Root Services Blog, Vol. I, No. 4, 2015
Copyright Root Services, 2015

DNS Rebinding Overview

Alexander Chopra[1]*, Michael Kaufman[2]

Abstract
This article provides an overview of DNS rebinding. A detailed explanation of DNS rebinding is provided and the associated problems with modern browsers are explored. A proof of concept experiment is explored and conclusions are made about the current state of browsers in relation to security surrounding DNS rebinding and how to progress.

Keywords
DNS Rebinding — Same-Origin Policy — DNS Pinning

[1] *Research and Development, Root Services, Rockville, Maryland, United States*
[2] *Research and Development, Root Services, Rockville, Maryland, United States*

Contents

Introduction 1
1 DNS Rebinding 1
1.1 Same-Origin Policy 1
1.2 Example of DNS Rebinding 1
1.3 Benefits 2
1.4 Proof of Concept 2
1.5 DNS Pinning 2
1.6 Possible Defense 2
2 Conclusion 3
Acknowledgments 3
References 3

Introduction

DNS rebinding is a type of cyber-attack that bypasses protections implemented within browsers to potentially circumvent firewalls as well as gain access to large numbers of IP addresses. DNS rebinding attacks are generally defended against through DNS pinning; the process of locking an IP address to the value that is the first received DNS response. The method of DNS pinning has become generally ineffective in modern browsers because of the introduction of plugins and third party applications that run alongside the browser. The same-origin policy which attempts to differentiate between different sites, protecting each from one another is seen to be easily bypassed in modern browsers, again bringing up the issue of DNS rebinding as a real threat in today's environment. This paper will explain at a high level the issues regarding the ineffective nature of DNS pinning and how DNS rebinding is still possible and will continue to be a problem going forward.

1. DNS Rebinding

1.1 Same-Origin Policy

The intent of the same-origin policy is to only permit scripts that run on web pages that originate from the same site to access other internal resources. Simply put, the same-origin policy attempts to restrict that if a script is loaded from one location, its ability to interact with resources outside of that original origin is limited or denied. As a result of the same-origin policy, the possibility for fraudulent data to be transmitted across multiple sites is lessened. In theory the same-origin policy should prevent scripts from running form hosts in which the request did not originate on.

1.2 Example of DNS Rebinding

To provide an example of the process, an attacker would register the domain name example.com and attract visitors to that domain which would contain a malicious script (JavaScript, Flash, etc.). The attacker would change the time-to-live (TTL) of the DNS requests to example.com to something very small. The attacker would also alter the result from the DNS requests to example.com to their own attack server. The attacker would rebind example.com to their target address, creating a situation in which the browser believes that both example.com and the target address both belong to the same origin. As a result of this, the attacker would be able to utilize and modify that script to gain access to otherwise internal resources of their target. The attacker does not necessarily need to know

the IP address of the target, as they could just make multiple attempts at the internal addresses. Assuming the target is company.com, an attacker could guess internal.company.com (and other possible internal sub-domains) until they hit an internal address that is valid. Once a machine at the internal address has been located, the attacker could crawl the intranet connected to that internal machine. Often, the internal nature of the computer suggests that sensitive information is placed on these resources that are not connected to the public internet, creating a potentially valuable target for a would-be attacker.

1.3 Benefits

An attacker could find benefit from launching a DNS rebinding attack on an unsuspecting client computer for many reasons including monetary gain or even gaining access to sensitive information on a corporate network. DNS rebinding allows an attacker to both read and write on network connections. The cost to launch a DNS rebinding attack to a potential attacker is very cost-effective and can be done with relative ease. To launch such an attack, an attacker would only need a server they own (or possibly have hijacked) and time. In comparison with other methods of IP hijacking, it could be argued that because the attacker would be able to legitimately sign all DNS records that are provided by the attacker's DNS server utilized in the attacks, it is a more legal method of committing these sorts of attacks[1]. IP hijacking is a possible outcome of the DNS rebinding attack and an attacker could possibly have access to a large number of IP addresses for their own use. Click fraud is one of the ways in which an attacker could generate a large amount of income through utilizing those IP address to create impressions on their own advertisements in a believable manner. Those IP addresses could also be utilized for mass mailing campaigns to send spam mail to a large number of recipients before being blacklisted.

1.4 Proof of Concept

In a proof of concept demonstration, Stanford spent 50 cents per 1000 impressions, costing them about 10 dollars per day for a duration of three days. The proof of concept demonstration resulted with about 45,000 unique IP addresses for rebinding attempts; 60 percent of the attacks were successful[1]. Subversion of DNS pinning in this regard is one of the most cost-effective methods to exploit these IP addresses and as a result is something that should be taken seriously.

1.5 DNS Pinning

Originally, the concept of DNS pinning was introduced to attempt to combat possible loopholes in the system to prevent DNS rebinding attacks. DNS pinning caches the resolved IP address of the hostname attempting the connection into its own database, disregarding any TTL specification[1]. The concept of DNS pinning began to show its flaws with the introduction of various extensions and plugins for browsers. These various third party plugins, such as Adobe Flash, Microsoft Silverlight or Java to name a few examples, hold their own pinning databases, which creates new vulnerabilities into the DNS pinning system. Firewalls can be circumvented in this manner through DNS rebinding such that the firewall would allow access and not block the threat presented properly or even at all. It is now possible to introduce a situation in which the browser pins one resolved IP address into its database and a plugin for that browser pins a differing IP address into its own database, allowing for an attacker to "read and write data directly on sockets to a host" as a result of there being multiple distinct databases instead of just one[1]. Some plugins such as Adobe Flash Player, attempt to protect against DNS rebinding by requiring a response back from the server with a XML policy ensuring that the access is OK. The attacker can circumvent this easily by responding back to the client with their own XML allowing access from all domains and ports, giving the impression back to the client that all is well and this is not an issue. Other plugins restrict certain types of access but are able to be circumvented through multi-pin vulnerabilities, allowing direct socket access to the attacker[1].

1.6 Possible Defense

Many possible defenses to DNS rebinding attacks have been proposed that will protect current generation browsers and plugins as well as into the future. One of the most common defense to these types of attacks can be an open source piece of software that acts as a resolver that prevents an external hostname to resolve into an internal address[1]. An example of the resolver proposed is called "dnswall" and is very lightweight and easily able to be deployed to assist in preventing these types of attacks. The second possible defense is patches to third party plugins to prevent the vulnerabilities brought up because of the multiple DNS pinning databases that are kept, introduced by plugins and extensions. The third proposed defense against DNS rebinding attacks is a complete overhaul of the DNS system as well as more secure methods of DNS pinning to avoid these issues in the future[1]. Of the proposed solutions for defending against DNS rebinding in a more effective manner, none of them seem to be viable because of the lack of uniformity of the change or the extensiveness of what would be required. Flash player, for example, was focused upon heavily throughout the Stanford study as an often used deployment method of this type of attack, but the solution presented requires too much from third parties to come to fruition it seems. The ideas presented focus on deployment of patches by these third party companies to fix many of the vulnerabilities that are brought up. Often this is not a priority of these plugin developers and without their involvement directly in trying to patch these security issues, the deployment of such a patch from these third parties would most likely come on a schedule that would be too lengthy to keep up with developments or the third party wouldn't be interested in deploying the patches to begin with, rather focusing on further development of their plugin. Google Chrome's App store as well as Mozilla Firefox's App store list extensions on their own website, allowing for individuals to quickly and easily list extensions publicly for anyone to download. The

ease of creation and deployment of plugins and extensions make many of the proposed solutions to patching third party plugins unreasonable as there are too many developers that would need to make these changes. A developer focusing on implementing a DNS rebinding attack could easily create an extension mimicking another extension without a proposed patch and gain a significant amount of user base before being removed from the listing; at which point re-listing would be just as easy. The proposed changes to the DNS system as a whole would most likely never come to fruition because of the massive amount of changes to the system that would be needed to properly implement the system. As previously noted, backwards compatibility will always be needed to ensure that users are not left behind, resulting in the slowness of deployment of some of these fixes from a global perspective. Others take a differing approach to solving the issue of DNS rebinding and multi-pin vulnerabilities by placing the majority (if not all) of the blame on the security design of the browser itself[2]. The change from the browser level in regard to securing itself from these types of attacks from its plugins, could be forceful in nature and would require developers to focus strongly on re-configuring their plugins and extensions to act accordingly in a secure manner rather than attempting to patch everything which is unreasonable. The inconsistency seen within the browser itself is analyzed in other scholarly articles which agree with the assertion that the browser itself is at fault solely and the plugins multi-pin vulnerabilities are a side-effect of this rather than the problem itself[3]. Reis argues that because of the complex nature of the internet and the desire for browsers to accommodate all, DNS rebinding became a problem rather than the various developments of the internet creating a problem over time with this issue.

2. Conclusion

It is unreasonable to depend on third party providers to provide a comprehensive solution to an existing problem. The motivation behind third party developers is minimal in implementing the changes that are required to correct the problem, and there will always be a certain number of developers refusing to update or taking advantage of the issue for purposes of exploitation. Users should be aware of the issues of DNS rebinding in their daily usage as well as when on corporate resources because of the existing threat that can be seen still exists as a result of this attack possibility. The general defense of DNS pinning most often fails against modern DNS rebinding attacks with current browsers because browsers have shifted to incorporate applications within the browser itself, creating more security holes.

Acknowledgments

Stanford for providing a proof of concept demonstration on DNS rebinding in a lab environment.

References

[1] C. Jackson, A. Barth, A. Bortz, W. Shao, and D. Boneh. Protecting browsers from dns rebinding attacks. *ACM Transactions on the Web*, 3, 2009.

[2] C. Grier, S. T. King, and D.S. Wallach. How i learned to stop worrying and love plugins. *In Web 2.0 Security and Privacy*, 1, 2009.

[3] C. Reis, S. D. Gribble, and H. M. Levy. Architectural principles for safe web programs. *In Sixth Workshop on Hot Topics in Networks*, 1, 2007.