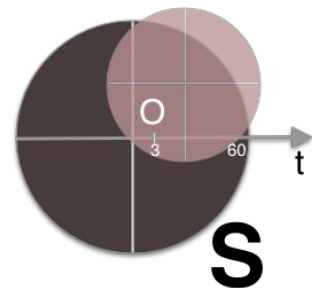


State of DNS Rebinding

Attack & Prevention Techniques and
the Singularity of Origin

Gérald Doussot & Roger Meyer | DEF CON 27



Contributions

New Tool: Singularity

- Everything you need for DNS rebinding
- **Reliable:** Default settings **just work**
- **Performant:** up to **20x faster** (3s vs 60s)
- **Built-in payloads:** Jenkins, Rails, AWS Instance Metadata, etc...
- **Autopwn:** Networks & services scan, **service detection and auto-exploitation**

Neat Technical Details/Techniques

- **Speed:** Multiple DNS answers and cache flooding
- **Protection Bypasses:** 3 **different bypasses** for common DNS rebinding protections
 - Using DNS trickery
- **Hook/Control** - Interactively browse the victim's internal network
 - Websocket proxying, no HTTP proxy needed!

Agenda

- A Refresher on DNS Rebinding
- DNS Rebinding Attack: Building on reliable foundations
- **The Need for Speed:** DNS rebinding in 3 seconds
- **Protection Bypasses:** 3 different bypasses for common DNS rebinding protections
- **Hook and Control:** Interactively browse the victim's internal network
- Scanning for Vulnerable Host Services
- **Automation:** Service detection & exploitation and orchestrating all the above

Introduction

Who Are We

- Gérald Doussot and Roger Meyer
- Security consultants at **nccgroup**
- San Francisco, CA
- Authors of **Singularity of Origin**,
a **DNS Rebinding Attack Framework**

<https://github.com/nccgroup/singularity>

The screenshot shows the GitHub repository page for **nccgroup / singularity**. The repository is described as "A DNS rebinding attack framework" with a link to <https://www.nccgroup.trust/us/about-us>. It has 172 commits, 9 branches, 0 releases, 5 contributors, and is licensed under MIT. The repository is tagged with **dns-rebinding**, **dns**, **vulnerability**, **attack**, and **iot**. The commit history shows a recent commit by **sanktjodel** adding support for CSRF tokens in Jenkins payload. The repository includes files such as **cmd/singularity-server**, **golang**, **html**, **screenshots**, **.gitignore**, **LICENSE**, **Readme.md**, **commandcontrol.go**, **firewall.go**, and **singularity.go**. The **Readme.md** file is expanded, showing the title **Singularity of Origin** and a description: "Singularity of Origin is a tool to perform DNS rebinding attacks. It includes the necessary components to rebind the IP address of the attack server DNS name to the target machine's IP address and to serve attack payloads to exploit vulnerable software on the target machine. It also ships with sample payloads to exploit several vulnerable software versions, from the simple capture of a home page to performing remote code execution. It aims at providing a framework to facilitate the exploitation of software vulnerable to DNS rebinding attacks and to raise awareness on how they work and how to protect from them. Detailed documentation is on the [wiki pages](#)."

Why Should You Care About DNS Rebinding

```
bind 0.0.0.0
```

Why Should You Care About DNS Rebinding

bind 127.0.0.1

Why Should You Care About DNS Rebinding



The screenshot shows a web browser window displaying a Medium article. The address bar shows the URL: <https://medium.com/coinmonks/the-call-is-coming-from-inside-the-house-dns-rebinding->. The Medium logo is visible in the top left. Below the logo, the text "Coinmonks" is followed by "CRYPTOFI" and "CREATE BITCOIN INVOICES". The article title is "The call is coming from inside the house — DNS rebinding in EOSIO keosd wallet". The author's name is "François Proulx in Coinmonks" with a "Follow" button. The date and time are "Jul 19, 2018 · 6 min read".

https://medium.com/coinmonks/the-call-is-coming-from-inside-the-house-dns-rebinding-

M

Coinmonks CRYPTOFI | CREATE BITCOIN INVOICES

The call is coming from inside the house — DNS rebinding in EOSIO keosd wallet

 François Proulx in Coinmonks [Follow](#)

Jul 19, 2018 · 6 min read

Why **Attacking Private Networks from the Internet with DNS Rebinding**

Corporation (US) | <https://medium.com/@brannondorsey/attacking-private-networks-from-the-internet-with-dns-rebinding>

Become a member

Attacking Private Networks from the Internet with DNS Rebinding



Brannon Dorsey [Follow](#)

Jun 19, 2018 · 20 min read

TL;DR Following the wrong link could allow remote attackers to control your WiFi router, Google Home, Roku, Sonos speakers, home thermostats and more.



Jul 19, 2018 · 6 min read

Just another hacking blog



[Home](#) [About](#) [Contact](#)

How your ethereum can be stolen through DNS rebinding

19 Jan 2018

With the new buzz around exploiting unauthenticated JSON-RPC services on localhost ignited by **Tavis Ormandy**, The first thing that came to my mind was ethereum clients(Geth, Mist and Parity).

Most of the ethereum clients run a JSON-RPC service on port 8545 on localhost, but since it's on localhost, we can't access it directly from user's browser due to **SOP**. This **issue** in the electrum wallet exploited the **CORS** headers to take control over the user's electrum wallet through JSON-RPC on localhost.

and more.



Jul 19, 2018 · 6 min read

Why

Corporation (U



Coinm

Att
Int



19 Jan

With th
first th

Most o
access
contro

T
at
H

and more.



Jul 19, 2018 · 6 min read

https://bl
https://bou.ke/blog/hacking-developers/

Bouke van der Bijl [Github](#) [Twitter](#)

How to steal any developer's local database

Aug 2016

If you're reading this and you're a software developer, you're probably running some services locally. Redis, Memcached, and Elasticsearch are software products that many rely on. What you might not know, is that these locally running services are accessible by any website you visit, making it possible for bad guys to steal the data you have locally!

Bouke van der Bijl

[Github](#)

[Twitter](#)

<https://labs.mwrinfosecurity.com/advisories/minikube-rce/>

MWR
LABS

[+ Advisories](#) [/var/log/messages](#) [Publications](#) [Tools](#) [Careers](#)

[← Advisories](#)

+

Minikube RCE & VM Escape

The Kubernetes dashboard service on Minikube is vulnerable to DNS rebinding attacks that can lead to remote code execution on the host.

Product	Minikube
Severity	High
CVE Reference	CVE-2018-1002103
Type	Remote Code Execution

https://bl https://bou.ke/blog/hacking-developers/

→ ↺ 🏠 ⓘ 🔒 https://benmmurphy.github.io/blog/2016/07/11/rails-webconsole-dns-rebinding/

Ben's Blog

Developer Musings

Blog | Archives

Rails Webconsole DNS Rebinding

The webconsole gem which ships with the Rails development server allows remote code execution via DNS Rebinding. I reported this issue to Rails on April 20th 2015. However, it may have been reported to them earlier because Homakov also found the issue independently and tweeted about it here:

📄 ... ☆ 🌟 ☰

es Publications Tools Careers

nning
that can lead to remote code execution on
ducts
nning
ays to

Why Should You Care About DNS Rebinding

- Prevalence of apps exposing **HTTP servers on localhost** (e.g. Electron)
- **IoT devices** exposing sensitive interfaces on internal networks
- **Misconceptions**
 - DNS rebinding is slow
 - DNS rebinding can be solved by out-of-the-box DNS filtering products or services

A Refresher on DNS Rebinding

On the Origin of Web Documents

The “**Origin**” of a resource is a tuple consisting of scheme, host and port.



Two documents A and B share the “**same-origin**” if they have identical scheme, host and port components.

- <https://mysite.com/hello.html> and <https://mysite.com/world.html> ✓
- <https://mysite.com/hello.html> and <https://attacker.com/hello.html> ✗

On the Origin of Web Documents

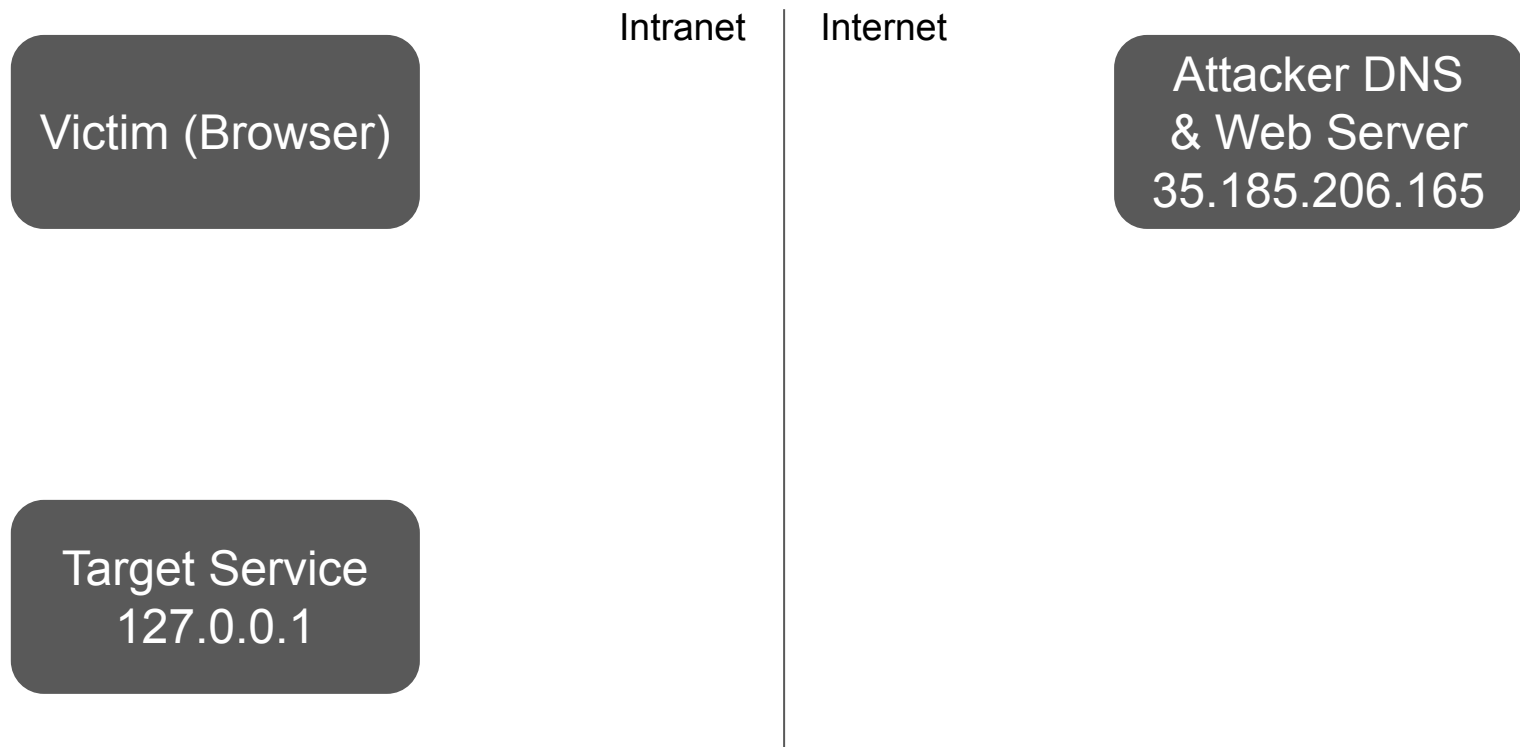
The “**same-origin policy**” dictates how two different origins may interact.

These interactions between origins are typically permitted: form submissions, links, redirects, content embedding (JavaScript, CSS).

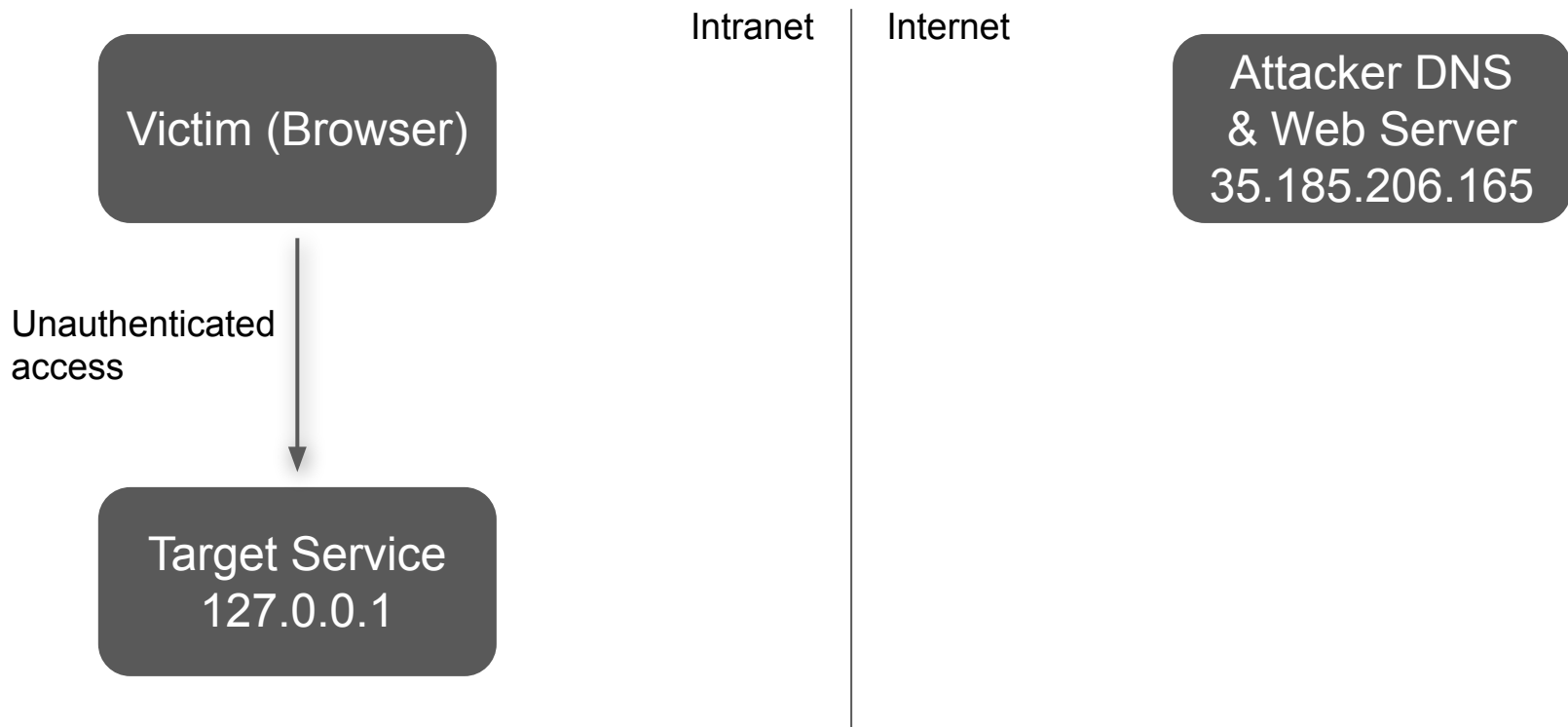
Cross-origin **reads** are typically not allowed e.g. reading the content of an HTML document located on [gmail.com](#) from site [attacker.com](#).

DNS Rebinding permits to bypass restrictions imposed by the **same-origin policy**.

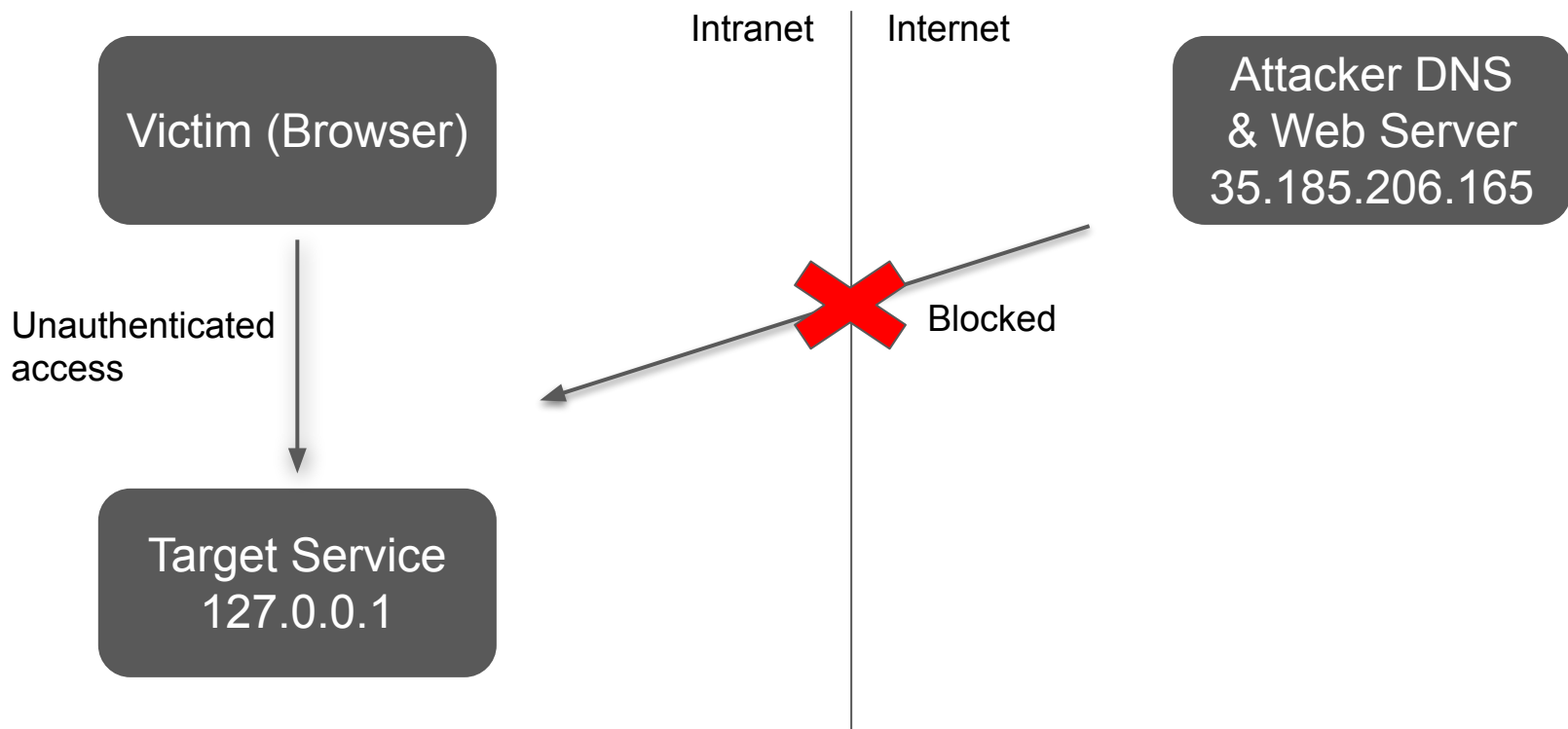
DNS Rebinding Attack Walkthrough



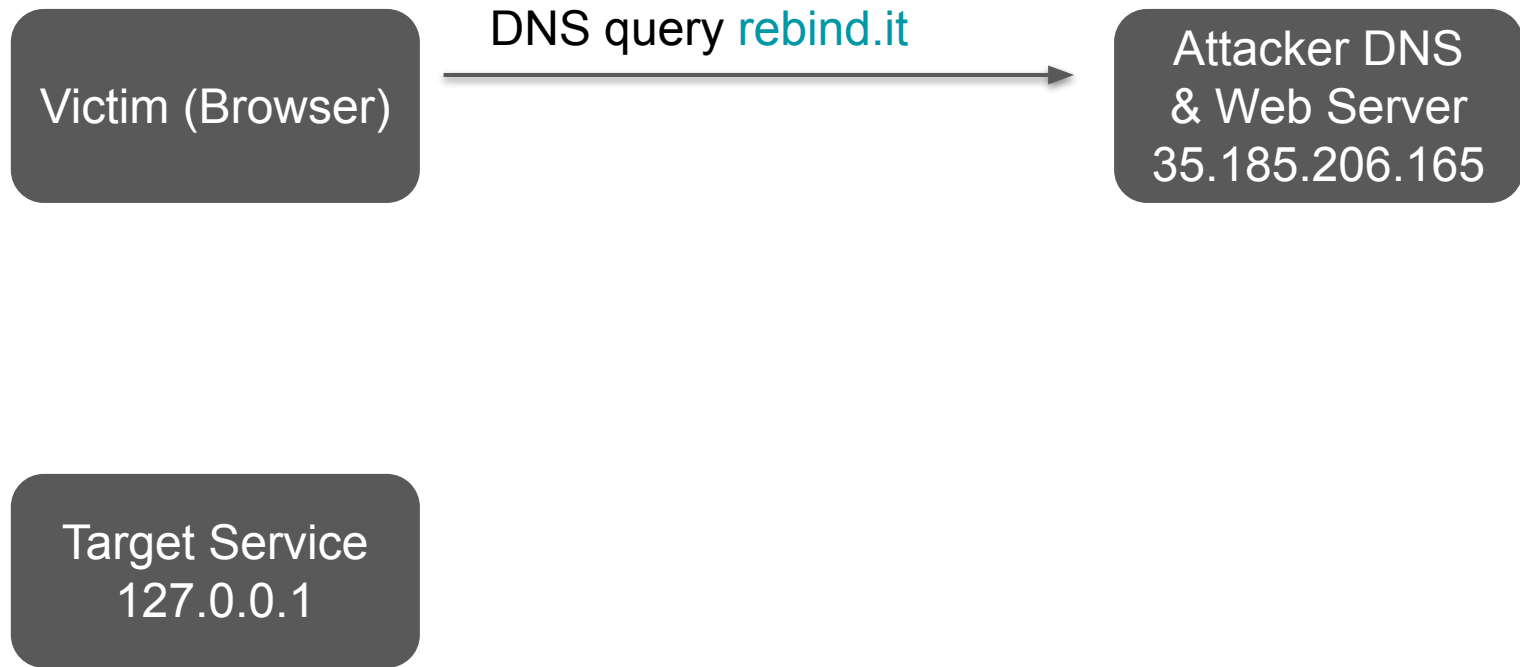
DNS Rebinding Attack Walkthrough



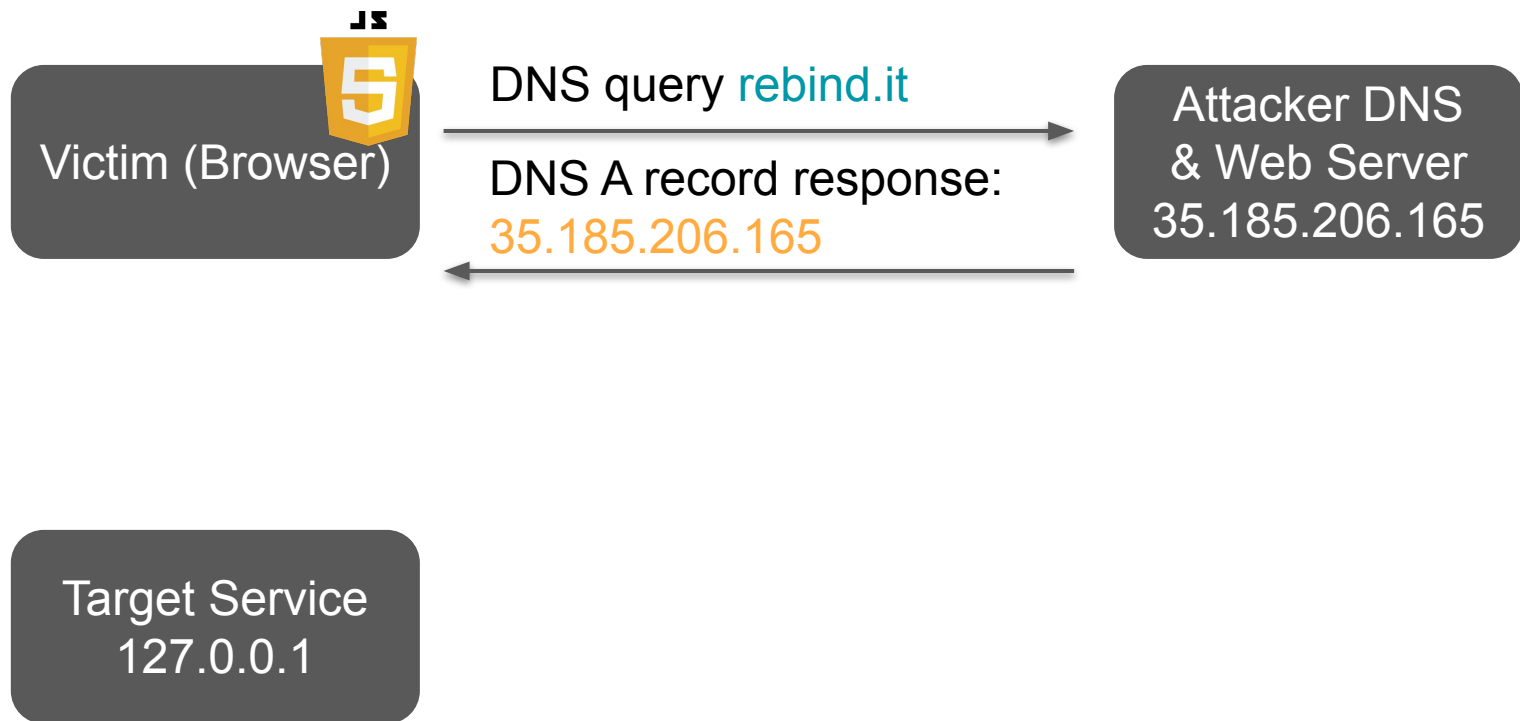
DNS Rebinding Attack Walkthrough



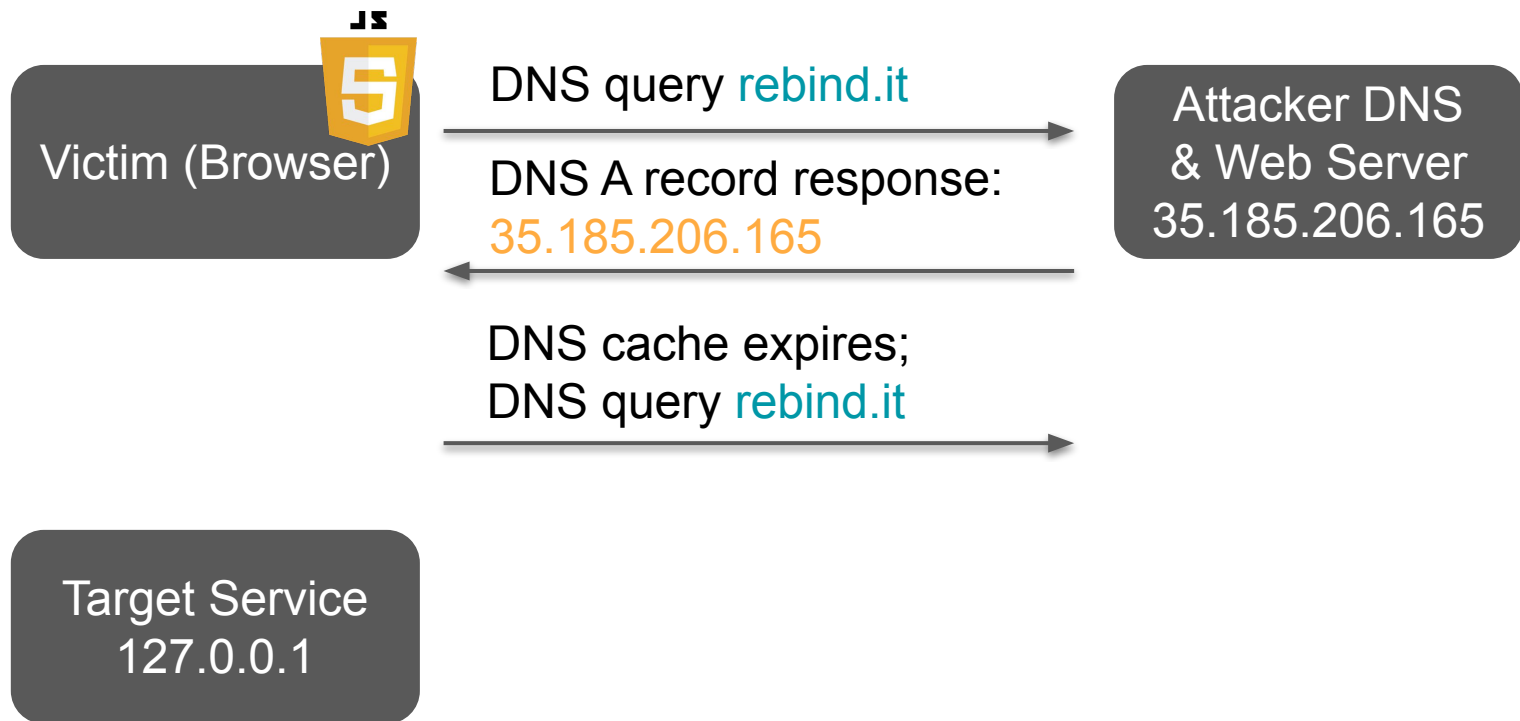
DNS Rebinding Attack Walkthrough



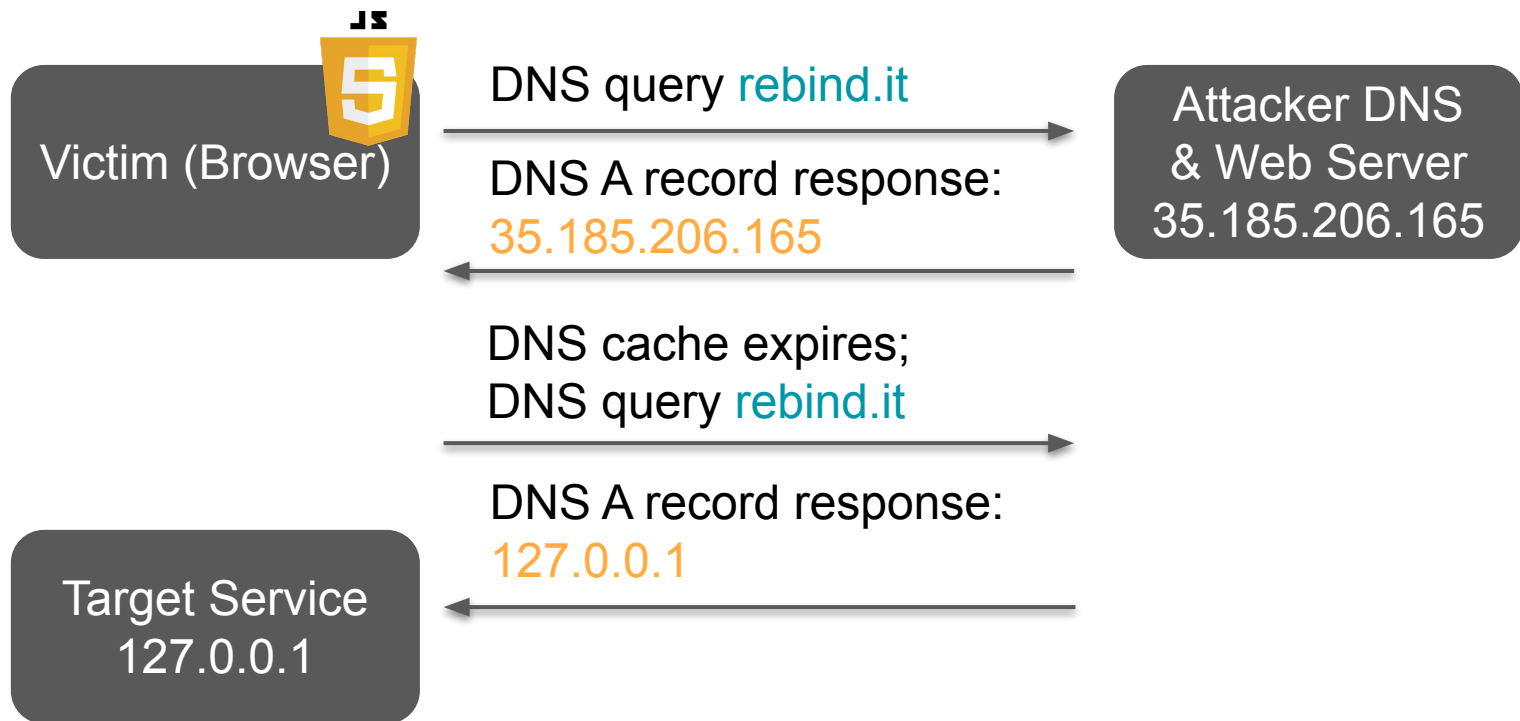
DNS Rebinding Attack Walkthrough



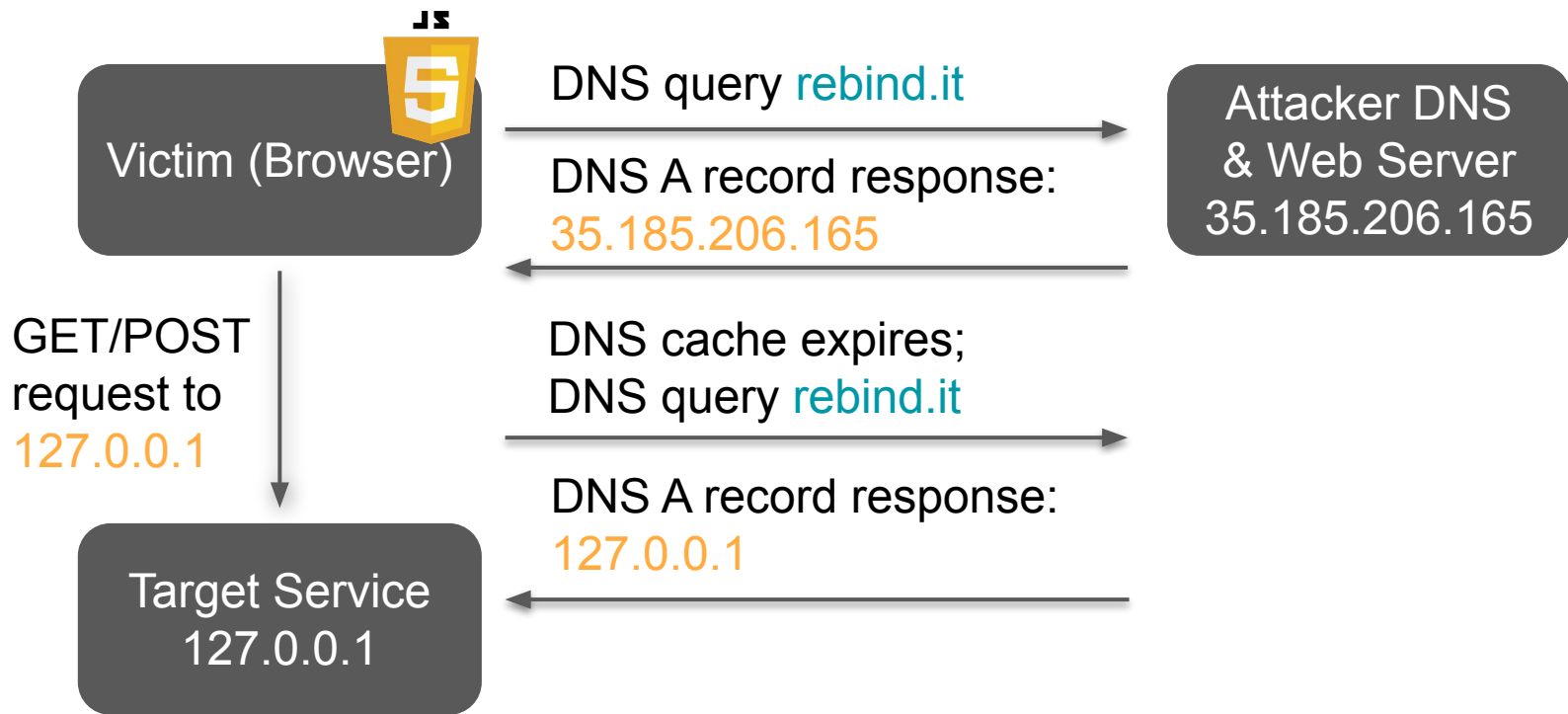
DNS Rebinding Attack Walkthrough



DNS Rebinding Attack Walkthrough



DNS Rebinding Attack Walkthrough



Learning More About the Basics of DNS Rebinding

- [2017] DEF CON 25 - Luke Young - There's no place like 127.001: Achieving reliable DNS rebinding (https://youtu.be/Q0JG_eKLcws)
- [2010] Black Hat USA & DEF CON 18 - Craig Heffner - How to hack millions of routers (<https://youtu.be/VAaqABpjiUQ> / <https://youtu.be/Zazk0plSoQg>)
- [2007] 24C3 - Dan Kaminsky - DNS Rebinding And More Packet Tricks (<https://youtu.be/YwbpnZe74ds>)
- [2018] NorthSec - Danny Cooper & Allan Wirth - Homeward Bound: Scanning Private IP Space with DNS Rebinding (<https://youtu.be/9iSvAS6ldiM>)
- [2013] WOOT '13 - Yunxing Dai, Ryan Resig - FireDrill: Interactive DNS Rebinding (<https://www.usenix.org/conference/woot13/workshop-program/presentation/Dai>)
- [2007] Stanford University - Protecting Browsers from DNS Rebinding Attacks (<https://crypto.stanford.edu/dns/>)

iOS Demo: DNS rebinding in 5 s (cache flooding)

10:50 PM Fri Jul 5

Not Secure — rebind.it

Singularity of Origin DNS Rebinding Attack

This attack typically takes ~1 min to work. This duration can be reduced to ~3s with the appropriate options. Check the [documentation](#). Try the new, experimental HTTP port [scanner](#). Test the automatic identification of vulnerable services on your network upon visiting this [page](#).

Attack Host Domain:

Attack Host: Target Host:

Target Port: [Request New Port](#)

Attack server listening on: 4000,3000,8080,95...
listening on the same port. The "Request New Port" line argument.

Attack Payload: [Start Attack](#)

Rebinding Strategy: [Request New Strategy](#)

Interval: [Request New Interval](#)

Flood DNS Cache: ☒ Attempt flushing the browser DNS cache. Successfully tested on Chrome.

Index Token: The attack uses this string to recognize whether it is accessing the attacker or target host. it must be placed in the index page of the attacker web server.

WS/Proxy Port: TCP port on which Singularity listens to handle websockets and proxy operations.

Please wait for DNS cache entries to expire.

Attack Successful from rebind.it.

Origin:
http://s-35.185.206.165-
router.asus.com-637234560-fs-e.d.rebind.it.

Target home page contents:
<HTML><HEAD><script>top.location.href=/
Main_Login.asp';</script>
</HEAD></HTML>

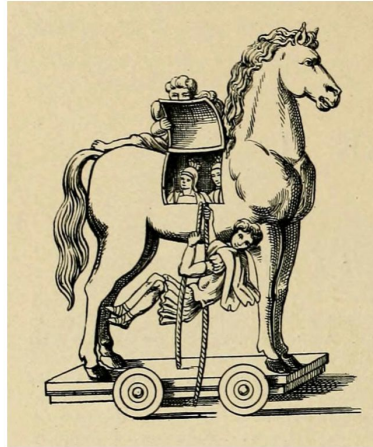
[Close](#)

DNS Rebinding Attack: Building on Reliable Foundations

You Visit a Completely Innocuous Looking Website

Adventures in Equestrianism

THOROUGHBRED HORSE RACING | STEEPLECHASING | AMERICAN QUARTER HORSE | ENDURANCE RIDING | RIDE AND TIE



Phasellus iaculis dui vel lacus pretium, sed tincidunt sem feugiat. Class aptent taciti sociosqu ad litora torquent



Planning Your Next Horse Riding Holidays

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris fermentum lorem tortor, sed feugiat sapien vestibulum a. Pellentesque vel massa in ipsum fermentum pellentesque. Nunc a leo eu risus condimentum dignissim. Nam elementum vitae tellus et porta. Nunc hendrerit cursus elit, nec condimentum dui mattis ut. Quisque malesuada sem in massa portitor, non vehicula lacus porta. Proin suscipit laoreet pharetra. Phasellus bibendum turpis a orci imperdiet, vel suscipit nulla pellentesque. Proin tristique orci a augue convallis, eu gravida mauris mollis. Aliquam felis lectus, efficitur a libero eu, tempor eleifend elit. Quisque sit amet urna vel justo blandit cursus. Proin ligula purus, scelerisque a est nec, ultrices lacinia ipsum. Phasellus ultricies augue non tellus dictum, ut volutpat lacus commodo. Aliquam scelerisque scelerisque sagittis. Pellentesque eget eleifend orci, eget portitor libero.

Nunc facilis elit ex, at tristique erat commodo id. Aliquam vel magna velit. Duis convallis quis ipsum id viverra. Duis magna nulla, hendrerit nec nisi quis, ornare varius metus. Ut nunc nunc, tristique at scelerisque non, consectetur non mi. Proin sapien dolor, commodo at arcu id, portitor congue nulla. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vivamus pellentesque eleifend arcu, sit amet porta erat pulvinar vitae. Quisque laoreet blandit



A Singularity of Origin Production | 2019

Malicious JavaScript Code Downloaded.

Sending First DNS Query...

What's in a Query?

Example DNS query from a browser to Singularity DNS server:

s-35.185.206.165-127.0.0.1-3504134790-fs-e.d.rebind.it

- (s)tart
- **35.185.206.165**: Attacker Host IP Address
- **127.0.0.1**: Target Service IP Address or Name
- **3504134790**: Session ID
- **fs**: DNS Rebinding Strategy - “first then second” IP address.
- (e)nd
- **d.rebind.it**: Attack Host Domain

Removing HTTP Performance Enhancing Techniques That Impede DNS Rebinding

HTTP Caching - We want the browser to get fresh copies of resources.

```
423 func (d *DefaultHeadersHandler) ServeHTTP(w http.ResponseWriter, r *http.Request) {  
424     w.Header().Set("Cache-Control", "no-cache, no-store, must-revalidate") // HTTP 1.1  
425     w.Header().Set("Pragma", "no-cache") // HTTP 1.0  
426     w.Header().Set("Expires", "0") // Proxies  
427     w.Header().Set("X-DNS-Prefetch-Control", "off") //Chrome
```

Keep-Alive - We don't want the browser to stick to the attacker's server.

```
758 // drop browser connections after delivering  
759 // so they dont keep socket alive and facilitate rebinding.  
760 httpServer.SetKeepAlivesEnabled(false)
```


TTL Values

1st query

```
$ dig +noall +answer s-35.185.206.165-127.0.0.1-123-fs-e.d.rebind.it  
s-35.185.206.165-127.0.0.1-123-fs-e.d.rebind.it. 0 IN A 35.185.206.165
```

2nd query

```
$ dig +noall +answer s-35.185.206.165-127.0.0.1-123-fs-e.d.rebind.it  
s-35.185.206.165-127.0.0.1-123-fs-e.d.rebind.it. 0 IN A 127.0.0.1
```

Why not 1 second? We hoped 0 second would break stuff[1]. It did not so far, as far as we know, it is a legitimate value[2].

[1] <https://mark.lindsey.name/2009/03/09/never-use-dns-ttl-of-zero-0/>

[2] <https://tools.ietf.org/html/rfc2181#page-10>

How Do We Know We've Successfully Rebinded?

Two ways to differentiate the attacker server from the target service:

```
$ curl -v http://s-35.185.206.165-127.0.0.1-3504134792-fs-e.d.rebind.it:8080/
```

```
(...)
```

```
HTTP/1.1 200 OK
```

```
X-Singularity-Of-Origin: t # Custom HTTP Header
```

```
(...)
```

```
<!--thisismytesttoken--><!doctype html><title>(...) # Index Token
```

Randomness and Catering for Potential Interference

IPS/IDS/other interference via spurious DNS queries

- **Challenge:** the environment IPS/IDS may make their own queries to the attacker domains in addition to the target, resulting in incorrect DNS/out of sequence DNS answers for the target.
- **Solution:** Use the random DNS rebinding strategy.
- Slower technique in general (but you could get lucky!).

The Need for Speed: DNS Rebinding in 3 Seconds

Implementation Details Matter!

DNS Rebinding **speed varies** based on a number of factors:

- **OS implementation:** Windows or Unix-like (Linux, macOS)
- **Browser vendor:** IE/Edge, Firefox, Chrome/Chromium Edge, Safari
- **Target specification:** local, remote
- **External factors:** Spurious DNS queries e.g. presence of IPS/IDS

DNS rebinding may take **40+ min** or **~3s** on Edge depending on the strategy!

We can automatically fingerprint to optimize for speed in some conditions. More on this later!

Multiple Answers Rebinding Strategy with Targets 127.0.0.1 / 0.0.0.0

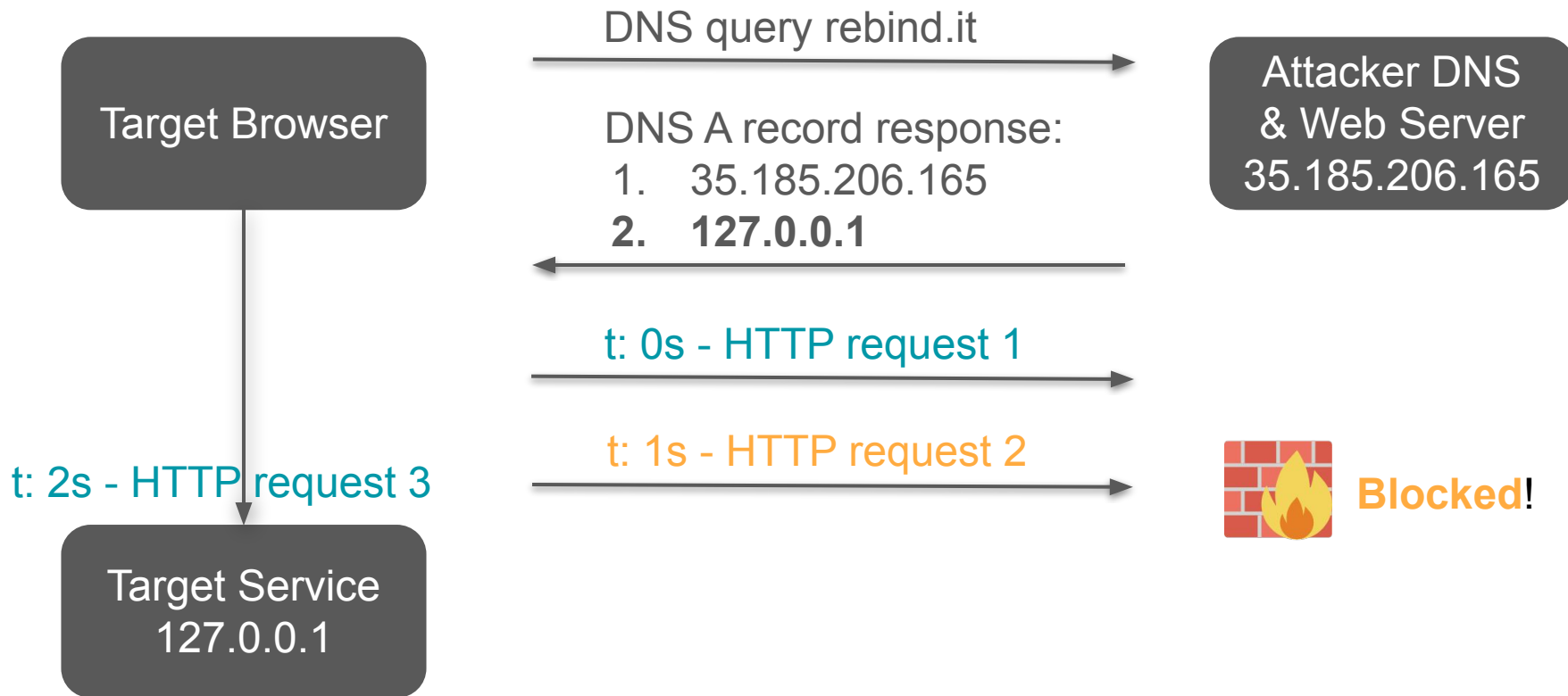
The time-varying (Singularity's "first then second") DNS rebinding technique is **~60 seconds** on all browsers except IE/Edge.

Multiple answers (respond with attacker and target addresses, then block attacker with ephemeral firewall rule) **is near instantaneous**. 127.0.0.1 works on Windows only.

We got it to work on Unix-y machines (Linux, macOS) with "0.0.0.0".

➡ **Solid and fast DNS rebinding against all "localhost" services.**

Multiple Answers Rebinding Strategy Illustrated



DNS Cache Flooding

Multiple Answers works well for the **loopback** (0.0.0.0 or 127.x.x.x) interface - inconsistent results for other target specifications.

On Google Chrome or Safari/iOS platforms, when flooding the **DNS cache with 1K+ queries** for which we receive valid answers, we observe DNS rebinding time with the time varying attack technique (first then second) of **5 to 40 seconds**, a substantial progress over the average of ~60 seconds.

Flooding the cache is performed in a web worker.

button is only available when the server is started with the "- dangerouslyAllowDynamicHTTPServers" command line argument.

Attack **Simple Fetch Get**

Payload

Start Attack

Toggle Advanced Options

Rebinding Strategy **First then second** Read the docs if changing from the default value to ensure that the attack will succeed.

Interval **1** How long to wait between attempts in seconds

Flood DNS Cache ☒ Attempt flushing the browser DNS cache. Successfully tested on Chrome.

Index Token **thisismytest** The attack uses this string to recognize whether it is accessing the attacker or target host. It must be placed in the index page of the attacker server.

WS/Proxy Port **3129** TCP port on which Singularity listens to handle websockets and proxy operations.

Please wait for DNS cache entries to expire.

Simple Fetch Get






target: 127.0.0.1:8080, session: 861969865, strategy: fs. DNS rebinding successful!

The screenshot shows the Network tab of a web browser. The filter is set to 'All'. The list of requests includes:

Name	Status	Type	Initiator	Size	Time	Waterfall
n2708281003.rebind.it	200 OK	fetch	flushdns	343 B	18.2...	
n2708281000.rebind.it	200 OK	fetch	flushdns	343 B	18.2...	
s-35.185.206.165-127.0.0.1-861969...	200 OK	fetch	payload.js:66	343 B	18.3...	
n2708281005.rebind.it	200 OK	fetch	flushdns	343 B	18.4...	
n2708281008.rebind.it	200 OK	fetch	flushdns	343 B	18.4...	
n2708281005.rebind.it	200 OK	fetch	flushdns	343 B	18.4...	
n2708281009.rebind.it	200 OK	fetch	flushdns	343 B	18.4...	
n2708281006.rebind.it	200 OK	fetch	flushdns	343 B	18.4...	
n2708281007.rebind.it	200 OK	fetch	flushdns	343 B	18.4...	
n2708281011.rebind.it	200 OK	fetch	flushdns	343 B	18.6...	
n2708281010.rebind.it	200 OK	fetch	flushdns	343 B	18.6...	
n2708281012.rebind.it	200 OK	fetch	flushdns	343 B	18.6...	

1029 requests | 398 KB transferred | 363 KB resources

Speed Measured / Target Definition

Browser	OS	Strategy	Time to Exploit	Fetch Interval	Target Spec
	Windows 10	MA	3 seconds	1 second	127.0.0.1
	Ubuntu	MA	3 seconds	1 second	0.0.0.0
	macOS	MA	3 seconds	1 second	0.0.0.0
	macOS, Ubuntu, Windows	FS+Cache Flooding	15-40 seconds	1 second	Any
	iOS	FS+Cache Flooding	5 seconds	1 second	Any

Protection Bypasses

DNS Rebinding Protection Bypasses

- Singularity can bypass all known DNS rebinding protections:
 - Unbound
 - Dnsmasq
 - pfSense
 - OpenWRT
 - OpenDNS (Cisco Umbrella)
- Common recommendations and default configurations **do not provide complete protection**

Common DNS Protections

Approaches:

- Block RFC 1918 IP addresses
 - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
- Block localhost (127.0.0.0/8)
- Block local (internal) networks
- Block 0.0.0.0/8

Tools:

- Dnsmasq & Unbound widely used
 - pfSense, OpenWRT, home routers (e.g. FRITZ!Box, ASUS)
- Public DNS services
 - OpenDNS: “Block internal IP addresses”: Blocks RFC 1918 IP addresses

Dnsmasq

- `--stop-dns-rebind` : Reject private IP ranges
- `--rebind-localhost-ok` : Exempt 127.0.0.0/8 from rebinding checks
- This blocks RFC1918 addresses, 0.0.0.0/8, and 127.0.0.0/8
- **localhost** is **not blocked**



Unbound

- `private-address`: Configure specific internal IP address range to be blocked
- This blocks RFC1918 IP addresses
- Does **not block** `0.0.0.0`, `127.0.0.1`, and `localhost`

 <https://nlnetlabs.nl/documentation/unbound/unbound.conf/>

bound

e

ext

results

lookup

reads

ite

private-address: *<IP address or subnet>*

Give IPv4 or IPv6 addresses or classless subnets. These are addresses on your private network, and are not allowed to be returned for public internet names. Any occurrence of such addresses are removed from DNS answers. Additionally, the DNSSEC validator may mark the answers bogus. **This protects against so-called DNS Rebinding**, where a user browser is turned into a network proxy, allowing remote access through the browser to other parts of your private network. Some names can be allowed to contain your private addresses, by default all the **local-data** that you configured is allowed to, and you can specify additional names using **private-domain**. No private addresses are enabled by default. We consider to enable this for the RFC1918

DNS Rebinding Protection Bypass #1: 0.0.0.0

- Wikipedia: “0.0.0.0 is a non-routable meta-address used to designate an invalid, unknown or non-applicable target”
- **Fact:** 0.0.0.0 works well on Linux and macOS to access the localhost
- This **bypasses** protections that block DNS responses of 127.0.0.1
- Singularity returns a DNS A record:

```
$ dig s-1.2.3.4-0.0.0.0-474794-fs-e.d.rebind.it
;; QUESTION SECTION:
;s-1.2.3.4-0.0.0.0-474794-fs-e.d.rebind.it. IN A
;; ANSWER SECTION:
s-1.2.3.4-0.0.0.0-474794-fs-e.d.rebind.it. 0 IN A 0.0.0.0
```


DNS Rebinding Protection Bypass #2: CNAME

- What if all internal IP addresses are blocked?
- Canonical Name records (CNAME) map one domain name to another
- We return a CNAME DNS record instead of an internal IP address
 - e.g. wiki.nccgroup.com or jenkins.internal.corp.com
- This **bypasses** protections that block DNS responses of private IP addresses
- The local, internal DNS server will then resolve the CNAME

```
$ dig s-1.2.3.4-wiki.nccgroup.com-123-fs-e.d.rebind.it
;; QUESTION SECTION:
;s-1.2.3.4-wiki.nccgroup.com-123-fs-e.d.rebind.it. IN A
;; ANSWER SECTION:
s-1.2.3.4-wiki.nccgroup.com-123-fs-e.d.rebind.it. 9 IN  CNAME wiki.nccgroup.com.
```

DNS Rebinding Protection Bypass #2a: localhost

- localhost is a hostname that means this computer
- We return a CNAME (Canonical Name) DNS record of “localhost.”
- This **bypasses** protections that block DNS responses of 127.0.0.1

```
$ dig s-1.2.3.4-localhost-123-fs-e.d.rebind.it
;; QUESTION SECTION:
;s-1.2.3.4-localhost-123-fs-e.d.rebind.it. IN      A
;; ANSWER SECTION:
s-1.2.3.4-localhost-123-fs-e.d.rebind.it. 0 IN CNAME localhost.
```

Hook and Control :
interactively browse
the victim's internal
network after DNS
rebinding

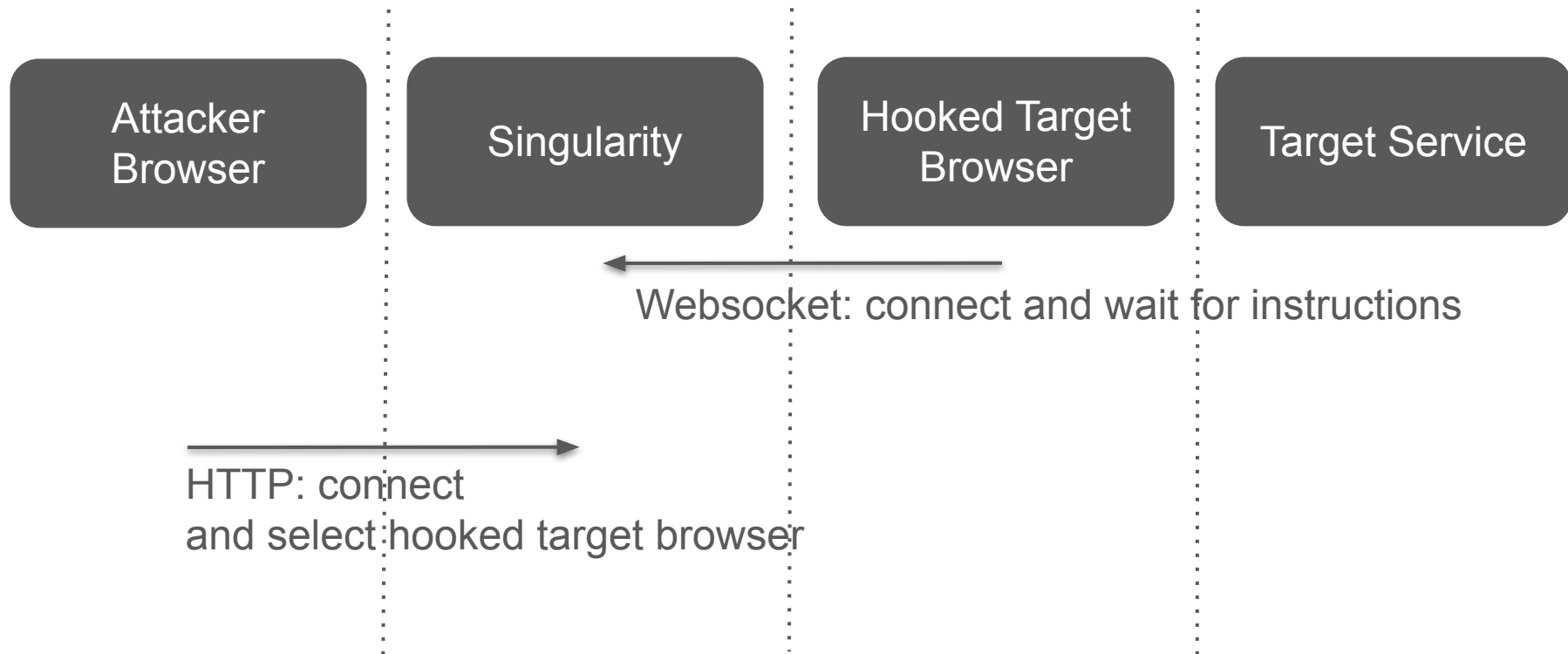
Experimenting with Proxying without an HTTP Proxy

HTTP tools such as BeEF (Browser Exploitation Framework - <https://beefproject.com/>) and FireDrill (<https://www.usenix.org/conference/woot13/workshop-program/presentation/dai>) can use a hooked browser via XSS or DNS rebinding as a gateway to otherwise unreachable networks such as home or corporate environments.

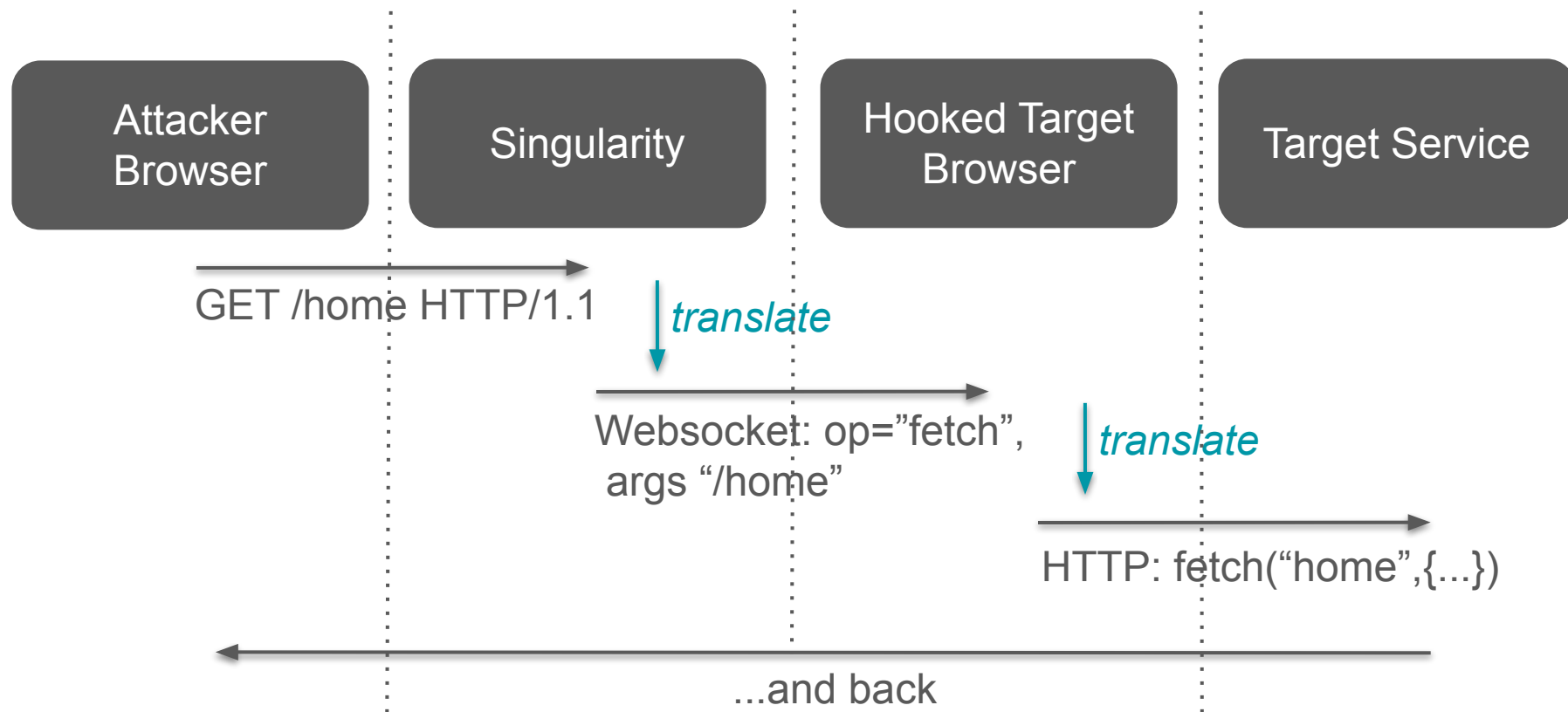
We know that BeEF requires to configure the attacker browser or operating system to use the BeEF HTTP proxy e.g. “<http://beef.attacker.com:3120/>”. We do not know how FireDrill does it since its code is unfortunately not available.

We implemented browsing of services via a hooked browser in Singularity, without requiring the attacker setting up its browser to use an HTTP proxy for fun.

Proxy Architecture



Proxy Architecture



Proxying without an HTTP Proxy

- Customized Golang's RoundTripper <https://golang.org/pkg/net/http/#RoundTripper>
- Using WS plain text protocol to package `fetch()` requests and responses -
Inflate size of data in transit: $\text{len} \approx 4/3$ of $\text{len}(\text{message})$ using base64 encoding.

Attacker Browser - Any user agent: Web browser, `curl`, HTTP inspecting proxy, SQLMap, etc.

Dealing with Split Brains: Syncing the state between the attacker and target's browsers

The initial assumption was that we did not have to care about cookies. Our first test case was Duplicati, a backup application which was vulnerable to DNS rebinding attack and has a web interface listening on localhost.

```
102     this.get_import_url = function(passphrase) {  
103         var rurl = this.apiurl + '/backups/import?x-xsrf-token=' + encodeURIComponent($cookies.get('xsrftoken'));  
104         if ((passphrase || '').trim().length > 0)  
105             rurl += '&passphrase=' + encodeURIComponent(passphrase);  
106  
107         if (this.proxy_url != null)  
108             return this.proxy_url + '?x-proxy-path=' + encodeURIComponent(rurl);  
109         return rurl;  
110     };
```

Oops.

Dealing with Split Brains: Syncing the state between the attacker and target's browsers

Cookies may be used as **CSRF tokens** or other purposes.

For **non HttpOnly** cookies:

- Read them from target browser => transmit to the **Singularity server**
- Singularity **sets them on the attacker browser for the target domain** (the DNS domain constructed by Singularity).

For **HttpOnly** cookies:

- We don't care - they can't be read by JS so they cannot be used by JS.
- The target browser handles (receives and transmits) them for us.

Dealing with Split Brains: Syncing the state between the attacker and target's browsers

To be able to read cookies from a response to a `fetch()` request, **you must pass the option `{credentials: 'include'}` to the `fetch()` request.**

If the application requires HTTP authorization (`WWW-Authenticate`), then we must forego completely about passing cookies, unless we know the credentials in advance and pass them without being challenged for authentication.

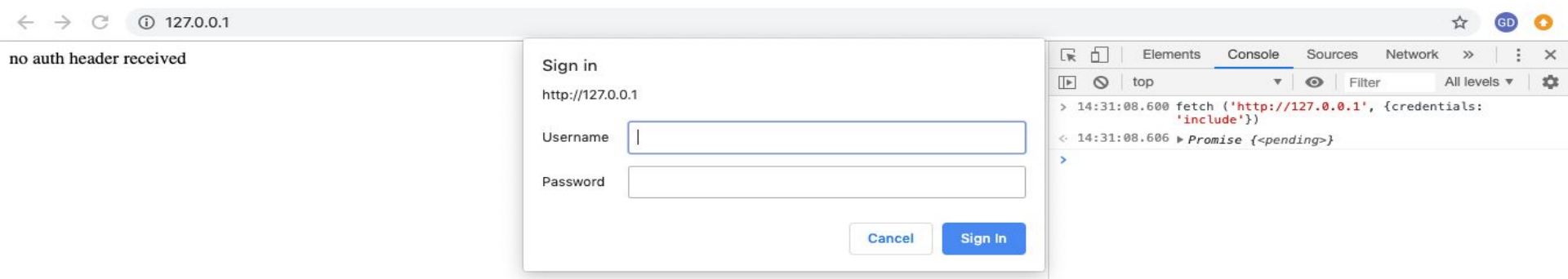
Why? Let's test in the next slides 

Dealing with Split Brains: Syncing the state between the attacker and target's browsers

`fetch ('http://127.0.0.1', {credentials: 'include'})`

→ Authentication dialog box popup in victim's browser

→ Victim 🤔

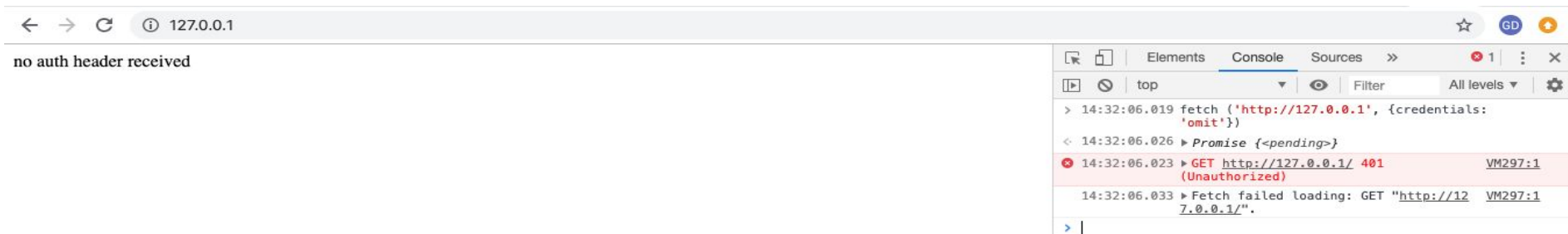


Dealing with Split Brains: Syncing the state between the attacker and target's browsers

`fetch ('http://127.0.0.1', {credentials: omit})`

→ No authentication dialog box

→ Victim 🙄



Demo 2: Hook & Control

The image shows a desktop environment with two windows. The background window is a Chromium Web Browser displaying a web page with the URL `rebind.it/manager.html`. It contains two input fields: "Index Token" with the value `thisismytesttoken` and "WS/Proxy Port" with the value `3129`. Below these fields is a message: "Please wait for DNS cache entries to expire." A terminal window in the foreground shows the output of a command: `target: 0.0.0.0:8080, session: 536786693, strategy: ma. DNS rebinding successful!`. The foreground window is a Firefox Web Browser displaying the Jenkins dashboard. The dashboard has a dark header with the Jenkins logo and a sidebar with links: "New Item", "People", "Build History", "Manage Jenkins", "Credentials", and "New View". The main content area says "Welcome to Jenkins!" and includes a button that says "Please create new jobs to get started."

Activities Chromium Web Browser

Singularity of Origin DNS x +

Not secure | rebind.it/manager.html

Index Token thisismytesttoken

WS/Proxy Port 3129 TCP port on

Please wait for DNS cache entries to expire.

Hook and Control

target: 0.0.0.0:8080, session: 536786693, strategy: ma. **DNS rebinding successful!**

Activities Firefox Web Browser Jul 7 22:03

Dashboard [Jenkins] - Mozilla Firefox

Hooked WS Clients x Dashboard [Jenkins] x +

536786693.rebinder.rebind.it:3129

Jenkins

Jenkins

- New Item
- People
- Build History
- Manage Jenkins
- Credentials
- New View

Welcome to Jenkins!

Please create new jobs to get started.

Scanning for Vulnerable Hosts Services

Old World and Cool Hacks (Embedding Images, Measuring Requests Response Time)

Many astute attempts to replicate `nmap` behavior without the power of raw sockets.

Often unreliable / do too much for our purposes e.g. we don't care about whether a SSH port is open or not.

Does it speak HTTP? We are interested in DNS rebinding and DNS rebinding deals with the HTTP protocol only (so far).

Leveraging Modern APIs and Focusing on What Matters (**fetch**, **abort**, HTTP only)

```
11 function scan(targetdata, duration) {  
12     let sendDate = new Date().getTime();  
13     var controller = new AbortController();//NO IE support  
14     var signal = controller.signal;  
15     timeout(duration, fetch(`http://${targetdata.address}:${targetdata.port}/`, {  
16         mode: 'no-cors',  
17         credentials: 'omit',  
18         signal  
19     }),controller)
```


Leveraging Modern APIs and Focusing on What Matters (fetch, abort, http only)

Solution:

- Wrap in a web worker - distribute scan targets across 4 web workers
- **fetch()** resource headers with timeout (300 ms) - Don't bother with resp. body
 - Timeout drives how fast scans can go - how long we hang, waiting for a response
 - When an unhandled protocol (e.g. SSH) or when a port is firewalled (No TCP RST packet)
- **Fast** for: open HTTP ports, closed ports
- **Slower** for: firewalled ports, slow HTTP services & possibly specific protocols
- Pro-tip:
 - Use a lower timeout when scanning LAN and/or fast HTTP services.
 - Use higher timeouts when scanning different networks e.g. across VPN.

Leveraging Modern APIs and Focusing on What Matters (fetch, abort, http only)

Other bits and pieces:

- Use the classical WebRTC IP address leak when available to obtain the IP address of the machine and derive a subnet (Chrome, Firefox).
- `fetch ('http://127.0.0.1', {credentials: omit})` → No authentication dialog box → Victim 😊 . Didn't we cover this before? 🤔
- Considering performing a second scan pass for potentially slower services (Singularity implementation TODO list).

Automation: Service
Detection &
Exploitation and
Orchestrating all the
Above

Auto Detection and Exploitation of All Things Accessible by the Target Web Browsers

- “Autoattack.html” automation and orchestration sample file
- Customizable
- Permits to leverage all features of Singularity
 - Specific exploitation payload or auto-selection of payload to deliver based on detected service
 - Targets selection + optional detection
 - Ports selection + optional port scanning
 - Default DNS strategy selection + optional detection of best strategy to use in specific cases
 - Various options such as flooding DNS cache, visibly hiding activity etc.
- Future work: more auto-optimization so you don't have to read the extensive wiki (<https://github.com/nccgroup/singularity/wiki>) .



Choosing the Right Targets 0.0.0.0, "localhost", CNAMEs, Weak Host Model

- Mix and match different specifications of same target for reliability, security controls bypass and speed (“0.0.0.0”, “localhost”, “127.0.0.1”).
- Find and use the external IP address to exploit routers / Wifi APs’ internal network facing administration interface (weak end system model - <https://www.defcon.org/images/defcon-18/dc-18-presentations/Heffner/DEFCON-18-Heffner-Routers-WP.pdf>)
- Do some homework using **OSINT** - try to determine the local corporate domains, use a dictionary of service names and specify them as CNAMEs e.g. [jenkins.internal.corp.com](#). This is likely to pay off.



Service Detection

Singularity comes with a number of attack payloads targeting services such as Chrome DevTools Remote Debugger, Amazon AWS instance metadata, Ruby on Rails etc.

We recently augmented a number of its payloads with a service detection routine.

```
35      // Invoked to determine whether the rebinded service
36      // is the one targeted by this payload. Must return true or false.
37      async function isService(headers, cookie, body) {
38          return headers.get("X-Jenkins") !== null;
39      }
```

Selecting the “**automatic**” payload will instruct Singularity to detect the service and **deliver the appropriate attack!**

Concluding Remarks:
There is Only One
HTTP Origin

How To Protect From DNS Rebinding:

Use DNS blacklists

Use DNSSEC?

Use this DNS service provider

Use this router/appliance/IPS!

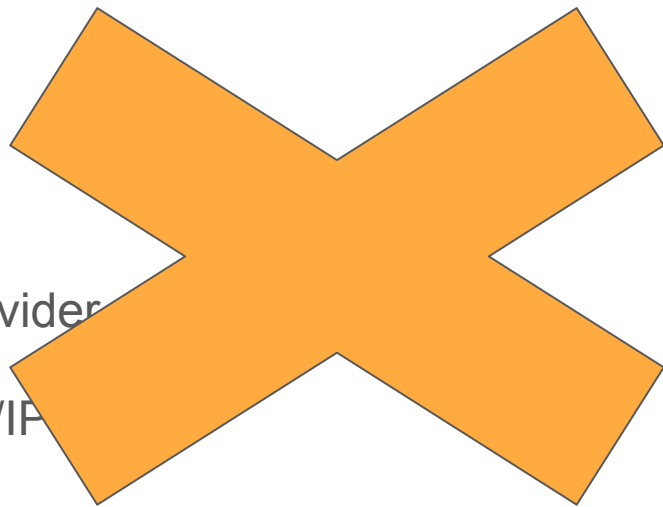
How To Protect From DNS Rebinding: Common Wisdom is Not Enough

Use DNS blacklists

Use DNSSec

Use this DNS service provider

Use this router/appliance/IP



...Do you understand all the subtleties of DNS rebinding? *And no, DNSSec does not help at all!*

How To Really Protect From DNS Rebinding

Use TLS on all services, external and internal including localhost
(<https://blog.filippo.io/mkcert-valid-https-certificates-for-localhost/>, <https://github.com/FiloSottile/mkcert/releases>).

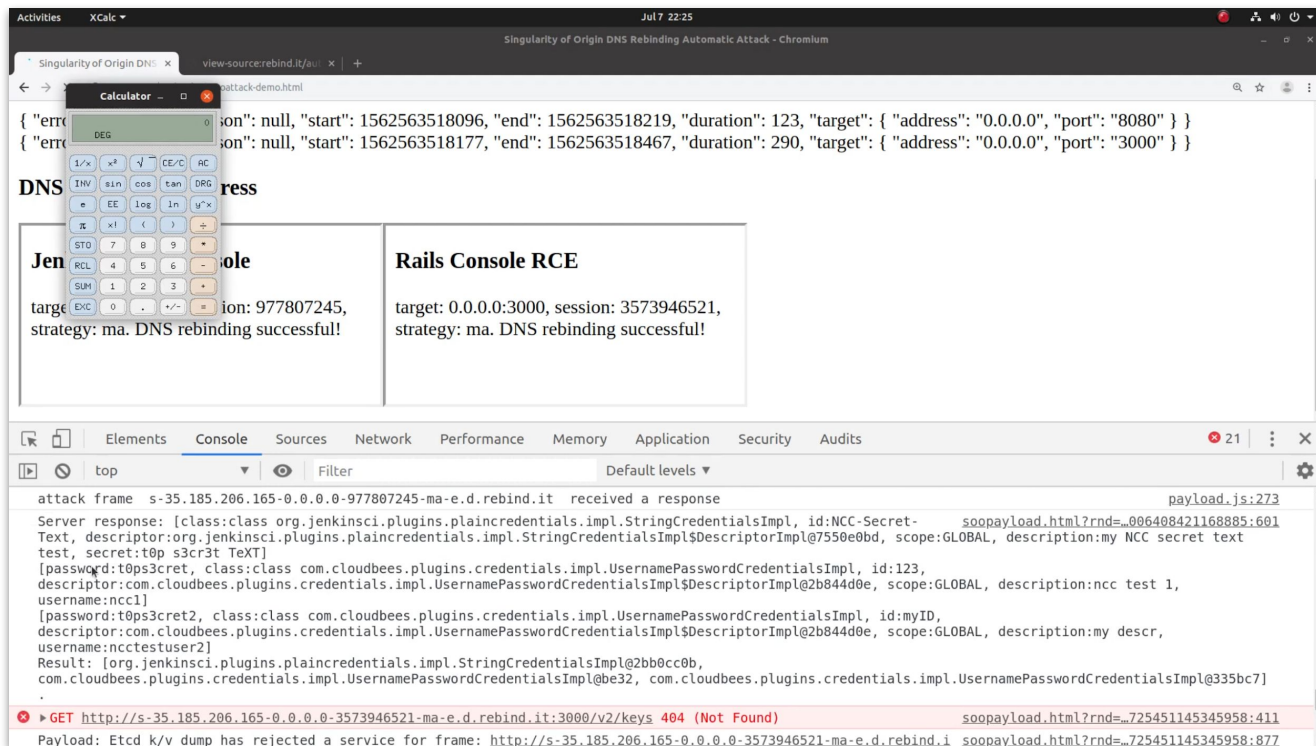
Always use authentication.

Validate the **Host** header of HTTP requests for correct values e.g. 127.0.0.1 (whitelisting).

The future? <https://wicg.github.io/cors-rfc1918/>

Demo 3: Automation

1. Portscan
2. Rebind in 3s
3. Auto-detect services
4. Exploit



Thank You

- Get the Slides+Notes at: https://bit.ly/Singularity_Defcon27
- Get Singularity of Origin at <https://github.com/nccgroup/singularity>
 - **DNS server** to rebind DNS names to IP addresses
 - **HTTP server** to serve HTML pages and JavaScript code to targets and to manage the attacks
 - **Sample attack payloads**: Chrome DevTools, Jenkins, & many more
 - Supports DNS **CNAME** to evade DNS filtering solutions
 - A simple, fast and efficient **HTTP port scanner** to identify vulnerable services
 - **Attack automation**: completely automate the scanning and exploitation
 - **Hook & control** to exploit victim browser as HTTP proxy to access internal network resources
- Contact us:
 - gerald.doussot@nccgroup.com
 - roger.meyer@nccgroup.com

