Hollywood mode off

Security testing at (large) scale

Claudio Criscione - @paradoxengine

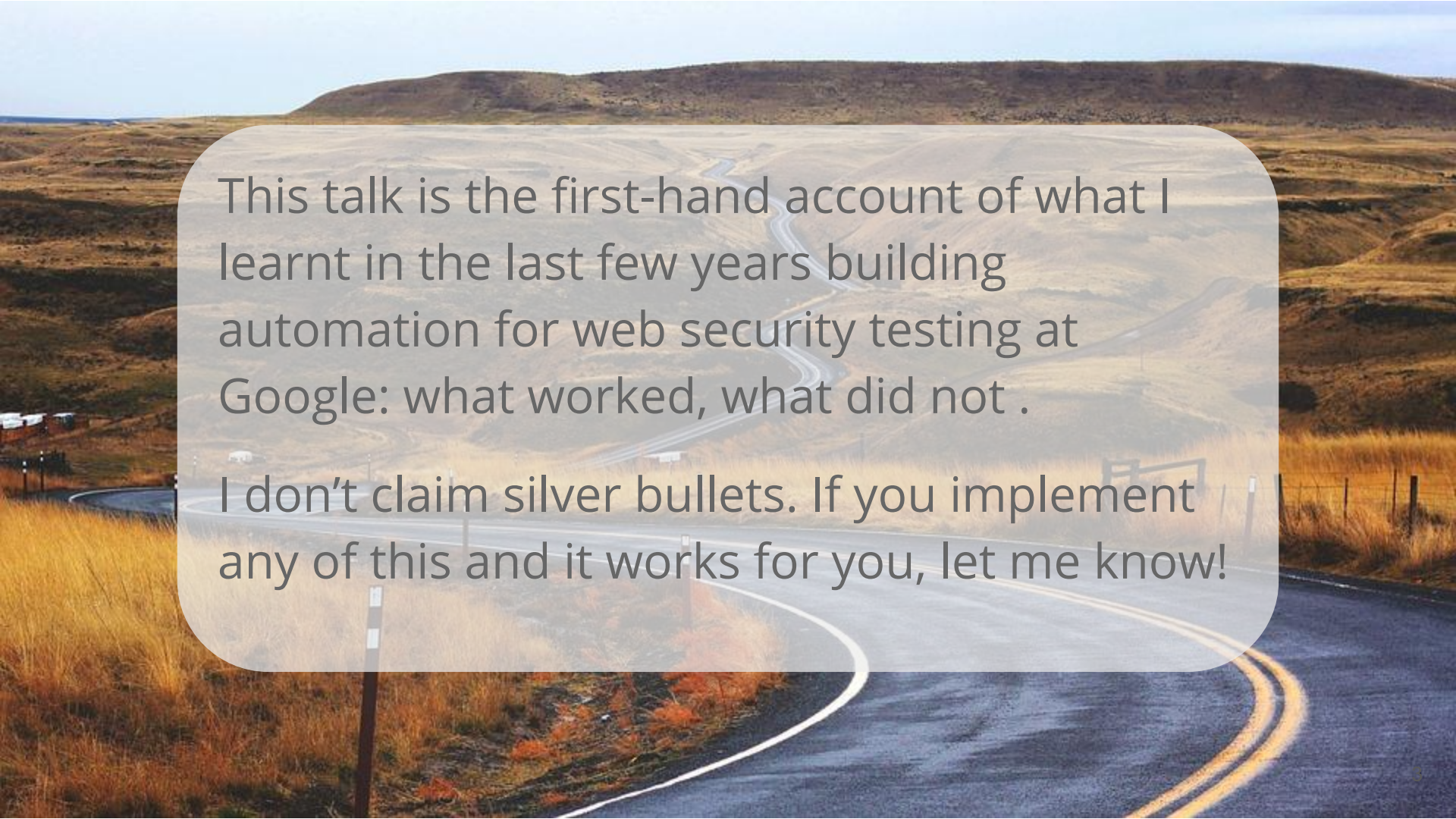
Who is this guy on stage?

Claudio Criscione

@paradoxengine

Security Automation @ Google Zurich

Fundamentally a “web security guy”



This talk is the first-hand account of what I learnt in the last few years building automation for web security testing at Google: what worked, what did not .

I don’t claim silver bullets. If you implement any of this and it works for you, let me know!

How do you “solve security” at a scale?


The Hollywood hacker

Scale kills the hacker stars.

Google’s 30,000+ engineers generate more than **30k CLs/day**, on a single-repo codebase with **2 billion lines of code** and over 86 TBs**.**

Any of those CLs might introduce security bugs.

Can’t I just hire the problem away?

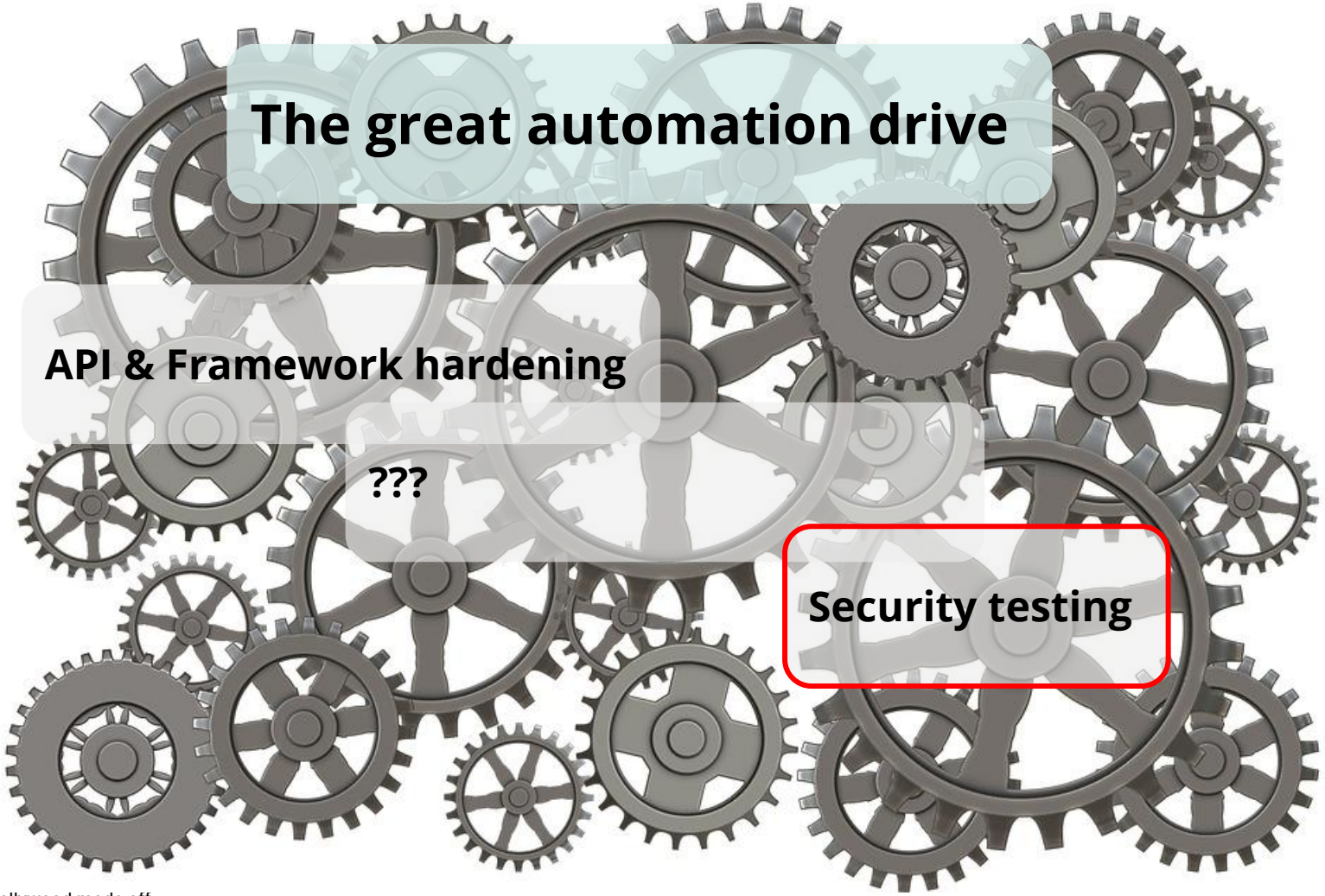
Problem 1

You can’t keep hiring security people, but you will (hopefully) keep growing!

Problem 2

Hiring (competent) security people is really hard!

The great automation drive



API & Framework hardening

???

Security testing

Enter security testing tools

So what’s the problem with that?

Most security tools are by security people for security people.

Remember the scale issue?

Target:
  At least one of these options has to be provided to define the
  target(s)

  -d DIRECT   Connection string for direct database connection
  -u URL, --url=URL Target URL (e.g. "http://www.site.com/vuln.php?id=1")
  -l LOGFILE   Parse target(s) from Burp or WebScarab proxy log file
  -x SITEMAPURL   Parse target(s) from remote sitemap(.xml) file
  -m BULKFILE   Scan multiple targets given in a textual file
  -r REQUESTFILE   Load HTTP request from a file
  -g GOOGLEDORK   Process Google dork results as target URLs
  -c CONFIGFILE   Load options from a configuration INI file

Request:
  These options can be used to specify how to connect to the target URL

  --method=METHOD   Force usage of given HTTP method (e.g. PUT)
  --data=DATA   Data string to be sent through POST
  --param-del=PARA.. Character used for splitting parameter values
  --cookie=COOKIE   HTTP Cookie header value
  --cookie-del=COO.. Character used for splitting cookie values
  --load-cookies=L.. File containing cookies in Netscape/wget format
  --drop-set-cookie Ignore Set-Cookie header from response
  --user-agent=AGENT HTTP User-Agent header value
  --random-agent   Use randomly selected HTTP User-Agent header value
  --host=HOST   HTTP Host header value
  --referer=REFERER HTTP Referer header value
  --headers=HEADERS Extra headers (e.g. "Accept-Language: fr\nETag: 123")
  --auth-type=AUTH.. HTTP authentication type (Basic, Digest, NTLM or PKI)
  --auth-cred=AUTH.. HTTP authentication credentials (name:password)
  --auth-private=A.. HTTP authentication PEM private key file
  --ignore-401   Ignore HTTP Error 401 (Unauthorized)
  --proxy=PROXY   Use a proxy to connect to the target URL
  --proxy-cred=PRO.. Proxy authentication credentials (name:password)
  --proxy-file=PRO.. Load proxy list from a file
  --ignore-proxy   Ignore system default proxy settings
  --tor   Use Tor anonymity network
  --tor-port=TORPORT Set Tor proxy port other than default
  --tor-type=TORTYPE Set Tor proxy type (HTTP (default), SOCKS4 or SOCKS5)
  --check-tor   Check to see if Tor is used properly
  --delay=DELAY   Delay in seconds between each HTTP request
  --timeout=TIMEOUT Seconds to wait before timeout connection (default 30)
  --retries=RETRIES Retries when the connection timeouts (default 3)
  --randomize=RPARAM Randomly change value for given parameter(s)
  --safe-url=SAFURL URL address to visit frequently during testing
  --safe-freq=SAFREQ Test requests between two visits to a given safe URL
  --skip-urlencode   Skip URL encoding of payload data
  --csrf-token=CSR.. Parameter used to hold anti-CSRF token
  --csrf-url=CSRFURL URL address to visit to extract anti-CSRF token
  --force-ssl   Force usage of SSL/HTTPS
  --hpp   Use HTTP parameter pollution method
  --eval=EVALCODE   Evaluate provided Python code before the request (e.g.
                     "import hashlib;id2=hashlib.md5(id).hexdigest()")

Optimization:
  These options can be used to optimize the performance of sqlmap

  -o   Turn on all optimization switches
  --predict-output   Predict common queries output
  --keep-alive   Use persistent HTTP(s) connections
  --null-connection Retrieve page length without actual HTTP response body
  --threads=THREADS Max number of concurrent HTTP(s) requests (default 1)

Injection:
  These options can be used to specify which parameters to test for,
  provide custom injection payloads and optional tampering scripts

  -p TESTPARAMETER   Testable parameter(s)
  --skip=SKIP   Skip testing for given parameter(s)
  --dbms=DBMS   Force back-end DBMS to this value
  --dbms-cred=DBMS.. DBMS authentication credentials (user:password)
  --os=OS   Force back-end DBMS operating system to this value
  --invalid-bignum   Use big numbers for invalidating values
  --invalid-logical Use logical operations for invalidating values
  --invalid-string   Use random strings for invalidating values
  --no-cast   Turn off payload casting mechanism
  --no-escape   Turn off string escaping mechanism
  --prefix=PREFIX   Injection payload prefix string
  --suffix=SUFFIX   Injection payload suffix string
  --tamper=TAMPER   Use given script(s) for tampering injection data

Detection:
  These options can be used to customize the detection phase

  --level=LEVEL   Level of tests to perform (1-5, default 1)
  --risk=RISK   Risk of tests to perform (0-3, default 1)
  --string=STRING   String to match when query is evaluated to True
  --not-string=NOT.. String to match when query is evaluated to False
  --regexp=REGEXP   Regexp to match when query is evaluated to True
  --code=CODE   HTTP code to match when query is evaluated to True
  --text-only   Compare pages based only on the textual content
  --titles   Compare pages based only on their titles

Techniques:
  These options can be used to tweak testing of specific SQL injection
  techniques

  --technique=TECH   SQL injection techniques to use (default "BEUSTQ")
  --time-sec=TIMESEC Seconds to delay the DBMS response (default 5)
  --union-cols=UCOLS Range of columns to test for UNION query SQL injection
  --union-char=UCHAR Character to use for bruteforcing number of columns
  --union-from=UFROM Table to use in FROM part of UNION query SQL injection
  --dns-domain=DNS.. Domain name used for DNS exfiltration attack
  --second-order=S.. Resulting page URL searched for second-order response

Fingerprint:
  -f, --fingerprint Perform an extensive DBMS version fingerprint

Enumeration:
  These options can be used to enumerate the back-end database
  management system information, structure and data contained in the
  tables. Moreover you can run your own SQL statements

  -a, --all   Retrieve everything
  -b, --banner   Retrieve DBMS banner
  --current-user   Retrieve DBMS current user
  --current-db   Retrieve DBMS current database
  --hostname   Retrieve DBMS server hostname
  --is-dba   Detect if the DBMS current user is DBA
  --users   Enumerate DBMS users
  --passwords   Enumerate DBMS users password hashes
  --privileges   Enumerate DBMS users privileges
  --roles   Enumerate DBMS users roles
  --dbs   Enumerate DBMS databases
  --tables   Enumerate DBMS database tables
  --columns   Enumerate DBMS database table columns
  --schema   Enumerate DBMS schema
  --count   Retrieve number of entries for table(s)
  --dump   Dump DBMS database table entries
  --dump-all   Dump all DBMS databases tables entries
  --search   Search column(s), table(s) and/or database name(s)
  --comments   Retrieve DBMS comments
  -D DB   DBMS database to enumerate
  -T TBL   DBMS database table(s) to enumerate
  -C COL   DBMS database table column(s) to enumerate
  -X EXCLUDECOL   DBMS database table column(s) to not enumerate
  -U USER   DBMS user to enumerate
  --exclude-sysdbs   Exclude DBMS system databases when enumerating tables
  --where=DUMPWHERE Use WHERE condition while table dumping
  --start=LIMITSTART First query output entry to retrieve
  --stop=LIMITSTOP   Last query output entry to retrieve
  --first=FIRSTCHAR First query output word character to retrieve
  --last=LASTCHAR   Last query output word character to retrieve
  --sql-query=QUERY SQL statement to be executed
  --sql-shell   Prompt for an interactive SQL shell
  --sql-file=SQLFILE Execute SQL statements from given file(s)

Brute force:
  These options can be used to run brute force checks

  --common-tables   Check existence of common tables
  --common-columns   Check existence of common columns

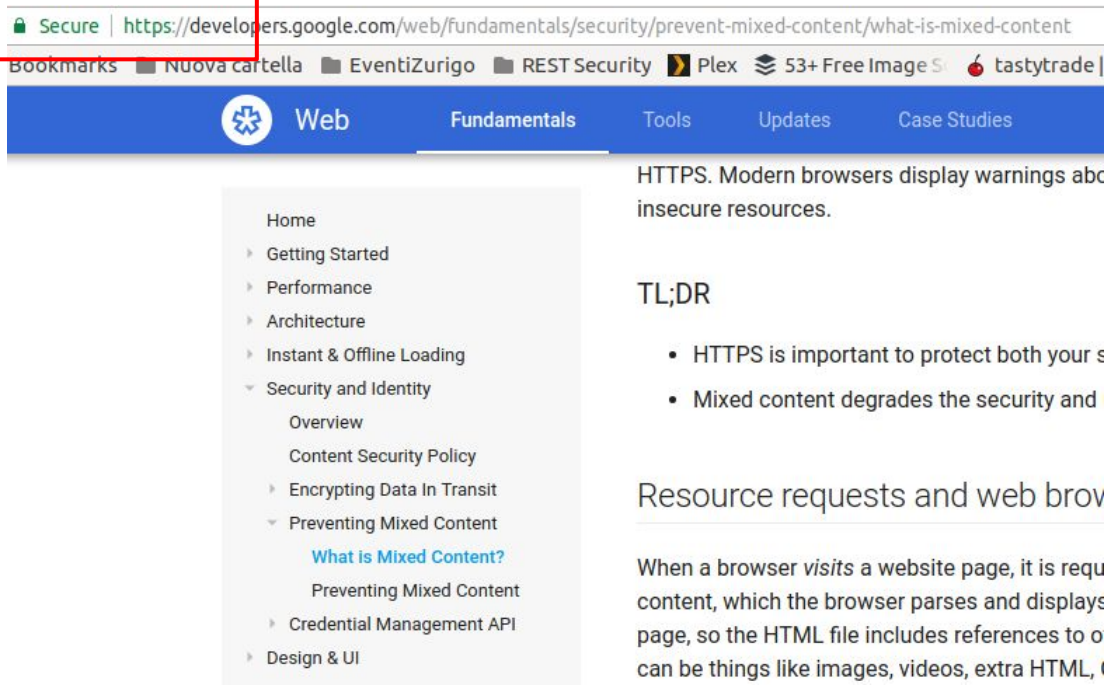
User-defined function injection:
  These options can be used to create custom user-defined functions

  --udf-inject   Inject custom user-defined functions
  --shared-lib=SHLIB Local path of the shared library

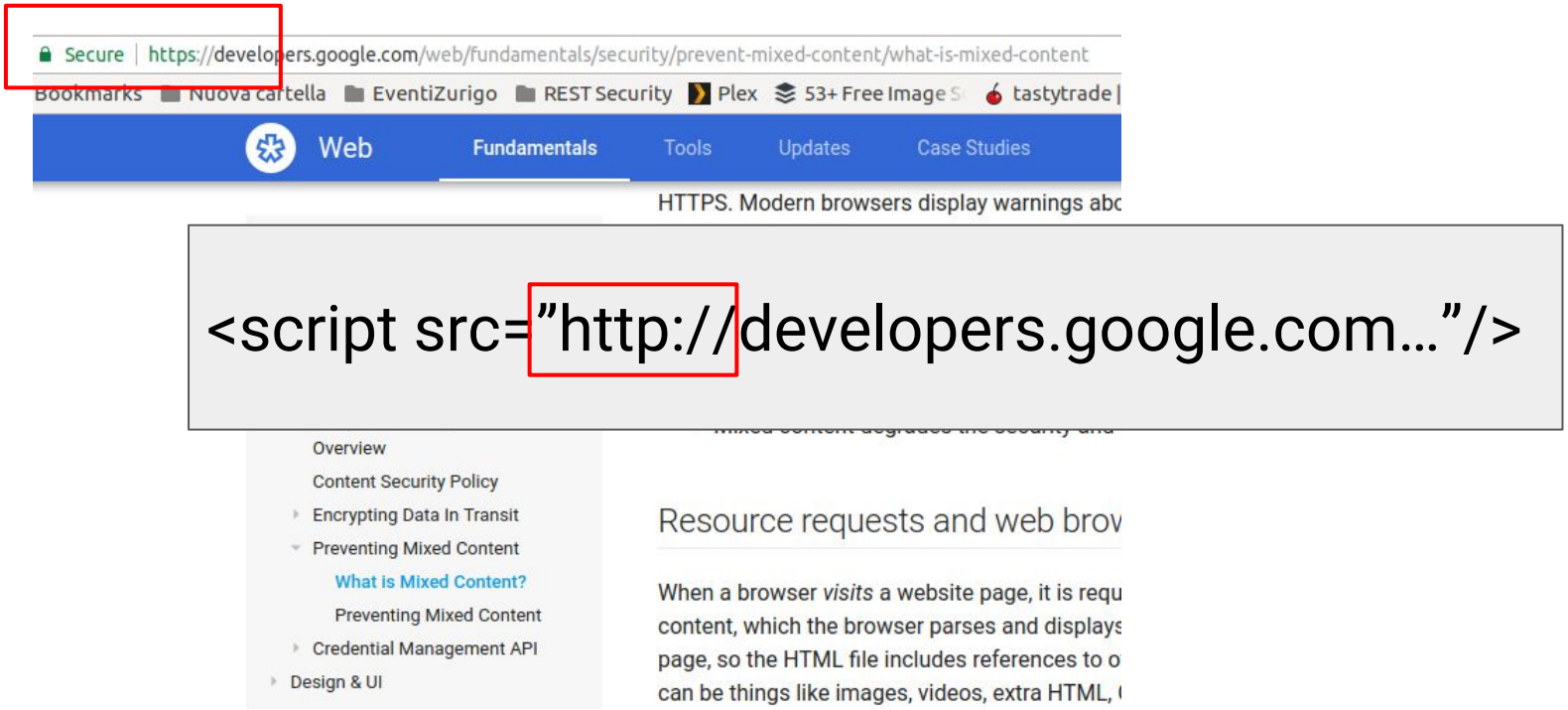
File system access:
  These options can be used to access the back-end database management
  system underlying file system

A simple case study: Hunting for mixed content

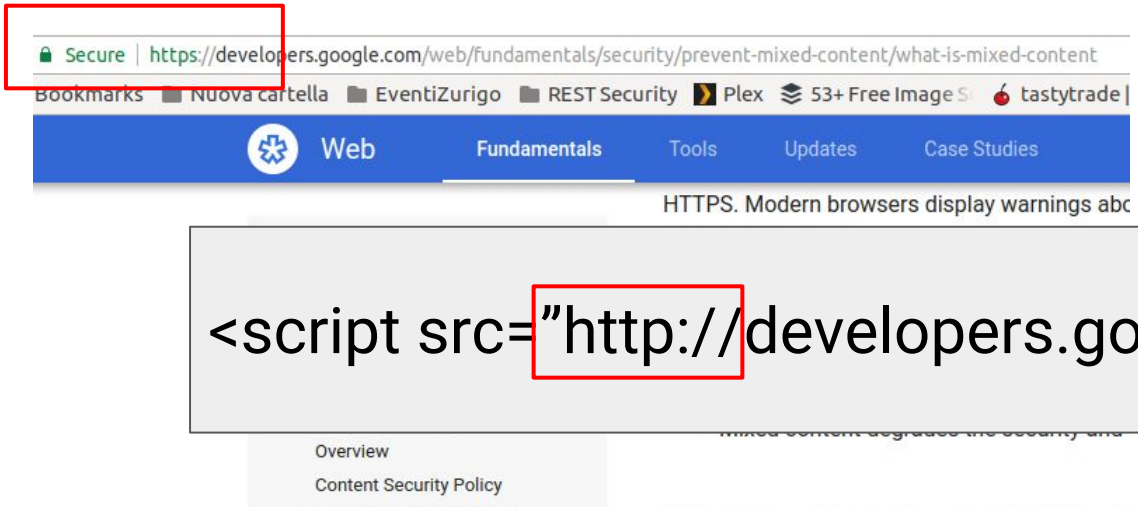
What is mixed content



What is mixed content



What is mixed content




```
<script src=”http://developers.google.com…”/>
```

```
Mixed Content: The page at 'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-distribution/avoid-mixed-content/simple-example.html' was loaded over HTTPS, but requested an insecure script 'http://googlesamples.github.io/web-fundamentals/samples/discovery-and-distribution/avoid-mixed-content/simple-example.js'. This request has been blocked; the content must be served over HTTPS.
```
simple-example.html:1

Why is it hard to find it?

“Static” HTML: easy

<script src="http://badidea.go…"/>

Why is it hard to find it?

“Static” HTML: easy

<script src="http://badidea.go…"/>

Dynamic, on-load JS: still OK

<script>
…
a = document.createElement('script');
a.src = “htt” + “p://badidea.go…”;

Why is it hard to find it?

“Static” HTML: easy

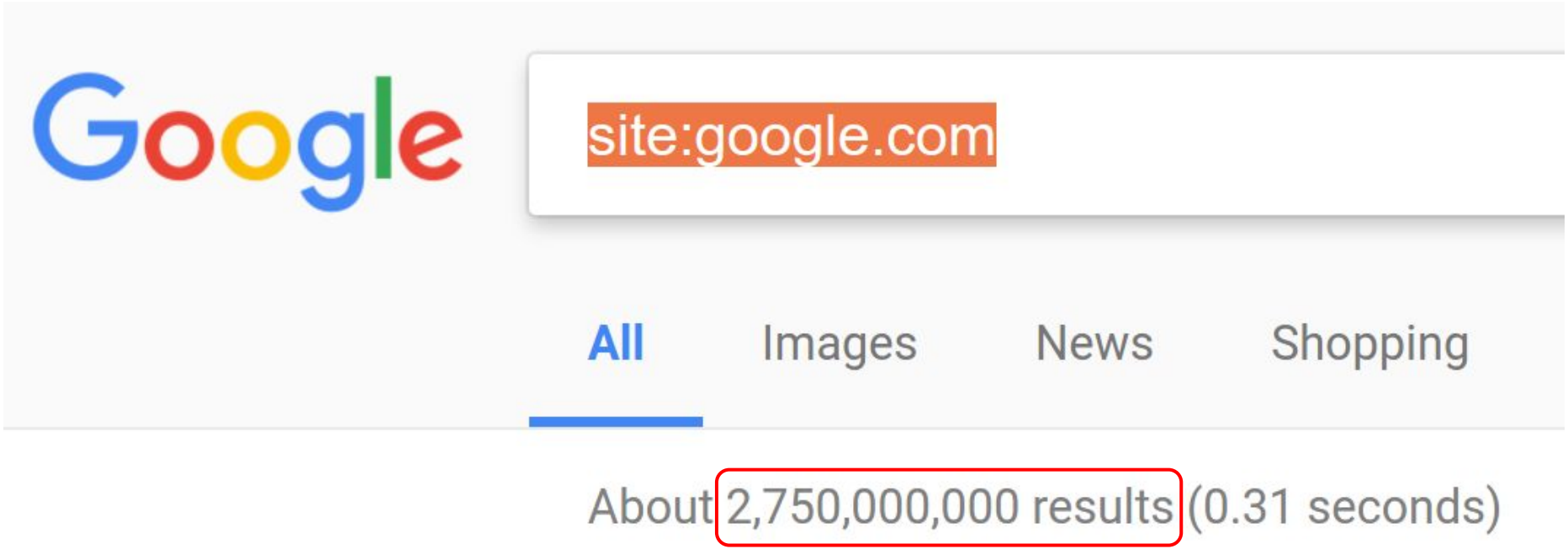
<script src="http://badidea.go…"/>

Interaction based events: good luck!

onclick = “runScriptThatLoadsHTTP”

Dynamic, on-load JS: still O

<script>
…
a = document.createElement('script');
a.src = “htt” + “p://badidea.go…”;


Google
site:google.com
All
Images
News
Shopping
About 2,750,000,000 results (0.31 seconds)

If only we had some automated procedures lying around to interact with large portions of our applications.

Enter Selenium WebDriver

“Selenium is a portable software-testing framework for web applications”

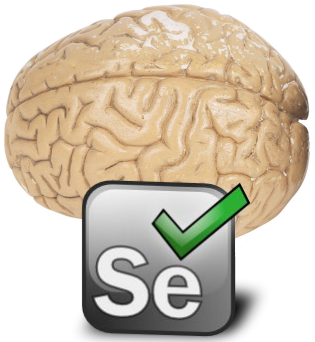
The coverage goals of end to end testing are in line with what we need.

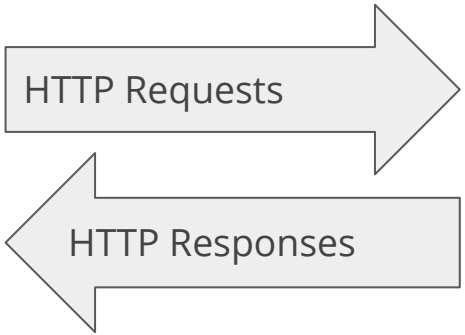
We have quite a few of them...

Google runs more than 150 million tests every day, and 13.000+ projects are continuously integrated.

That’s a lot of webdriver-based tests too.

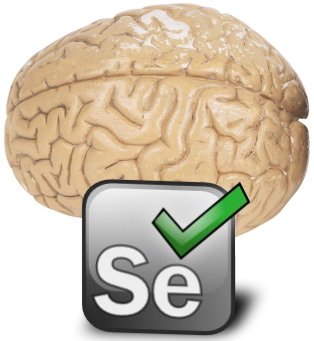
Identifying mixed content via HTTP proxy





By Source, Fair use, https://en.wikipedia.org/w/index.php?curid=25925698

Identifying mixed content via HTTP proxy

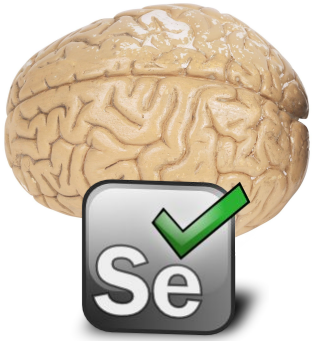


HTTP Requests

HTTP Responses

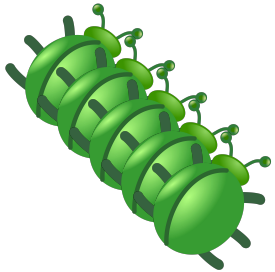
By Source, Fair use, https://en.wikipedia.org/w/index.php?curid=25925698

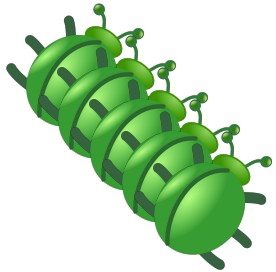
Identifying mixed content via HTTP proxy



HTTP Requests

HTTP Responses





Instrumenting webdriver tests

1. Add a proxy to the tests during setUp

```
MangoProxy mangoProxy = mangoBuilder.startProxy();
[...]
profile.setProxyPreferences(proxy)*;
```

2. Run the tests

*Recent webdriver code is likely to have a different syntax.

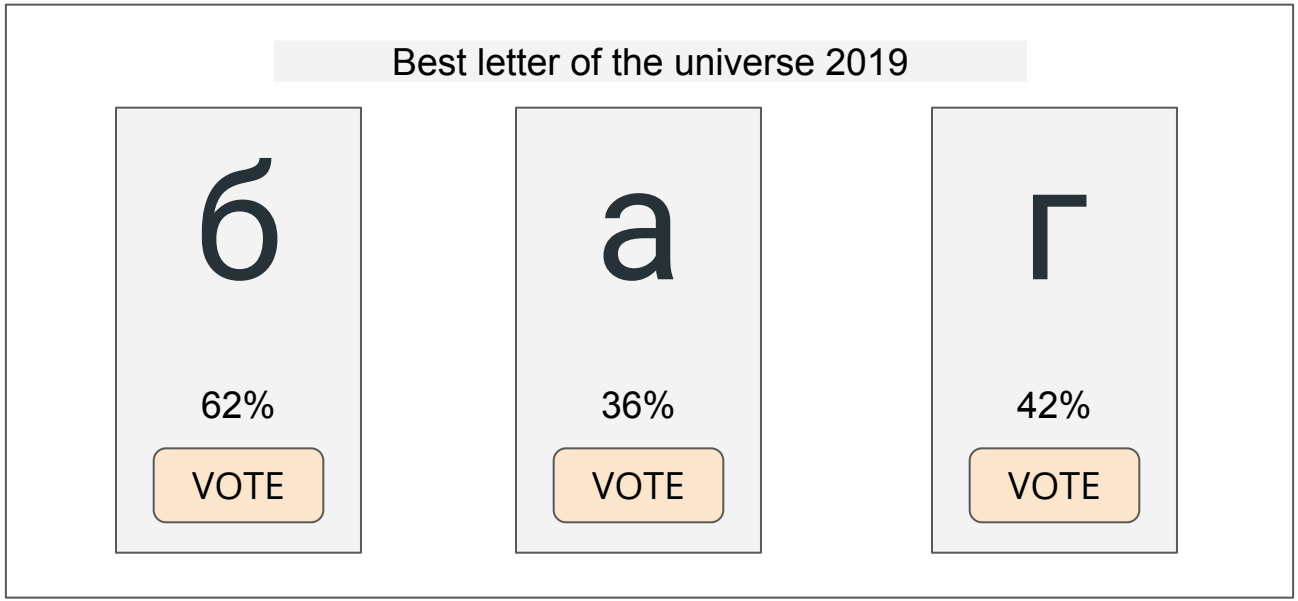
A more complex example: hunting cross site request forgery (XSRF)

What is XSRF?

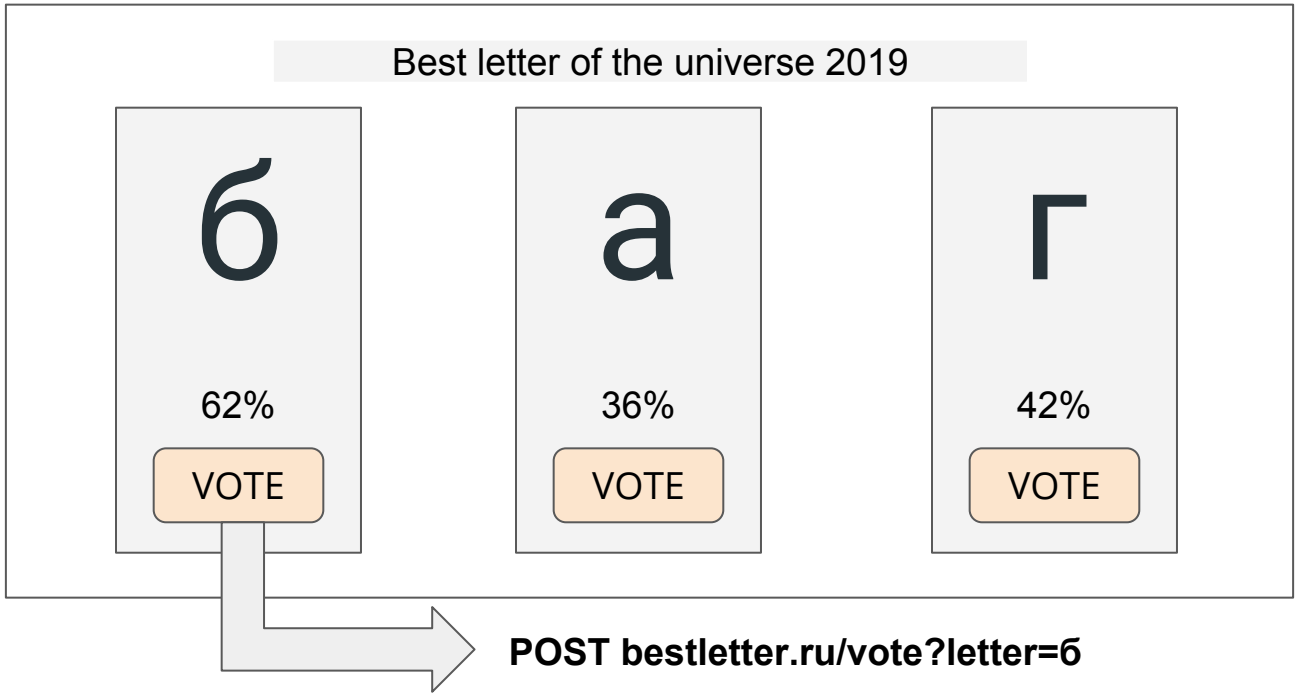
Using cross-site request forgery (XSRF), a malicious website can cause the victim's browser to make an authenticated request to a state-changing URL on another application, without the user's knowledge or consent.

If the target application doesn't take additional steps to confirm that the request is a result of a conscious user action, it’s bad news for the user.

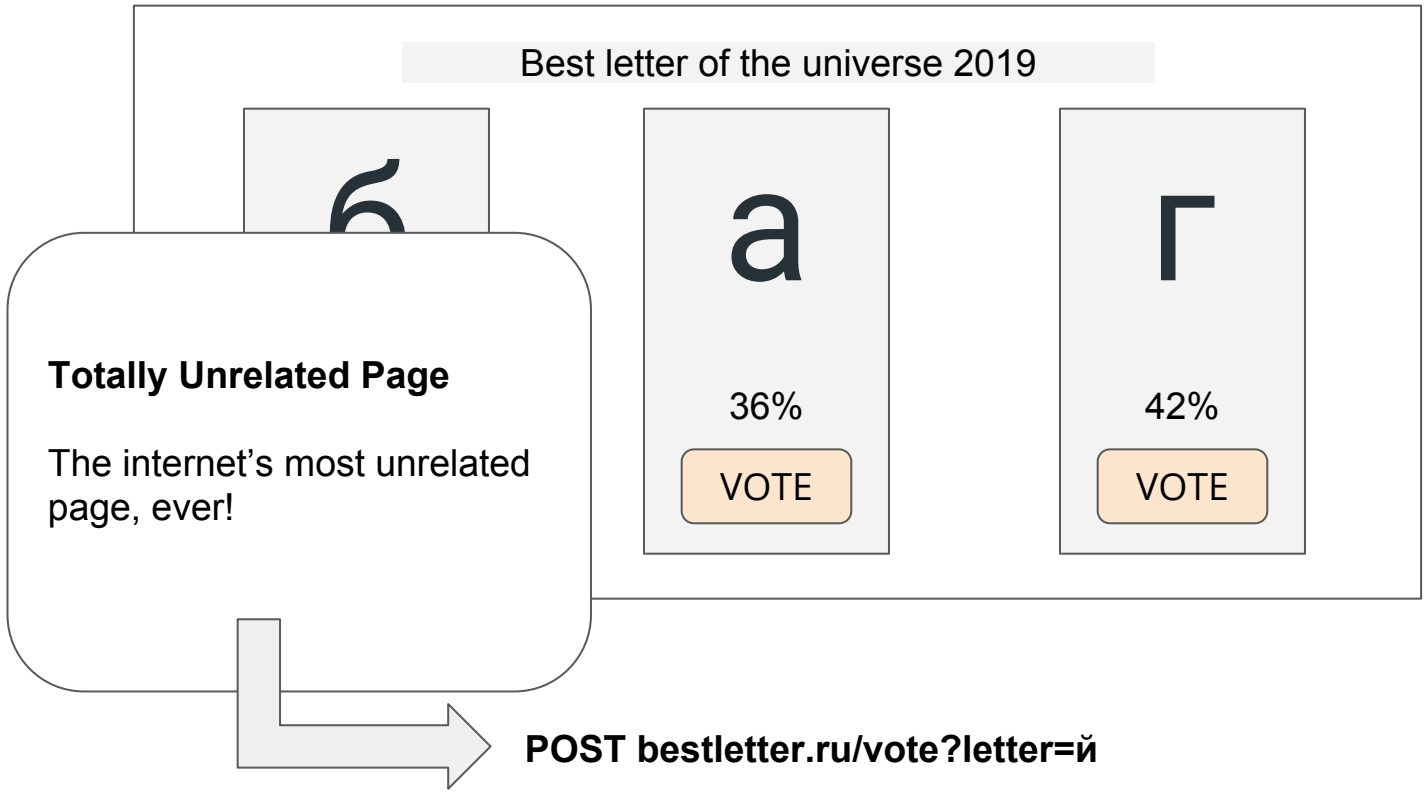
How does it look like?



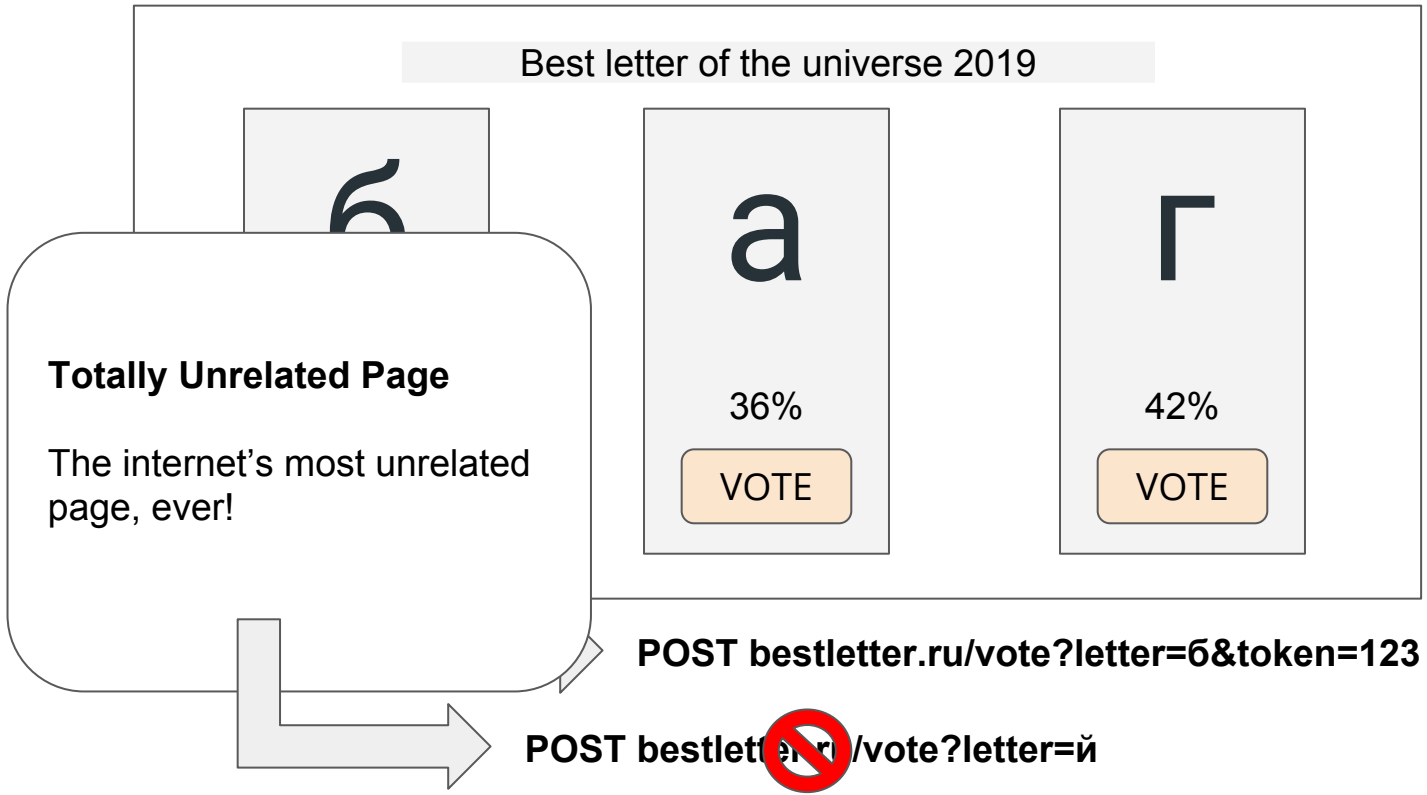
How does it look like?



How does it look like?



How does it look like?



Can we identify XSRF passively?

Build a list of token names, and check for their presence in all POST request.

Entropy analysis to guess for token values, headers, cookies...

Can we identify XSRF passively?

Build a list of token names, and check for their presence in all POST request.

Entropy analysis to guess for token values, headers, cookies...

Many false negative conditions
e.g. client adds token, server does not check

Can we identify XSRF passively?

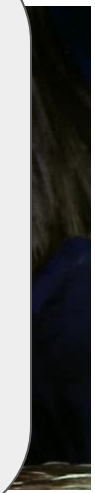
Build a
for the

Entrop
values

Many
e.g. cli
check

Execution after redirect

```
<?php
if (!$token) {
  http_response_code(403);
}
doSomething()
```



Can we identify XSRF passively?

Build a list of token names, and check for their presence in all POST request.

Entropy analysis to guess for token values, headers, cookies...

Many false negative conditions
e.g. client adds token, server does not check

Even more false positives
e.g. miss token

Can we identify XSRF passively?

Build a
for the

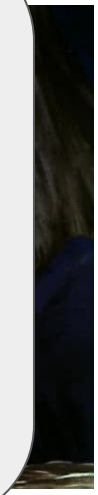
Entrop
values

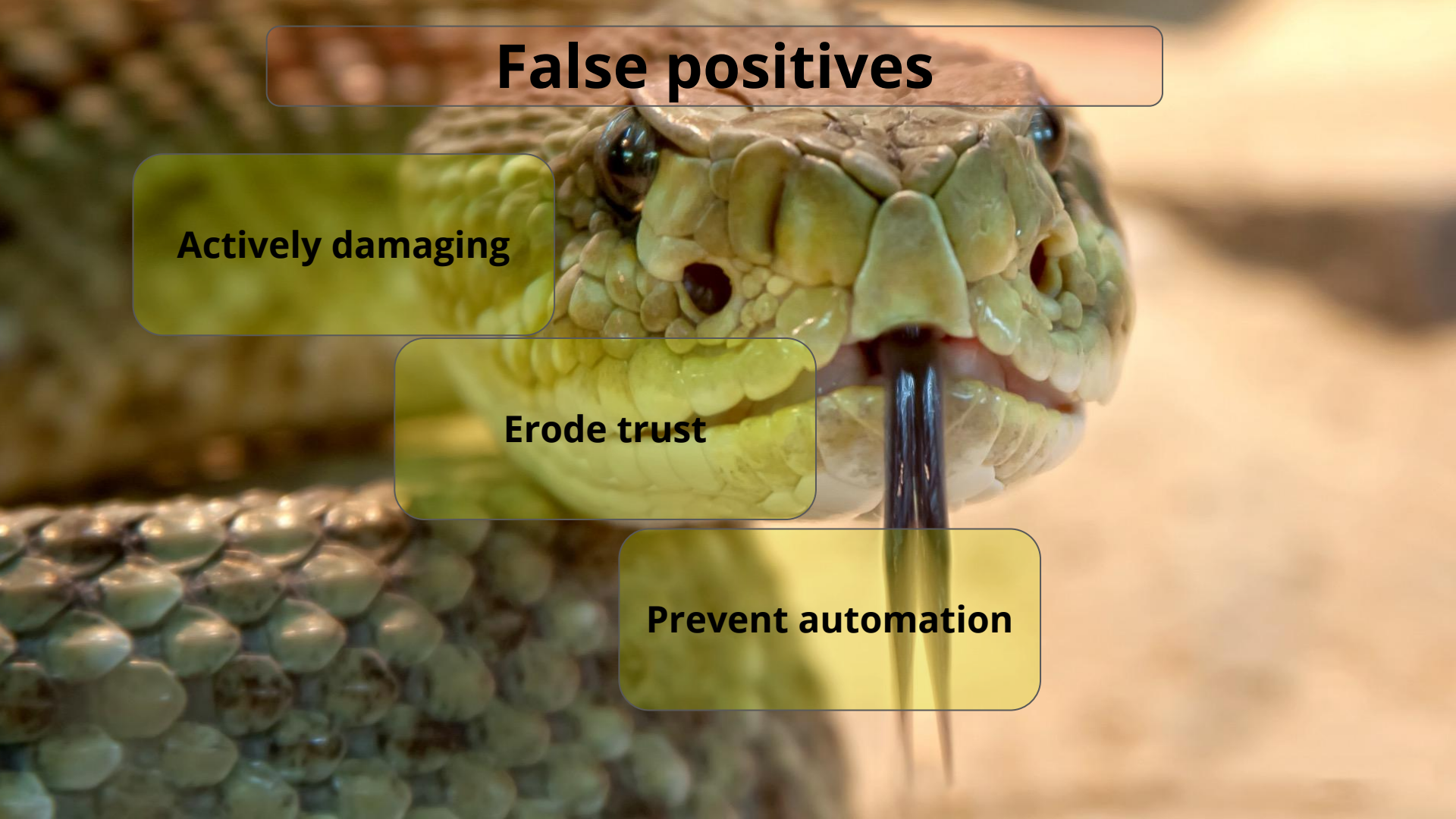
Many
e.g. cli
check

Even more false positives
e.g. miss token

What's the deal with false positives?

Can't you just report "potential bugs" and sort them out later? It's surely better than not knowing!




False positives
Actively damaging
Erode trust
Prevent automation

Always focus on low false positives, even at the cost of false negatives.

In the XSRF case: liberally over-flag tokens.

Repeat requests dropping tokens

POST /vote?letter=б&**token=XA...**

POST /vote?letter=б

Still flags irrelevant changes (false positives) and misses real bugs :-(

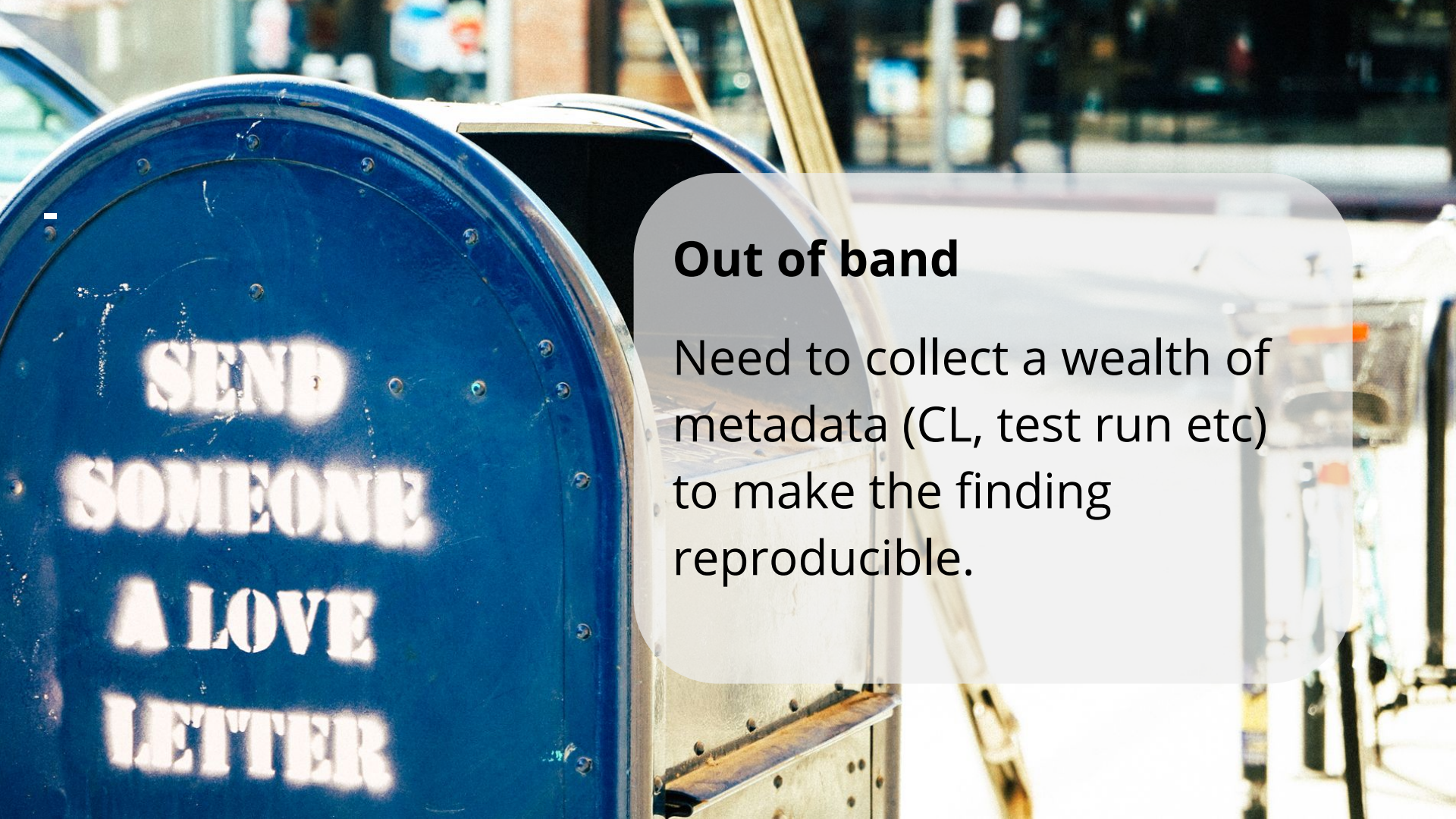
Mutation testing to the rescue

~~POST /vote?letter=б&**token=XA...**~~

POST /vote?letter=б

1. X% chance of mutating seemingly XSRF-Protected request, dropping XSRF token
2. Only mutate one request per test run
3. Flag cases where the mutated test still passes


How do you communicate bugs?



Out of band

Need to collect a wealth of metadata (CL, test run etc) to make the finding reproducible.

Failing tests when security issues are found

1. Add a proxy to the tests during setUp
2. Run the tests.
3. **Query Mango as part of teardown.**
4. Mark the test as failure and surface the failure as you would any other: block release, fail the integration, turn on the red lights.

```
if (mangoProxy.foundBugs())
  fail("Security issues found: " + trace);
```


Promote existing tests
to find security bugs.
Produce actionable,
useful results.

Where’s the catch?

Test vs Prod environment

Security tests have different requirements than integration tests.

Subtle differences have a large impact.

Why would you have SSL certs in QA?

Why would you enable XSRF checks in QA?

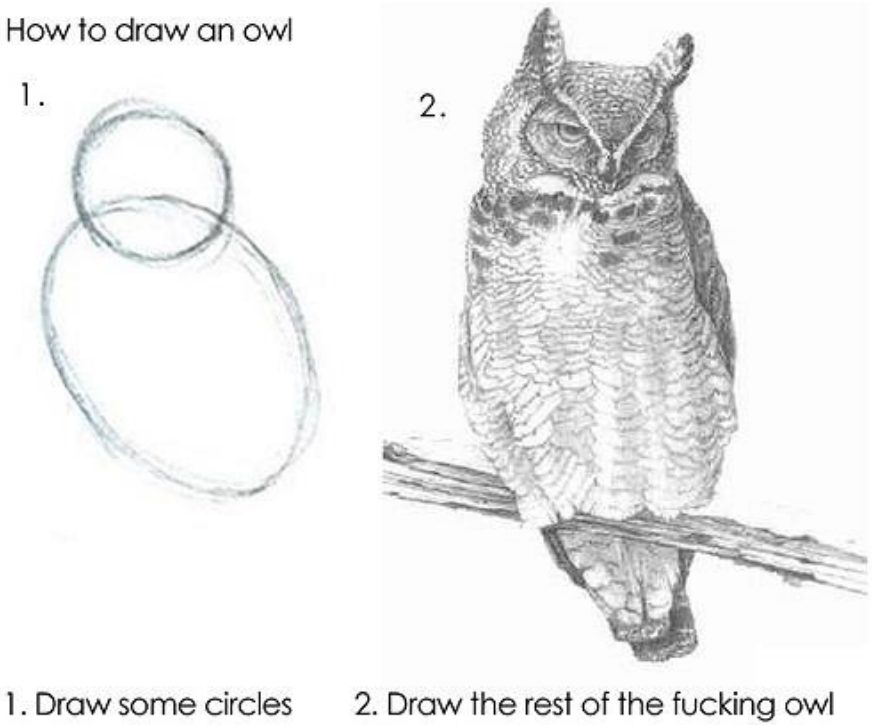
Test vs Prod environment

Security tests have different requirements than integration tests.

Consider (re)running instrumented integration tests against prod (!!).

Why would you enable XSRF checks in QA?

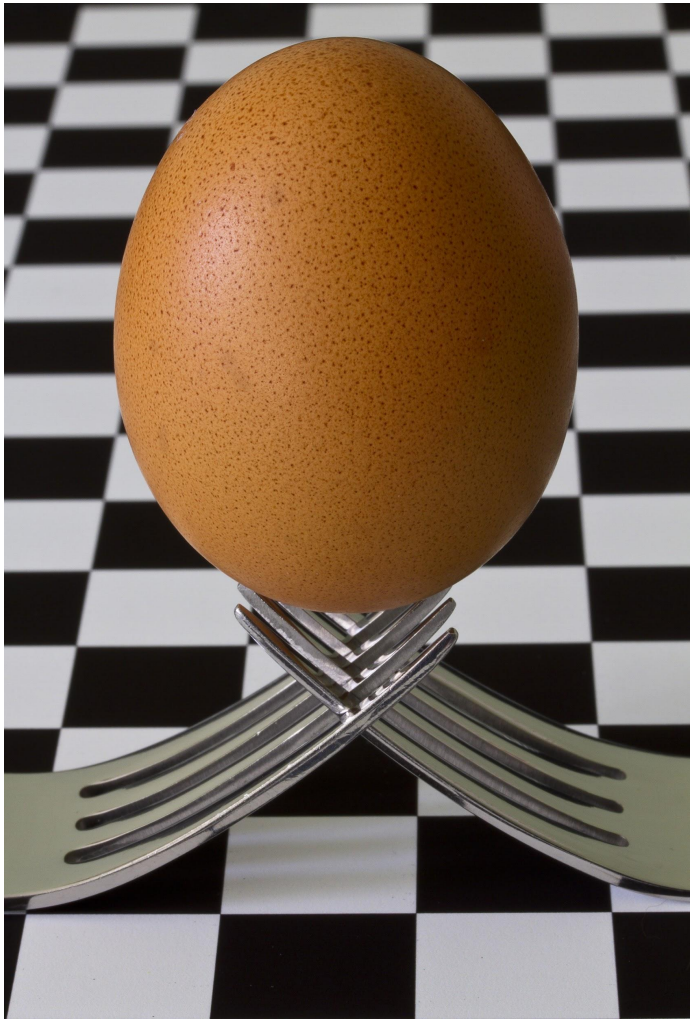
Tests take horrible shortcuts



Brittle tests

Can turn brittle tests into flaky

- Increases latency
- Reorders some requests
- Changes states on request replay



A moving target

Even though the integration is easy, our engineers seem to like changing stuff!

Ever changing test frameworks required work to keep integration.

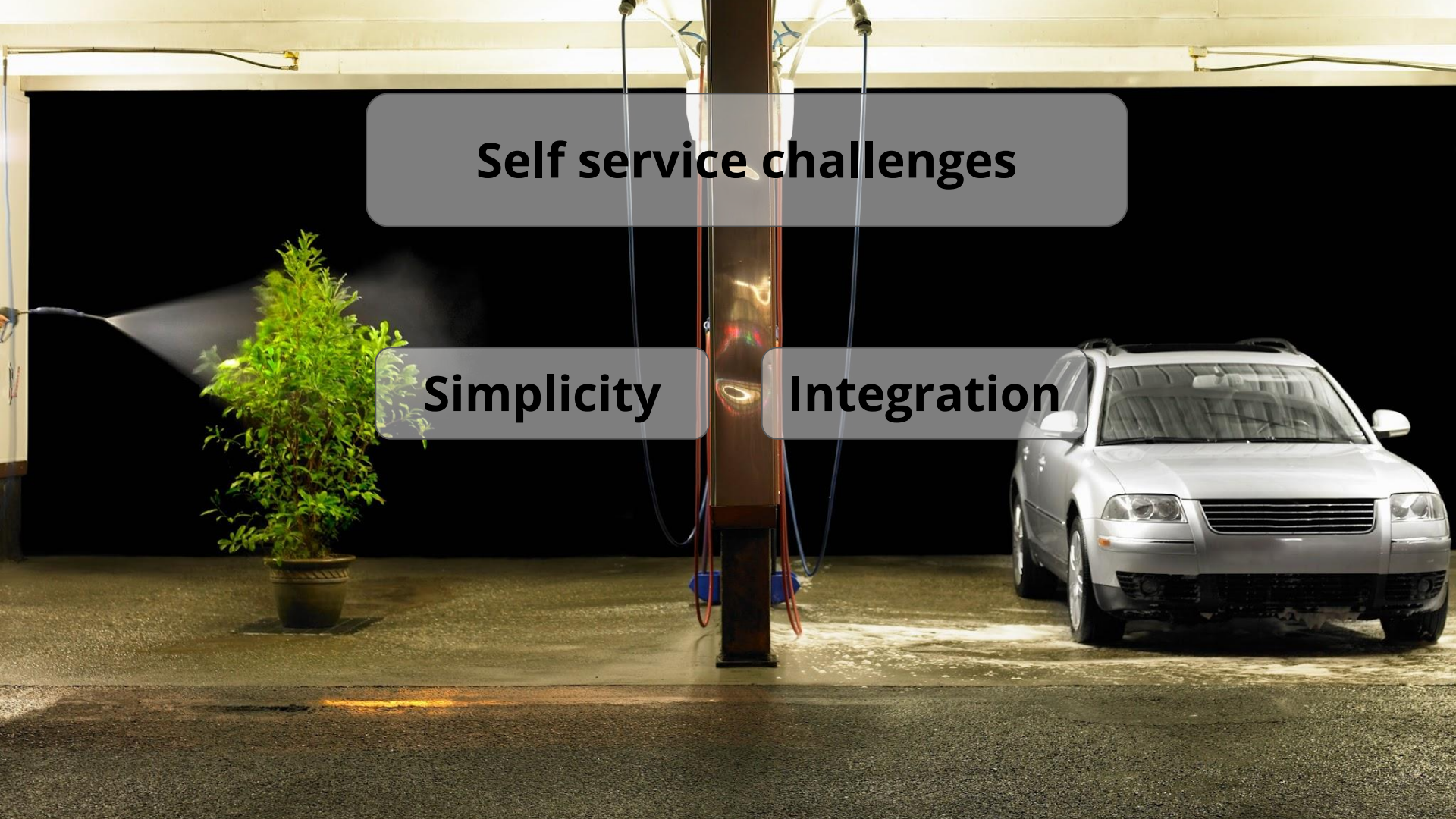
Did passive* test instrumentation work?

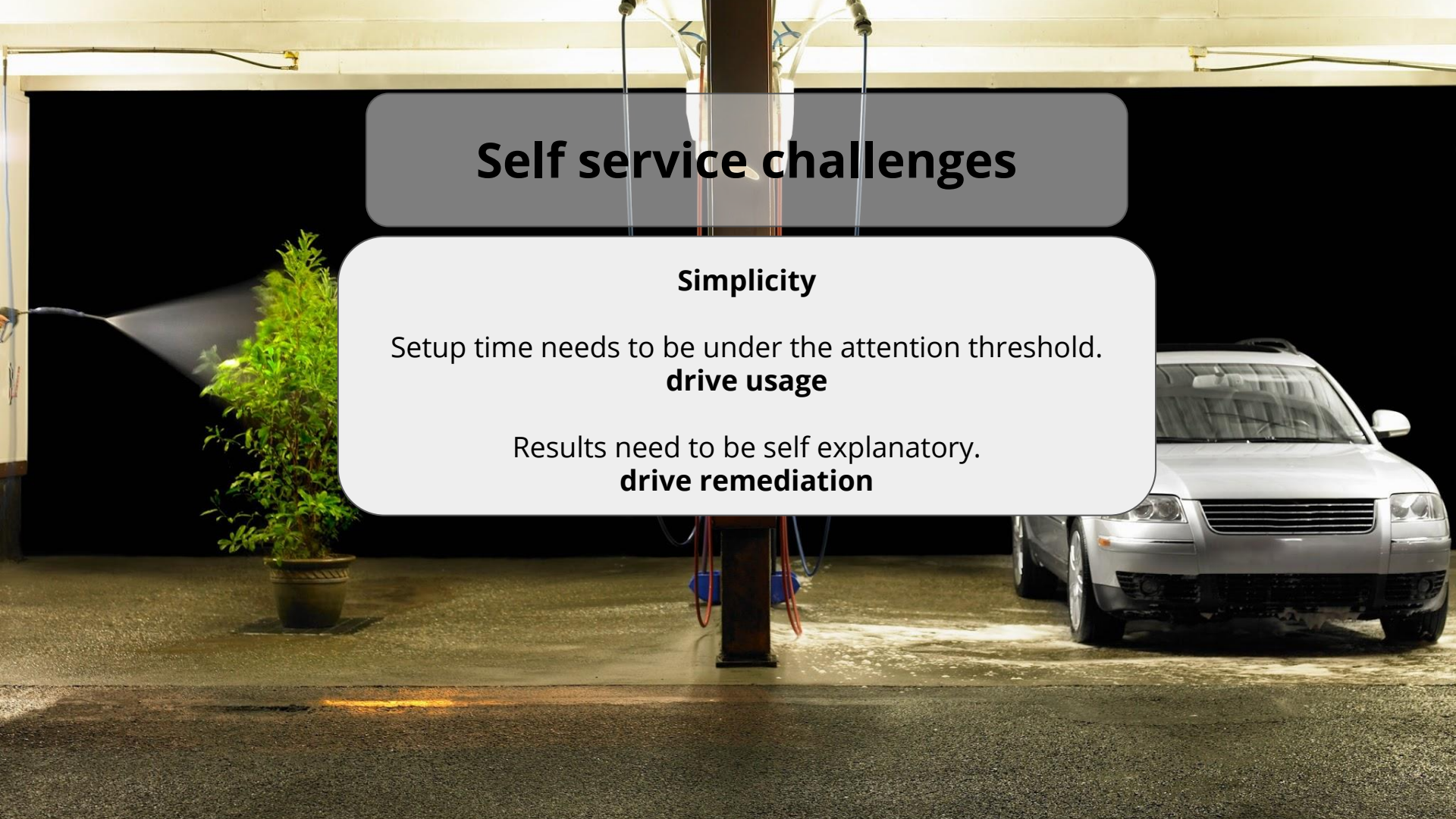
Almost.

It found a few* bugs.

We realized the cake really is a lie,
and moved aggressively to “self service”.

Don’t scale the security team, scale the security capabilities of others.


Self service challenges
Simplicity
Integration

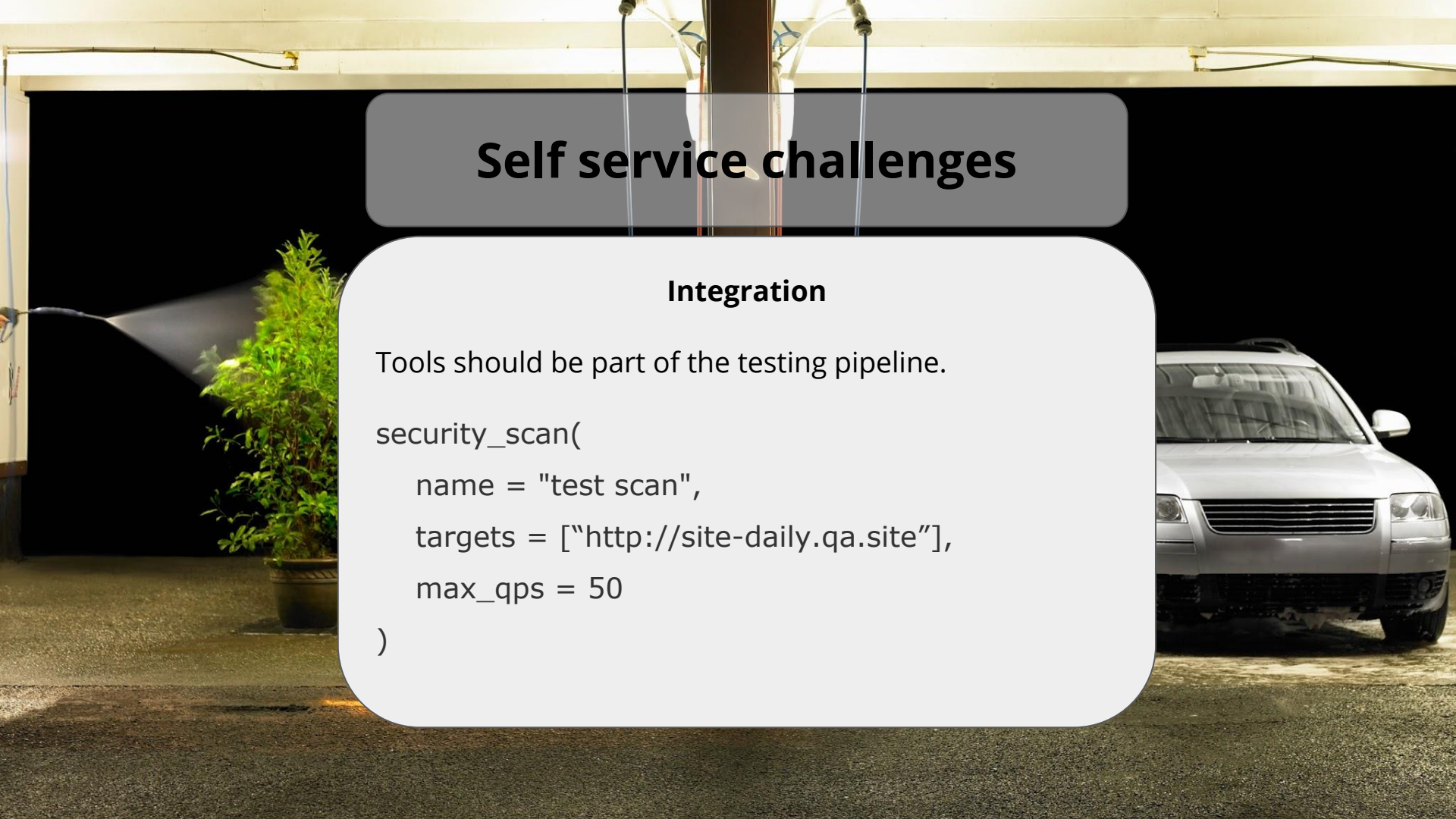


Self service challenges

Simplicity

Setup time needs to be under the attention threshold.
drive usage

Results need to be self explanatory.
drive remediation



Self service challenges

Integration

Tools should be part of the testing pipeline.

```
security_scan(
    name = "test scan",
    targets = [“http://site-daily.qa.site”],
    max_qps = 50
)
```

Did all of this work?

We identified more than 1500 security issues with our self-service approach.

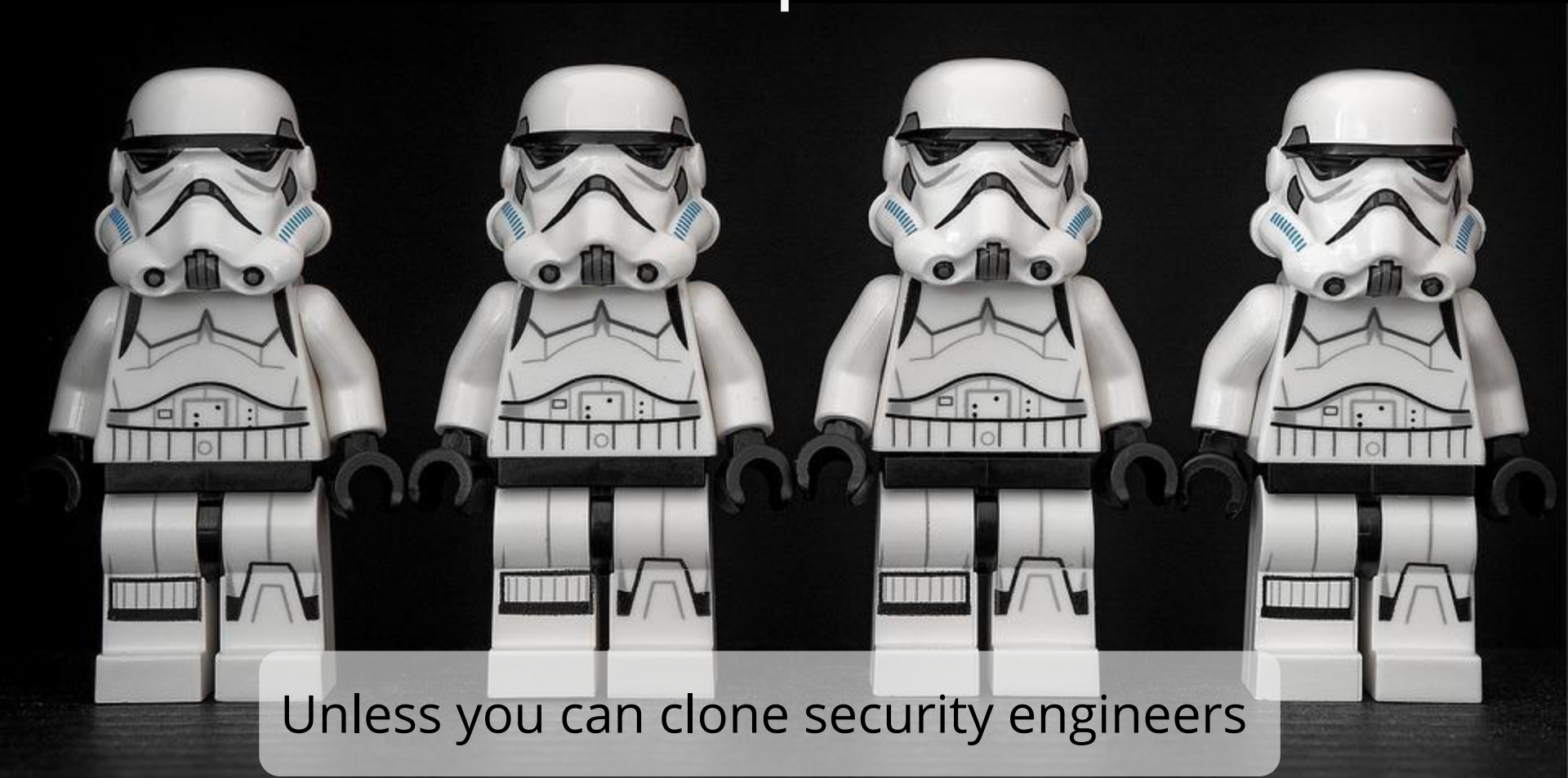
For reference, our Vulnerability Reward Program awarded 1000+ rewards in 2016.

Since we like a good challenge, we are now competing in the VRP ladder.
We'll see how we do in 2017!

Things I learnt

Wake up now!

Involve and empower teams

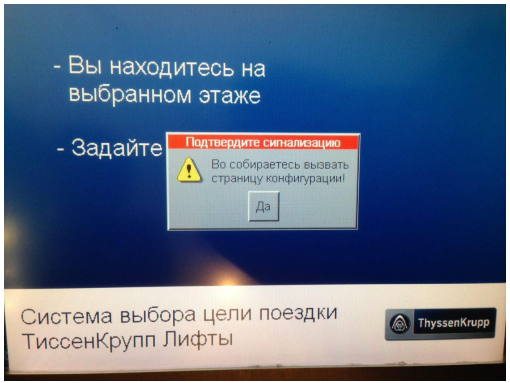


Unless you can clone security engineers


Focus on achievable targets
Go deep, not wide

Security is not the grinch

We just find different types of bugs



Thank you!
Questions?

Security testing at scale
Claudio Criscione - @paradoxengine