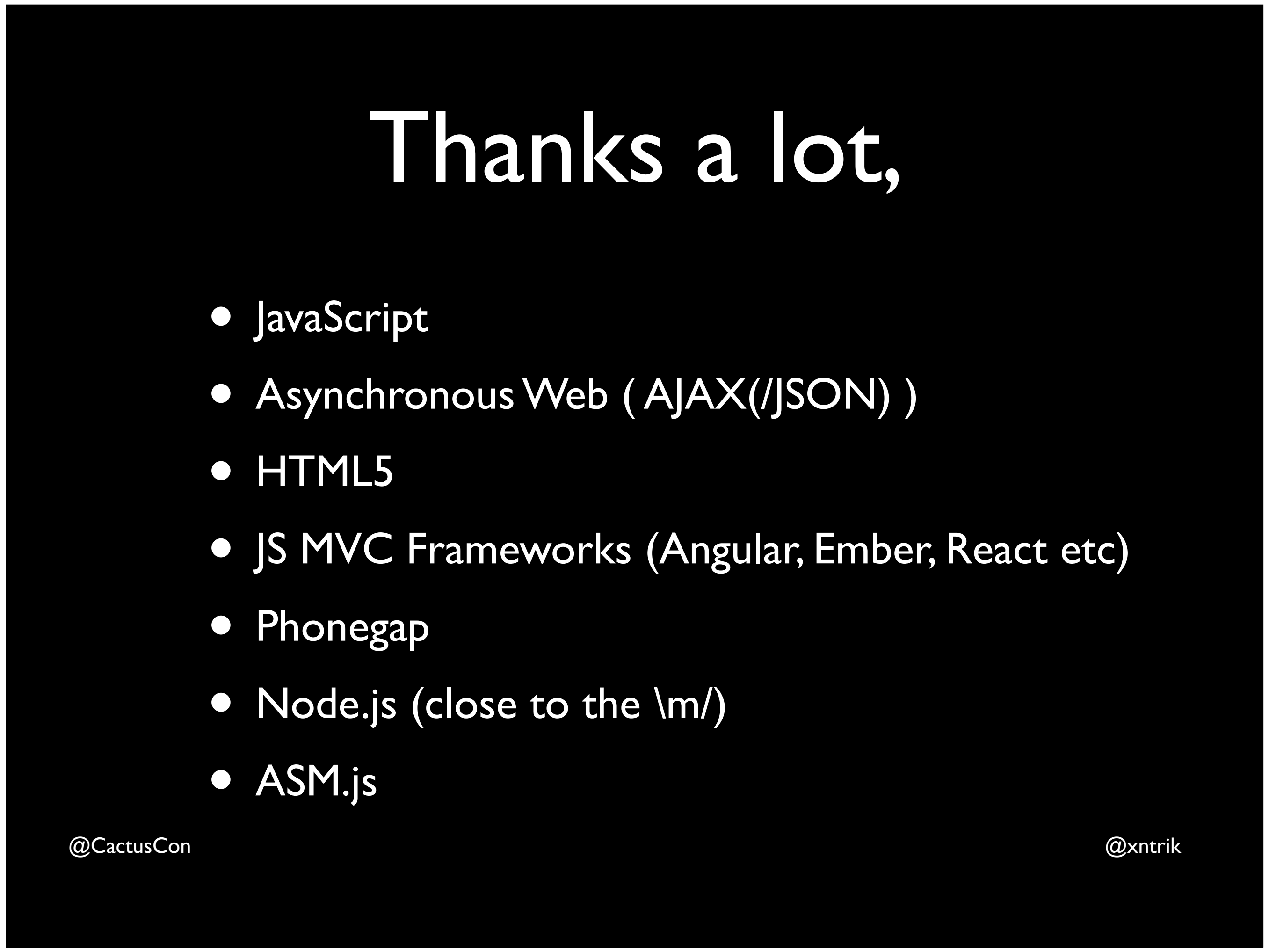


How would you risk assess me?

@CactusCon @xntrik

If you tried to perform a traditional information risk assessment on a 'web browser', if it was something you hadn't seen on your network before, you'd likely be concerned. It can access the Internet, while simultaneously accessing your Intranet, and is the primary platform to access everything.

Thanks a lot,



Thanks a lot,

- JavaScript
- Asynchronous Web (AJAX(/JSON))
- HTML5
- JS MVC Frameworks (Angular, Ember, React etc)
- Phonegap
- Node.js (close to the \m/)
- ASM.js

@CactusCon @xntrik

https://www.destroyallsoftware.com/talks/the-birth-and-death-of-javascript
Unreal engine in FF 250KLOC of C -> Compiled to asm.js -> Running in browser.

End of life...

@CactusCon

@xntrik

But not all browser technology is destined to work for ever, particularly those that give lower-level permissions. For example..

End of life...

- Flash
- Silverlight
- Java Applets

@CactusCon @xntrik

Non-native..
FF, native java, then applets, then self-signed, then formally signed, then CtP

Instead..

HTML5 APIs

WebSocket Protocol

WebWorkers

@CactusCon

@xntrik

WebRTC

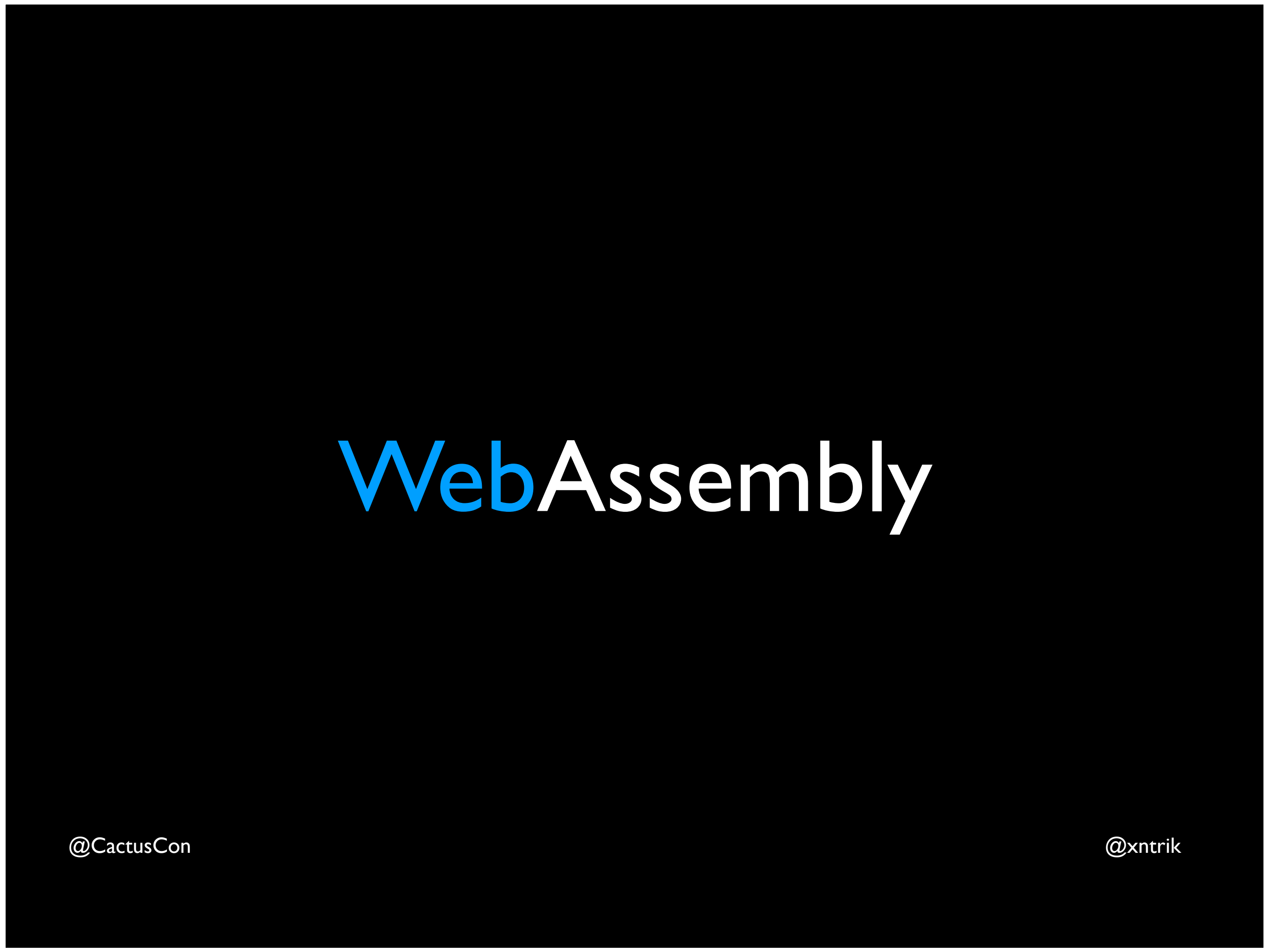
NaCl

@CactusCon

@xntrik

Google’s Native Client - to overcome the slowness of JavaScript.

Portable NaCl, or PNaCl (pronounced pinnacle) allows developers to compile into bitcode, which is then translated to host-specific executable when it’s run in Chrome.



WebAssembly

@CactusCon

@xntrik

https://medium.com/javascript-scene/what-is-webassembly-the-dawn-of-a-new-era-61256ec5a8f6#.xlr0wafv0

https://github.com/WebAssembly/design/blob/master/HighLevelGoals.md

Browser IS the OS

@CactusCon

@xntrik

Client-side testing?

@CactusCon

@xntrik

What is client-side testing? As opposed to traditional vulnerability assessments or penetration tests, client-side tests focus on your endpoints.

Perimeters getting stronger

@CactusCon

@xntrik

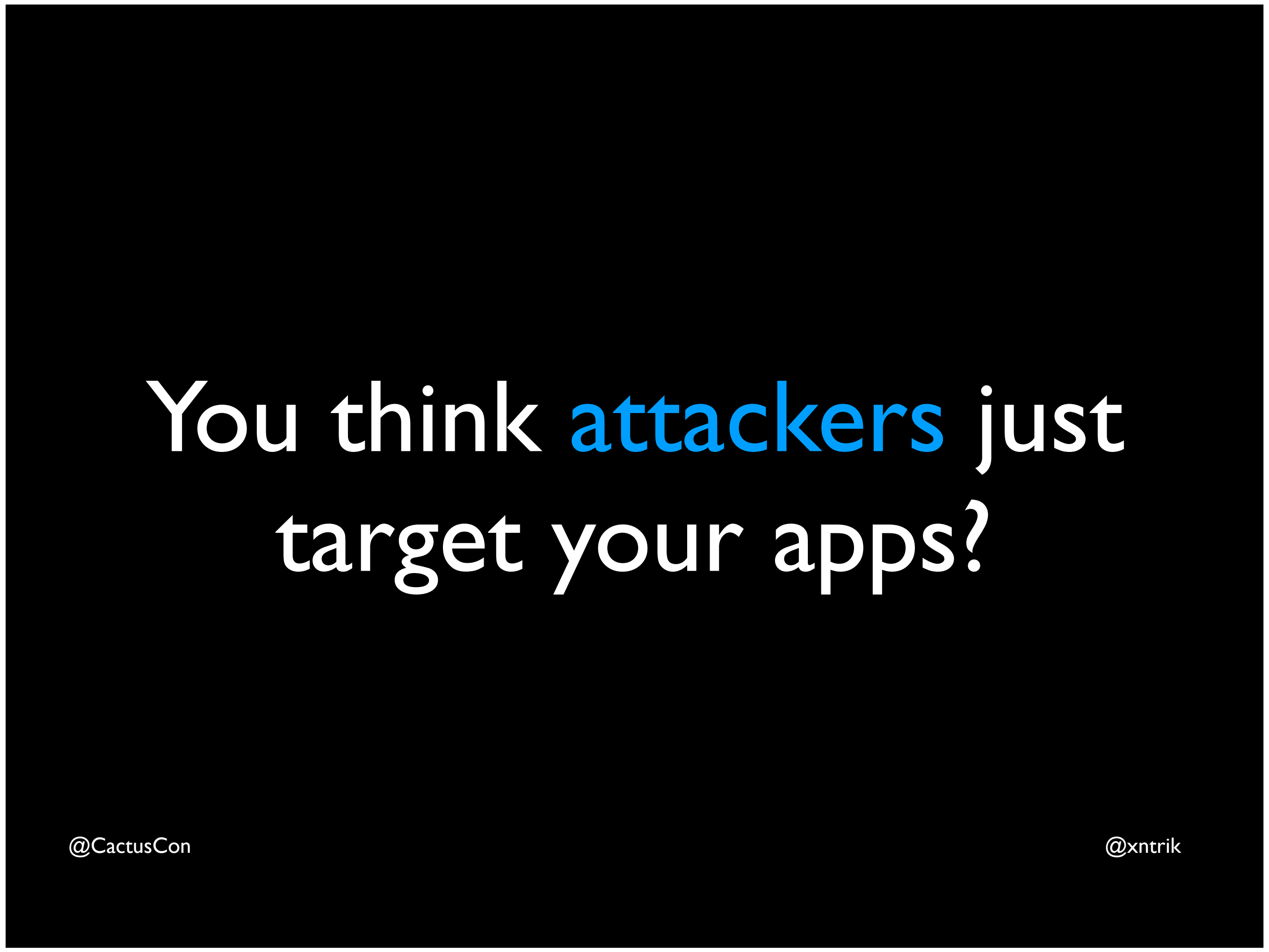
Edge controls are getting better (?). Web apps are in some instances getting more resilient. This may be to do with the constant barrage of attacks against web apps as soon as they’re online. Modern web dev frameworks are also better at providing secure-by-default options.

Your precious cheese isn't always exposed to the Internet

@CactusCon

@xntrik

Enterprises and corporates, regardless of the grown of SaaS and cloud offerings, are still often running their sensitive systems on the intranet, or other internal systems.



You think attackers just target your apps?

@CactusCon

@xntrik

It’s also realistic, drive-by downloads, xss scripting, social engineering attacks - y

What about your people?

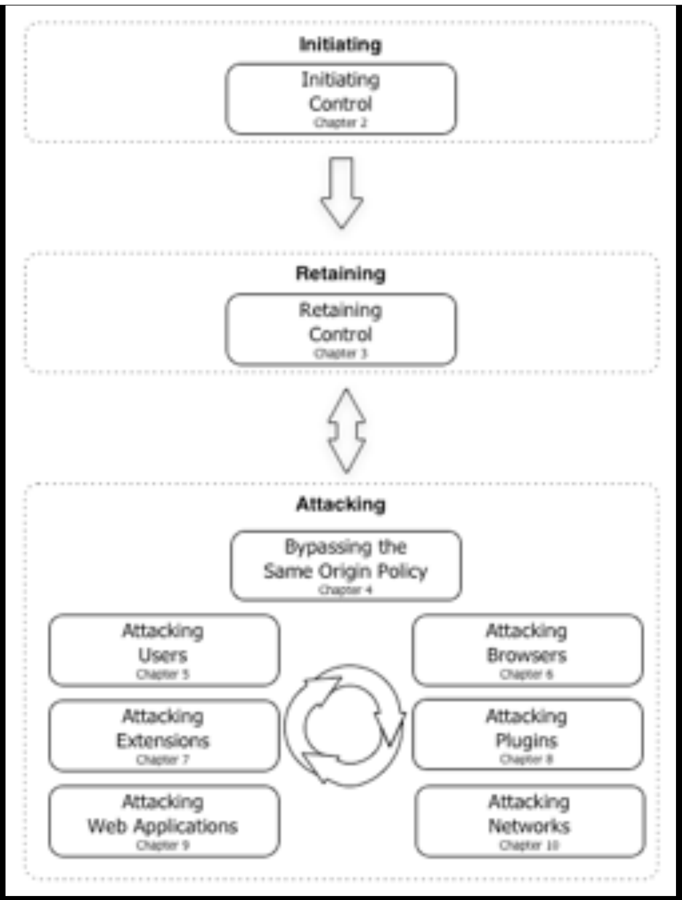
Benefits

- Expand and accurately measure the attack surface
- Properly consider the changing perimeter environment
- Realistic (You think real attackers aren't after your internal workstations and staff?)

Difficulties

- Not as well understood by testers
- Not as well understood by clients
- Discomfort related to Social Engineering related assessments

Effectiveness



@CactusCon @xntrik

In the Browser Hacker's Handbook we've broken down the chapters to follow a simple methodology for attacking browsers.

BeEF

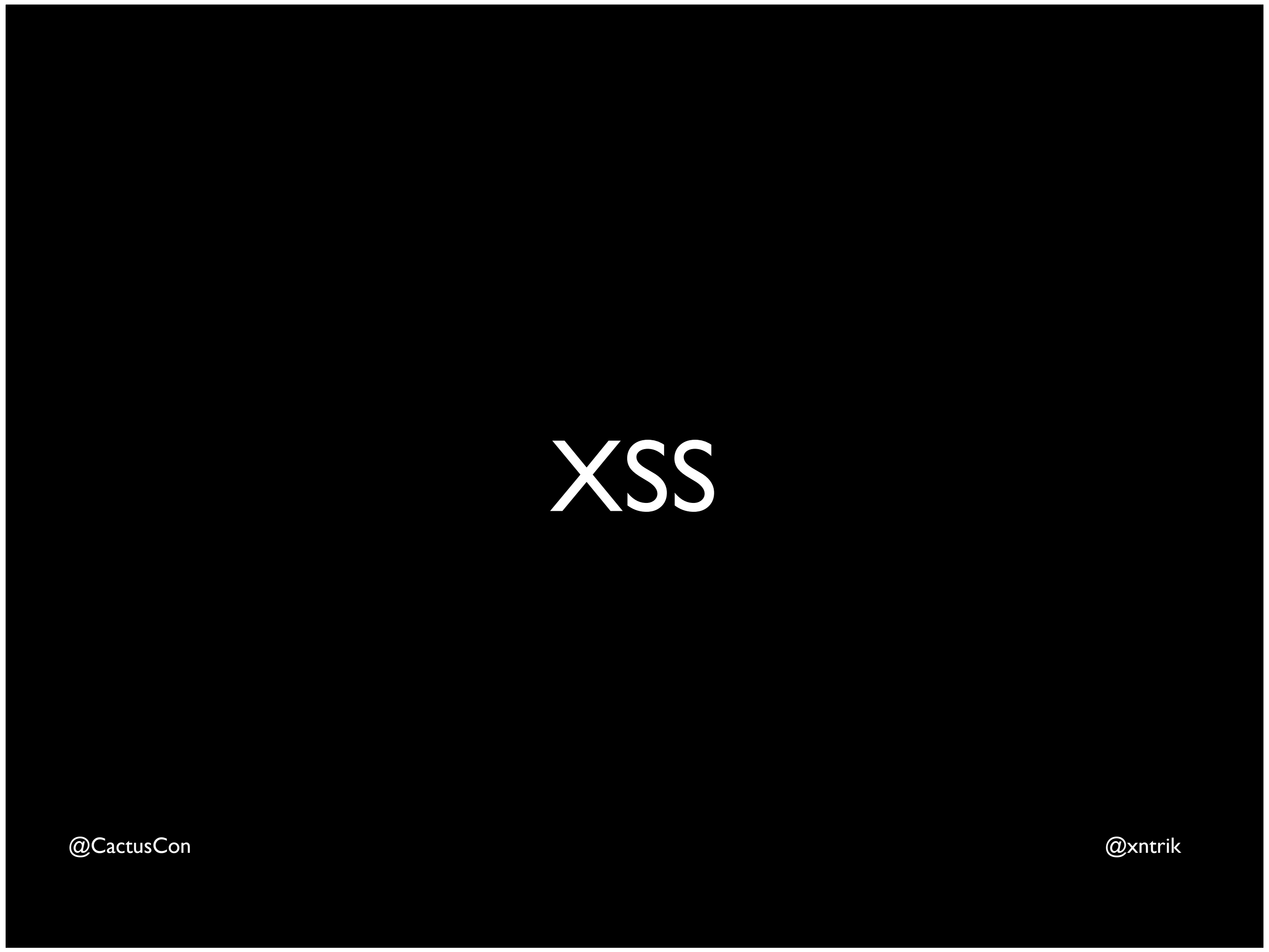
@CactusCon

@xntrik

So, in the context of browser attacking, I’m primarily talking about the Browser Exploitation Framework.

Phase 1
Hooking

Step 1 .. it's always hooking.. we use this term to combine the concepts of initiating and maintaining control.



XSS

@CactusCon

@xntrik

This is what BeEF was originally developed for.

Pwning Web Sites/Apps

@CactusCon

@xntrik

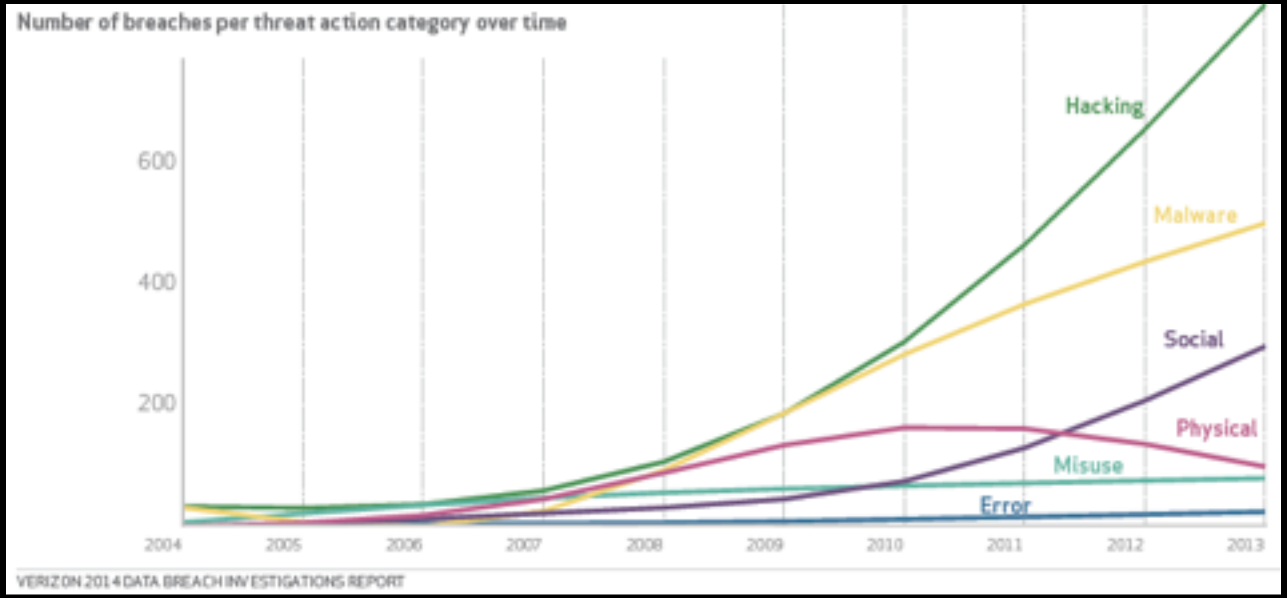
Malicious Ads

MitM

Social Engineering

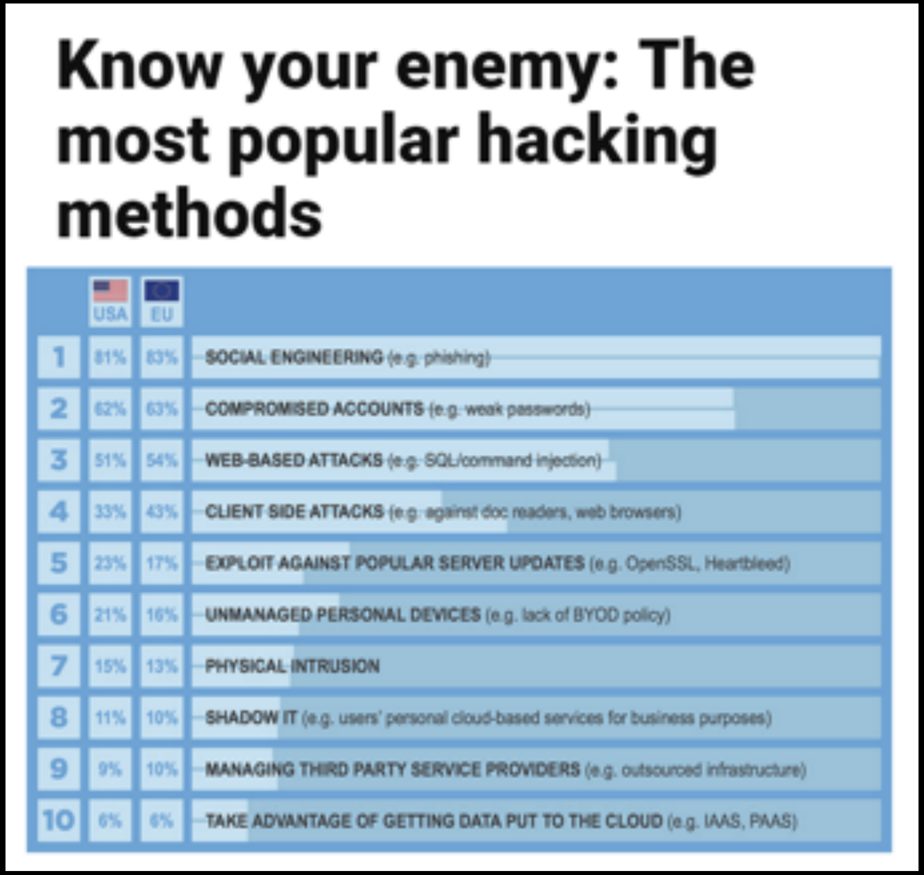
@CactusCon

@xntrik



@CactusCon @xntrik

The social engineering component, and leveraging browsers is one of my favourite aspects of these attacks, and not surprising, the recent Verizon Data Breach Investigation Report also highlighted the growth of the attack vector.



Know your enemy: The most popular hacking methods

	USA	EU	
1	81%	83%	**SOCIAL ENGINEERING** (e.g. phishing)
2	62%	63%	**COMPROMISED ACCOUNTS** (e.g. weak passwords)
3	51%	54%	**WEB-BASED ATTACKS** (e.g. SQL/command injection)
4	33%	43%	**CLIENT SIDE ATTACKS** (e.g. against doc readers, web browsers)
5	23%	17%	**EXPLOIT AGAINST POPULAR SERVER UPDATES** (e.g. OpenSSL, Heartbleed)
6	21%	16%	**UNMANAGED PERSONAL DEVICES** (e.g. lack of BYOD policy)
7	15%	13%	**PHYSICAL INTRUSION**
8	11%	10%	**SHADOW IT** (e.g. users' personal cloud-based services for business purposes)
9	9%	10%	**MANAGING THIRD PARTY SERVICE PROVIDERS** (e.g. outsourced infrastructure)
10	6%	6%	**TAKE ADVANTAGE OF GETTING DATA PUT TO THE CLOUD** (e.g. IAAS, PAAS)

Attribution: the internet?

Initiating Control ✓

Maintaining Control

@CactusCon

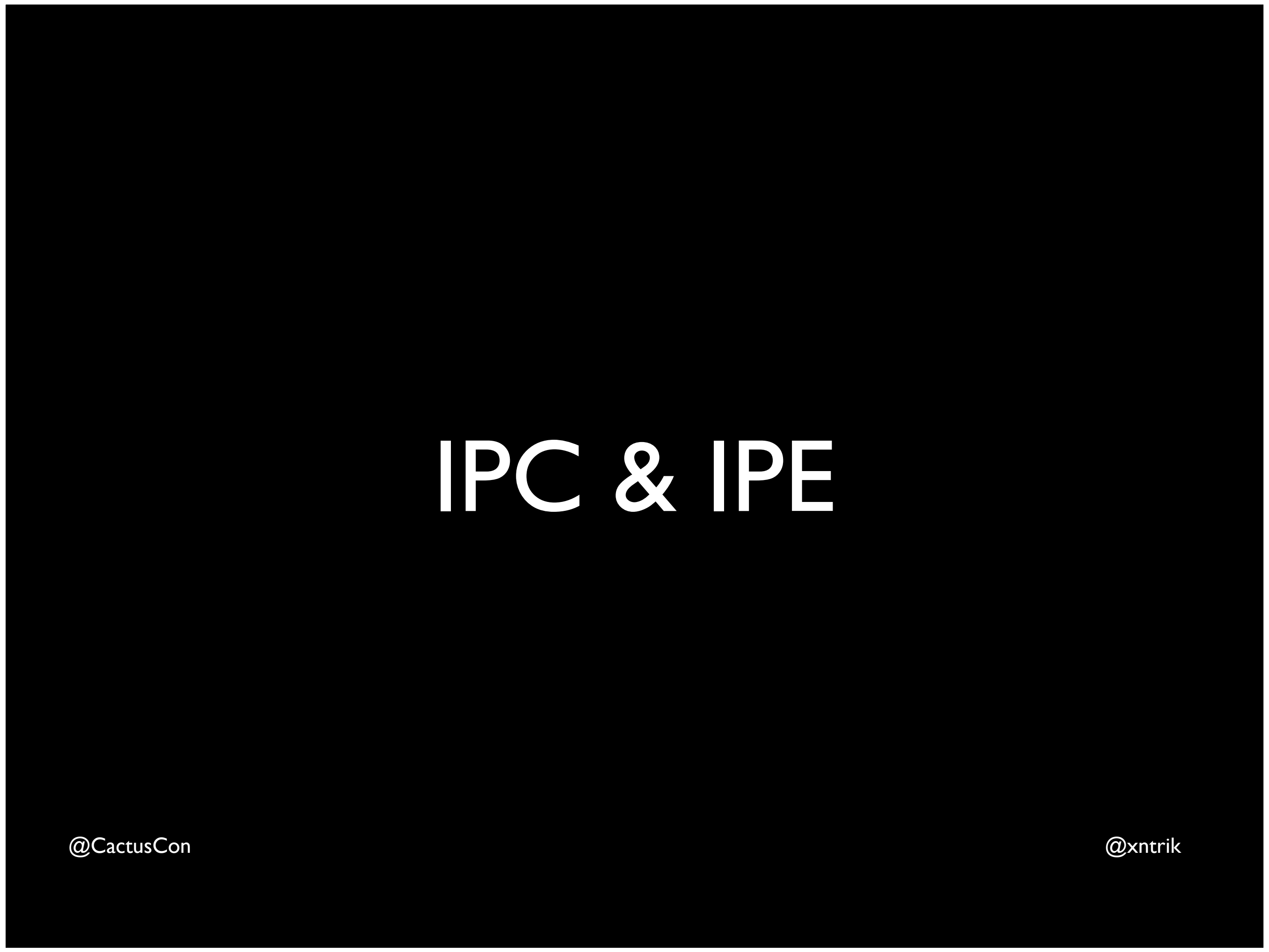
@xntrik

After the initial execution of your code, it helps to continue maintaining the channel…
Maintaining control includes the communications between a browser and your attacking server (think of a botnet, or command & control environment), and also persistence - how do you keep browser's under your control even in the face of user actions, such as clicking away?

Some attacks need time

Port Scanning

Fingerprinting



IPC & IPE

@CactusCon

@xntrik

Inter-protocol Communication and Exploitation

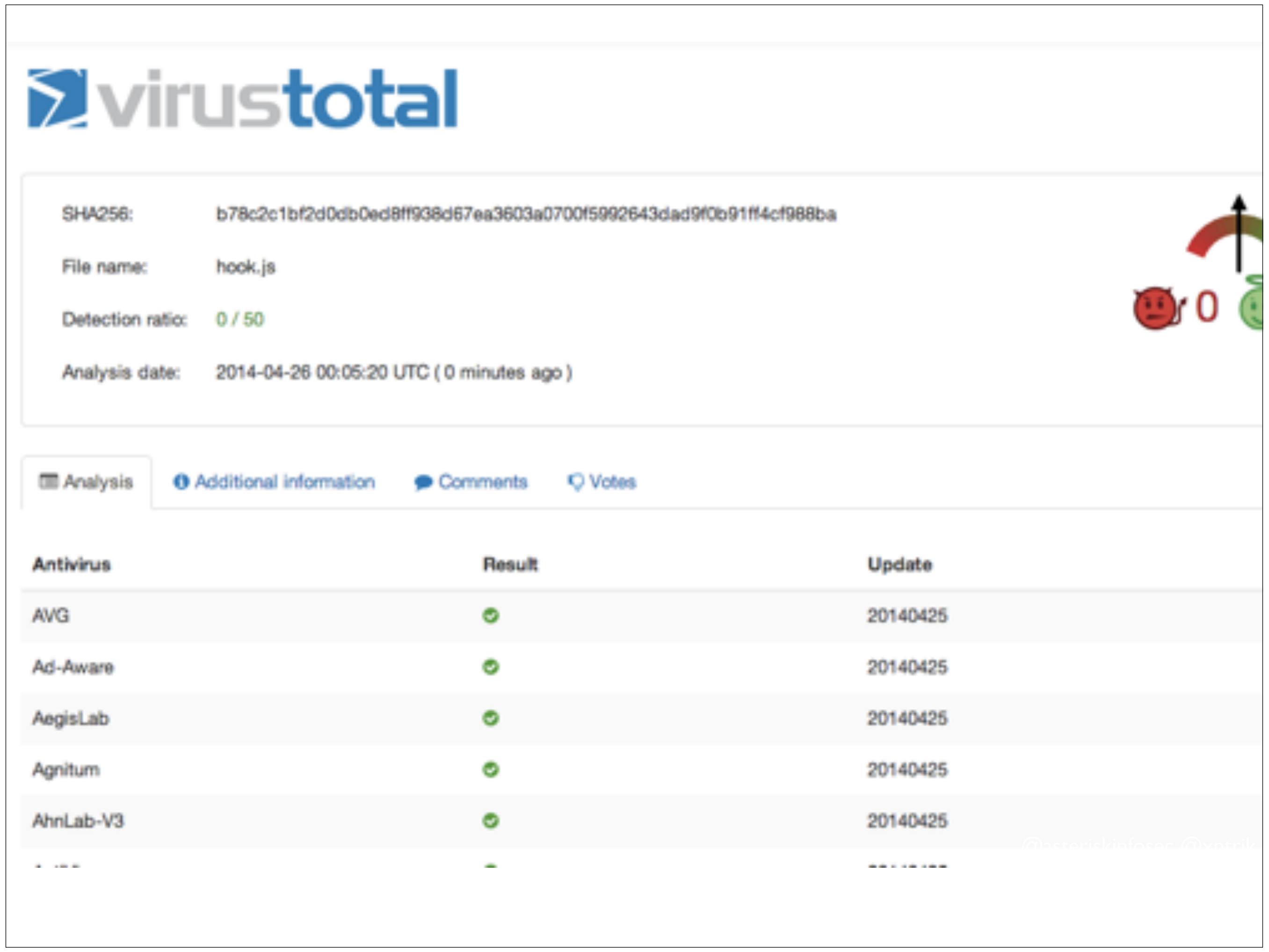
Phase 2 - Comms

- XMLHttpRequest
- WebSockets
- WebRTC
- DNS Tunnelling

Phase 3 - Persistence

- IFrames
- Handling browser close events
- MitB trickery

At this point your browser is hooked... and talking back to you. Now, if you're wondering how likely this is to get through your enterprise controls? Don't forget, all of these technologies, even to some degree WebSockets, are all native web traffic. This is NOT odd or malicious looking traffic.



SHA256:	b78c2c1bf2d0db0ed8ff938d67ea3603a0700f5992643dad9f0b91ff4cf988ba
File name:	hook.js
Detection ratio:	0 / 50
Analysis date:	2014-04-26 00:05:20 UTC (0 minutes ago)

Antivirus	Result	Update
AVG	✓	20140425
Ad-Aware	✓	20140425
AegisLab	✓	20140425
Agnitum	✓	20140425
AhnLab-V3	✓	20140425

This is what BeEF looks like

<magic>

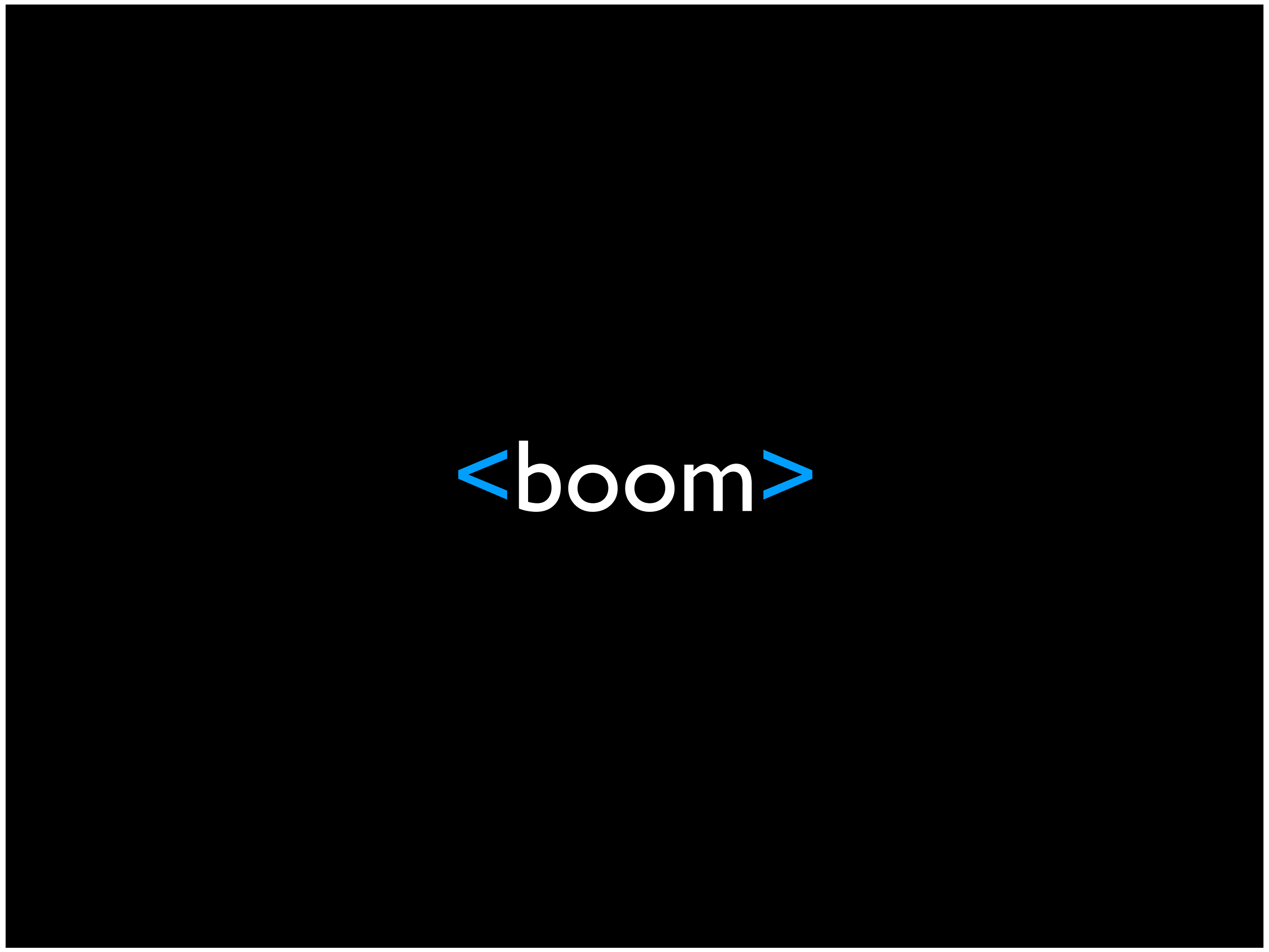
@CactusCon

@xntrik

https://youtu.be/1CXYYjzvIdM

Passive Attacks..

What about Active attacks?


<boom>

https://youtu.be/8D27fAS9HMk

Browser Hacking

Attack Classes

- Users
- Browsers
- Extensions
- Plugins
- Apps
- Networks (IPC/IPE)

Issues Highlighted

@CactusCon

@xntrik

Browsers have access through multiple channels

Browsers have access to many systems

The web, and modern web technologies, demand A LOT from browsers

So what can we do?

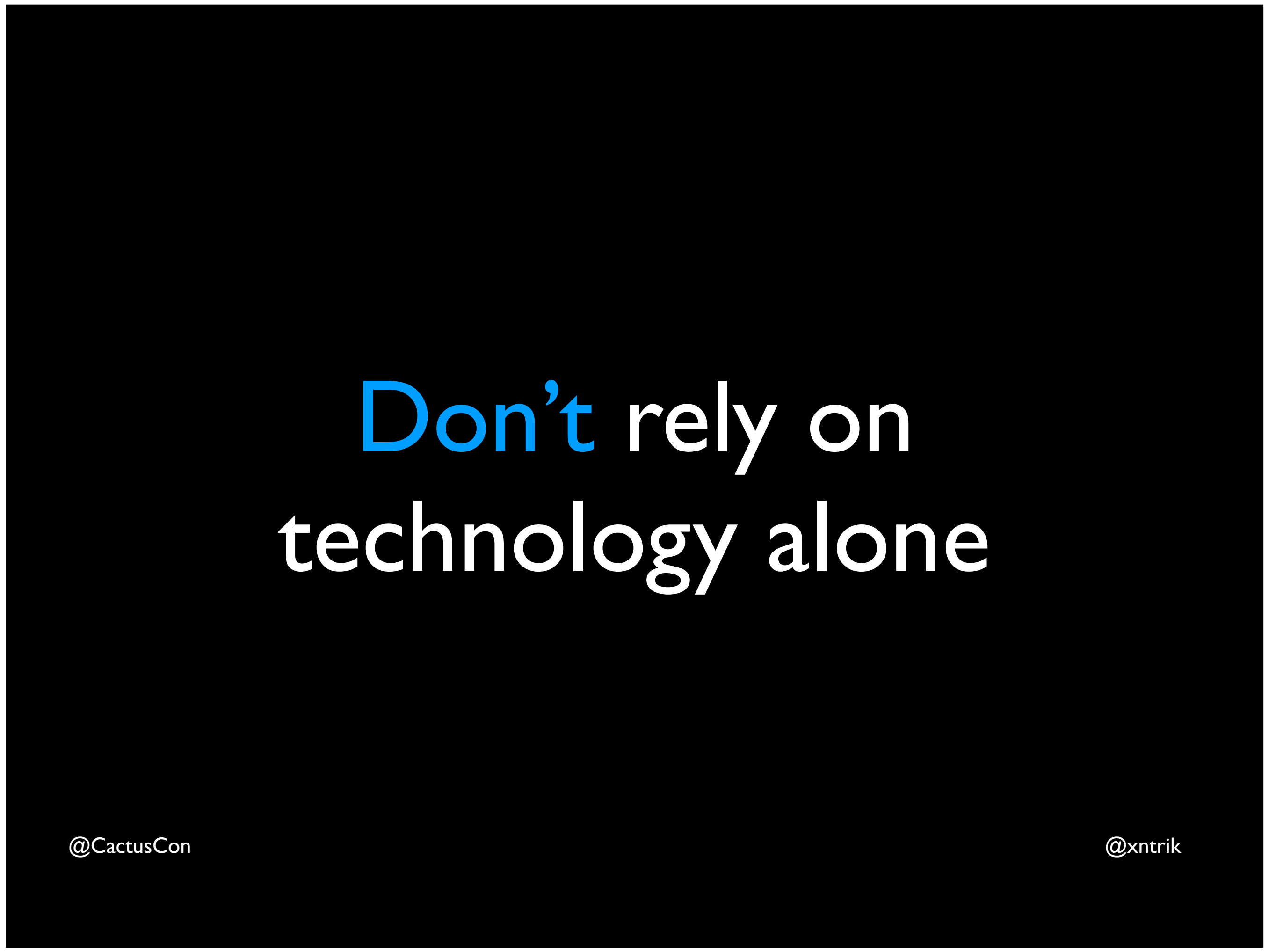
DRINK UP

Monitor

Bolster your IR

TEST your IR

Emulate these attack scenarios



Don’t rely on technology alone

@CactusCon

@xntrik

Especially for incident response.. get your security guys and dev guys playing with this. If they haven’t had an opportunity to play with BeEF or Metasploit, now is the time.

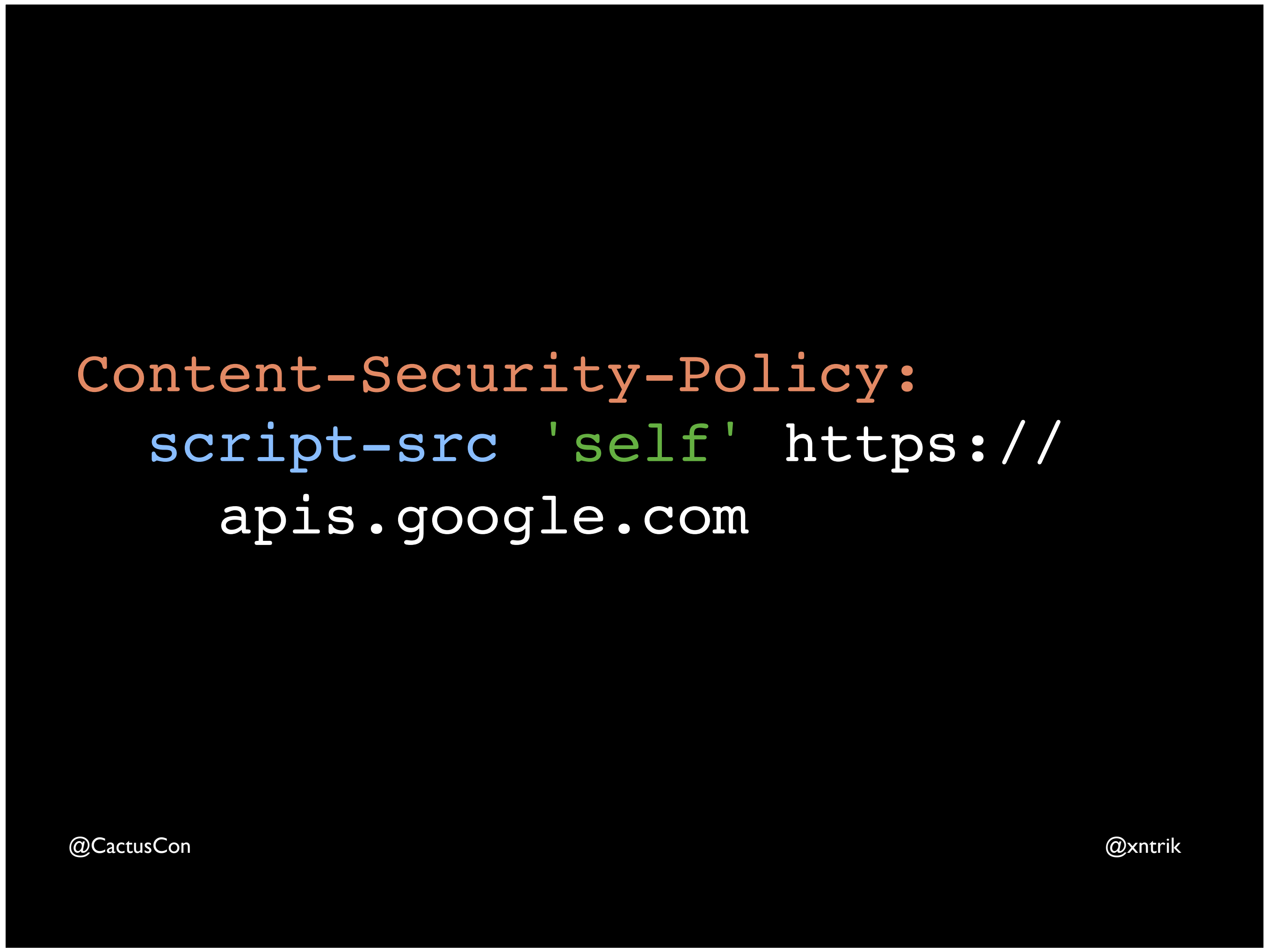
We're getting better at managing XSS (sorta)

Content Security Policy!

@CactusCon

@xntrik

(If) implemented properly, can make injecting arbitrary content more difficult



```
Content-Security-Policy:
  script-src 'self' https://
    apis.google.com
```

@CactusCon @xntrik

There are a few gotchas with CSP, in particular, if trying to retrofit the configuration over an existing JavaScript heavy site. For example, it works better if all your JavaScript logic is defined in external files which you can then ‘allow’, if you have a lot of inline JavaScript then it’s a bit more complex and may not work as effectively.

Thanks

- @WadeAlcorn
- @Antisnatchor
- @BeefProject
- ALL THE BeEF DEVS

- LinkedIn Assessment Crew
- Team Asterisk (Perth / Australia's radical sec team)