Remote Attacks Against SOHO Routers

08 February, 2010
Craig Heffner

Abstract

This paper will describe how many consumer (SOHO) routers can be exploited via DNS re-binding to gain interactive access to the router's internal-facing Web based administrative interface. Unlike other DNS re-binding techniques, this attack does not require prior knowledge of the target router or the router's configuration settings such as make, model, IP address, host name, etc, and does not use any anti-DNS pinning techniques.

Confirmed affected routers include those manufactured by Linksys, Belkin, ActionTec, Thompson, Asus and Dell, as well as those running third-party firmware such as OpenWRT, DD-WRT and PFSense.

Existing Attacks and Their Limitations

Many SOHO routers contain vulnerabilities that can allow an attacker to disrupt or intercept network traffic. These vulnerabilities include authentication bypass, cross-site scripting, information disclosure, denial of service, and perhaps the most pervasive vulnerability of them all, default logins [1, 2, 3]. However, these vulnerabilities all share one common trait: with very few exceptions they exist solely on the internal LAN, thus requiring an attacker to gain access to the LAN prior to exploiting a router's vulnerabilities.

There are various existing attacks which may be employed by an external attacker to target a router's internal Web interface. These attacks are best delivered via an internal client's Web browser when that client browses to a Web page controlled by the attacker.

Cross-Site Request Forgery (CSRF) is perhaps the easiest of these attacks and is well suited for exploiting authentication bypass vulnerabilities. However, it is restricted in several ways:

1. While it can be used to generate requests, it cannot be used to view the response. This is due to the browser's same-domain policy, and eliminates the possibility of using CSRF to exploit certain vulnerabilities, such as information disclosure vulnerabilities [4].

2. It cannot be used to exploit default logins on routers that employ Basic HTTP authentication. While this used to be possible by specifying a specially crafted URL (such as *http://user:password@192.168.1.1*), modern versions of Firefox will display a pop-up warning to the end user, and Internet Explorer no longer recognizes such URLs as valid (see Appendix A).

3. It requires the attacker to pre-craft his attack to target specific routers. The attacker's exploit Web page must know or guess the router's internal IP or host name, as well as the router's make and/or model. Although there have been some efforts to produce JavaScript-based router identification code, it is experimental and generally does not work against routers that employ Basic HTTP authentication for reasons described in #2 above [5].

Due to the limitations of CSRF, DNS re-binding attacks are very attractive to a remote attacker, as they provide a means to circumvent the browser's same-domain policy and allow the attacker's code (typically JavaScript) to interact with the router's internal Web interface [6].

The object of a DNS re-binding attack is to get a user to load JavaScript from an attacker’s Web page and then re-bind the attacker’s domain name to the IP address of the user’s router. The attacker's JavaScript code can then make interactive XMLHttpRequests to the router by opening a connection to the attacker's domain name since that domain now resolves to the IP address of the victim's router. This connection is allowed by the browser's same-domain policy since the JavaScript is running in a Web page that originated from the attacker's domain.

However, DNS re-binding attacks are well known and Web browsers have implemented various protections to help mitigate the threat of a DNS re-binding attack. Most notably, DNS pinning is implemented in all modern browsers. DNS pinning attempts to prevent DNS re-binding attacks by caching the first DNS lookup that the browser performs; without a second DNS lookup, the same IP address will always be used for a given domain name until the browser’s cache expires or the user exits the browser.

For this reason anti-DNS pinning attacks have become increasingly popular. These attacks attempt to circumvent browser DNS caching and force the browser to perform a second DNS lookup, at which time the attacker's domain name gets re-bound to an IP address of the attacker's choosing.

While DNS-rebinding attacks are possible, they have the noted disadvantage of time: such attacks take 15 seconds in Internet Explorer, and 120 seconds in Firefox [7]. Further, IE8 and Firefox 3.x browsers no longer appear to be vulnerable to these types of DNS re-binding attacks (see Appendix B).

The Multiple A Record Attack

An older method of DNS re-binding is the “multiple A record” attack as discussed in Stanford's *Protecting Browsers From DNS Rebinding Attacks* paper [8]. This attack is better known to DNS administrators as DNS load balancing.

In this scenario, an attacker's DNS response contains two IP addresses: the IP address of the attacker's server followed by the IP address that the attacker wishes to re-bind his domain name to. The client's Web browser will attempt to connect to the IP addresses in the order in which they appear in the DNS response:

1. The client's browser connects to the first IP address in the DNS response, which is the IP address of the attacker's Web server, and retrieves the HTML file containing the attacker's JavaScript.

2. After the client request is completed, the attacker's Web server then creates a new firewall rule to block further connections from that client's IP address.

3. When the attacker's JavaScript initiates a request back to the attacker's domain via an XMLHttpRequest, the browser will again try to connect to the attacker's Web server. However, since the client's IP is now blocked, the browser will receive a TCP reset packet from the attacker's Web server.

4. The browser will automatically attempt to use the second IP address listed in the DNS response, which is the IP address of the client’s router. The attacker's JavaScript can now send requests to the router as well as view the responses.

One notable restriction to the multiple A-record attack is that it can not be used to target internal IP addresses, such as the internal IP address of a router. If a DNS response contains any non-routable IP addresses, the browser will attempt to connect to those addresses first, regardless of their order in the DNS response. This obviously breaks the exploit, as the attacker needs the client's browser to connect to his public Web server first and retrieve his malicious JavaScript.

DNS Rebinding and the Weak End System Model

But what happens if an attacker uses the multiple A-record attack to re-bind his domain to the router's *public* (WAN) IP address? While it may not be the intended or expected behavior, this attack will in fact succeed if the router's TCP/IP stack implements the weak end system model [9]. While the weak end system model is certainly not unique to Linux, Linux is perhaps the most popular embedded OS that supports it, and thus it is provided as an example here.

In the weak end system model, the TCP/IP stack will accept and process an incoming packet as long as the destination IP matches *any* of its local IP addresses. The IP address does not need to match the IP address of the interface on which the packet was received [10].

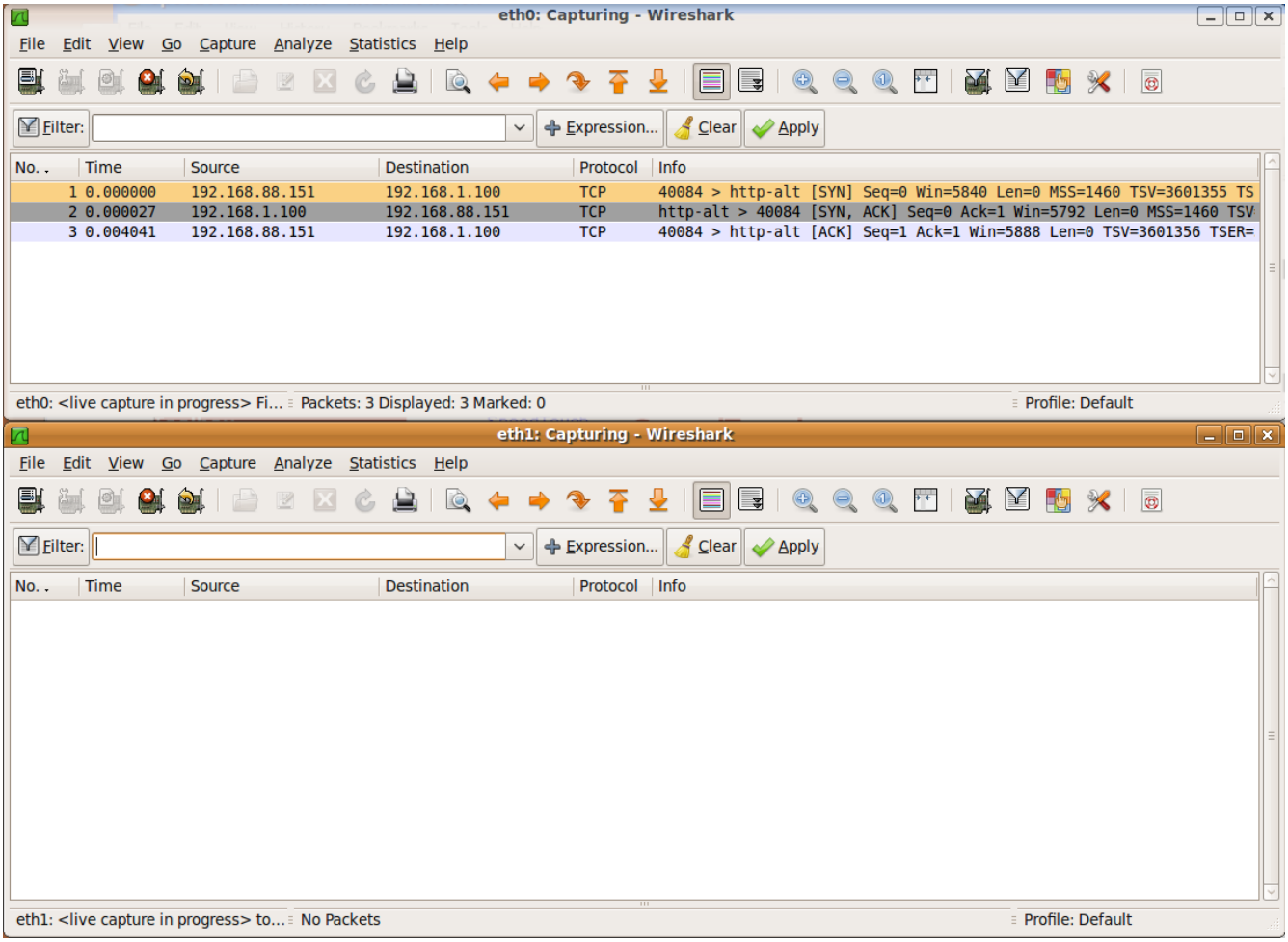
For example, consider a router with two interfaces, eth0 and eth1, where eth0 is assigned the class C IP address of 192.168.88.5 and eth1 is assigned the class C IP address of 192.168.1.100. These are clearly two separate IP address ranges on two physically distinct network interfaces:

```
root@router:~# route -n
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U         0      0        0 eth1
192.168.88.0    0.0.0.0         255.255.255.0   U         0      0        0 eth0
```

Now let us also assume that the router has a listening service running on port 8080 and bound exclusively to the eth1 interface (192.168.1.100):

```
root@router:~# netstat -plnt
Proto   Local Address           Foreign Address         State       PID/Program name
tcp     192.168.1.100:8080      0.0.0.0:*               LISTEN      22393/nc
```

Notice what happens when a host on the eth0 network attempts to connect to the router's eth1 IP address of 192.168.1.100:



The connection succeeds, even though the IP address of the eth0 interface does not match the destination IP address in the client's packets. At first glance this is not particularly interesting; after all, a router is supposed to route requests.

Upon closer examination of the above Wireshark captures however, one will note that **the request is not being routed at all**, as evidenced by the lack of any traffic on the eth1 interface. Due to the weak end system model, **the request is being accepted and processed entirely on the eth0 interface**, as if the listening service were being run on the eth0 interface when in fact it is not. The router sees that the destination address matches *one* of its local addresses and simply accepts the connection.

To understand how this can be exploited by an attacker, consider that most routers are quite simple in nature, and they often run their services on all interfaces rather than on just one particular interface:

```
root@OpenWrt:/# netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:80                    *:*                     LISTEN
tcp        0      0 *:53                    *:*                     LISTEN
tcp        0      0 *:22                    *:*                     LISTEN
tcp        0      0 *:23                    *:*                     LISTEN
```

Since these **services are bound to all interfaces**, iptables rules are applied to prevent inbound connections on the WAN interface. Below are the simplified iptables rules relevant to this discussion:

```
-A INPUT –i eth1 –j DROP
-A INPUT –j ACCEPT
```

Any incoming traffic on the WAN interface (eth1) is dropped, while inbound traffic on the LAN interface(s) is allowed. Note that **these rules are applied based on the inbound interface name, and do not specify the destination IP address**. This is because the WAN IP is likely assigned via DHCP from an ISP and is subject to change at any time.

However, as evidenced in the above Wireshark captures, when an internal user on eth0 connects to the external IP address on eth1 **the network traffic never actually traverses the TCP/IP stack of the eth1 interface.** The iptables rules put in place to block incoming traffic to a router's WAN interface (eth1) will not prevent internal clients from connecting to services on the WAN interface because the traffic will never actually traverse the WAN interface.

Armed with this information, an attacker can use the multiple A-record attack to re-bind his domain name to the router's public IP and access the router's Web interface via an internal client's Web browser. Not only does this make the multiple A-record attack a viable DNS re-binding vector, but it also eliminates the need for the attacker to know or guess the target router's internal IP address.

A Practical Example

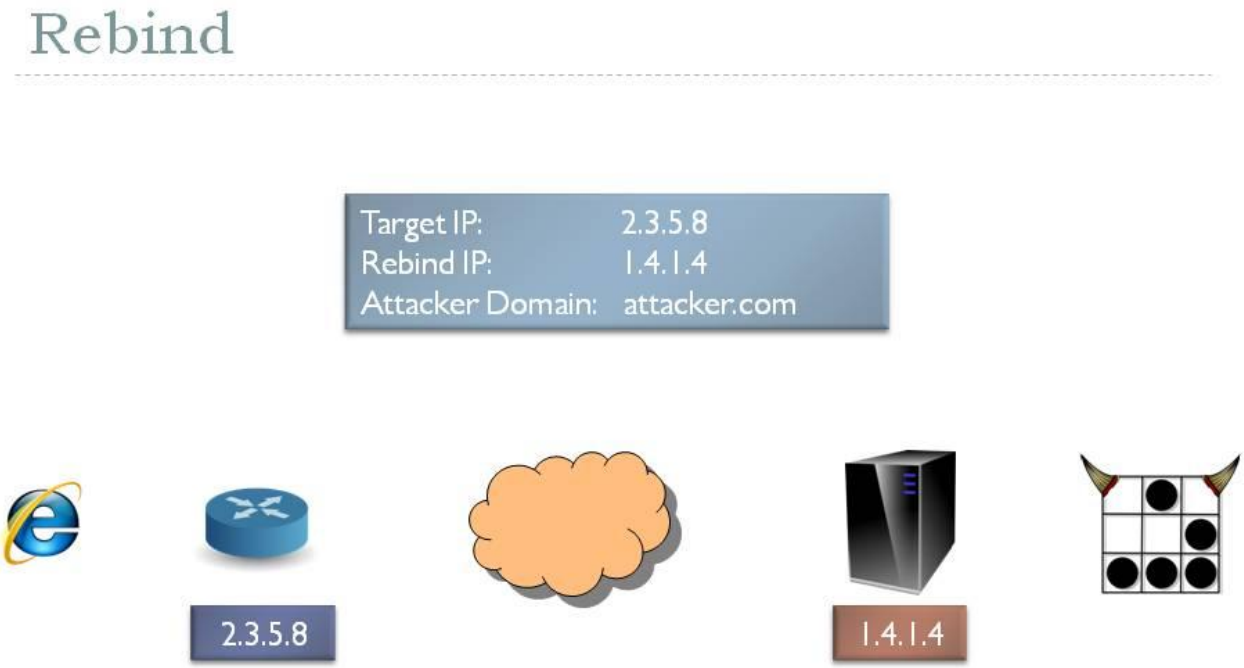
In order to provide a practical demonstration of this attack, Rebind was created. Rebind is a tool that not only implements the DNS re-binding attack, but uses it to turn the client's Web browser into a real-time proxy through which the attacker can interact with the client's router as if the attacker were on the LAN himself. This is similar in concept to David Bryne's anti-DNS pinning presentation at BlackHat 2007 and eliminates the requirement for the attacker's JavaScript code to perform any interpretation or analysis of the client's router; it simply acts as a proxy between the attacker and the client's router [11].

Rebind integrates a DNS server, two Web servers and an HTTP proxy server into a single Linux binary. The only prerequisites for running Rebind are that:

1) The attacker must register a domain name
2) The attacker must register his attack box as a name server for his domain

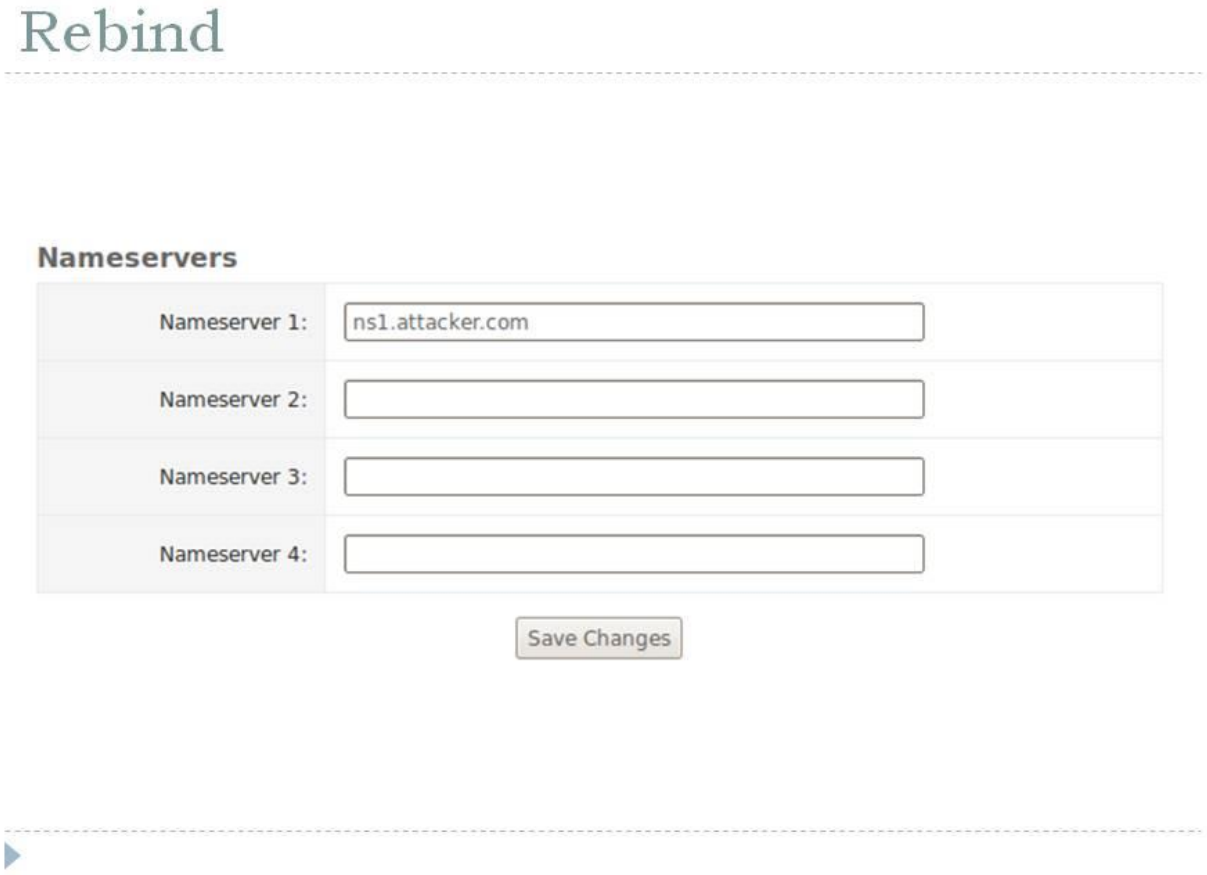
Both of these requirements can be fulfilled through almost any domain registrar [12]. In the following example scenario, the attacker's domain is attacker.com, and he has registered a DNS server named ns1.attacker.com. The ns1.attackercom entry points to the IP address of his attack box. The following depicts how Rebind works:

1. **Example scenario. The attacker owns the domain name attacker.com and is running Rebind on his server located at 1.4.1.4. The victim's public IP is 2.3.5.8:**

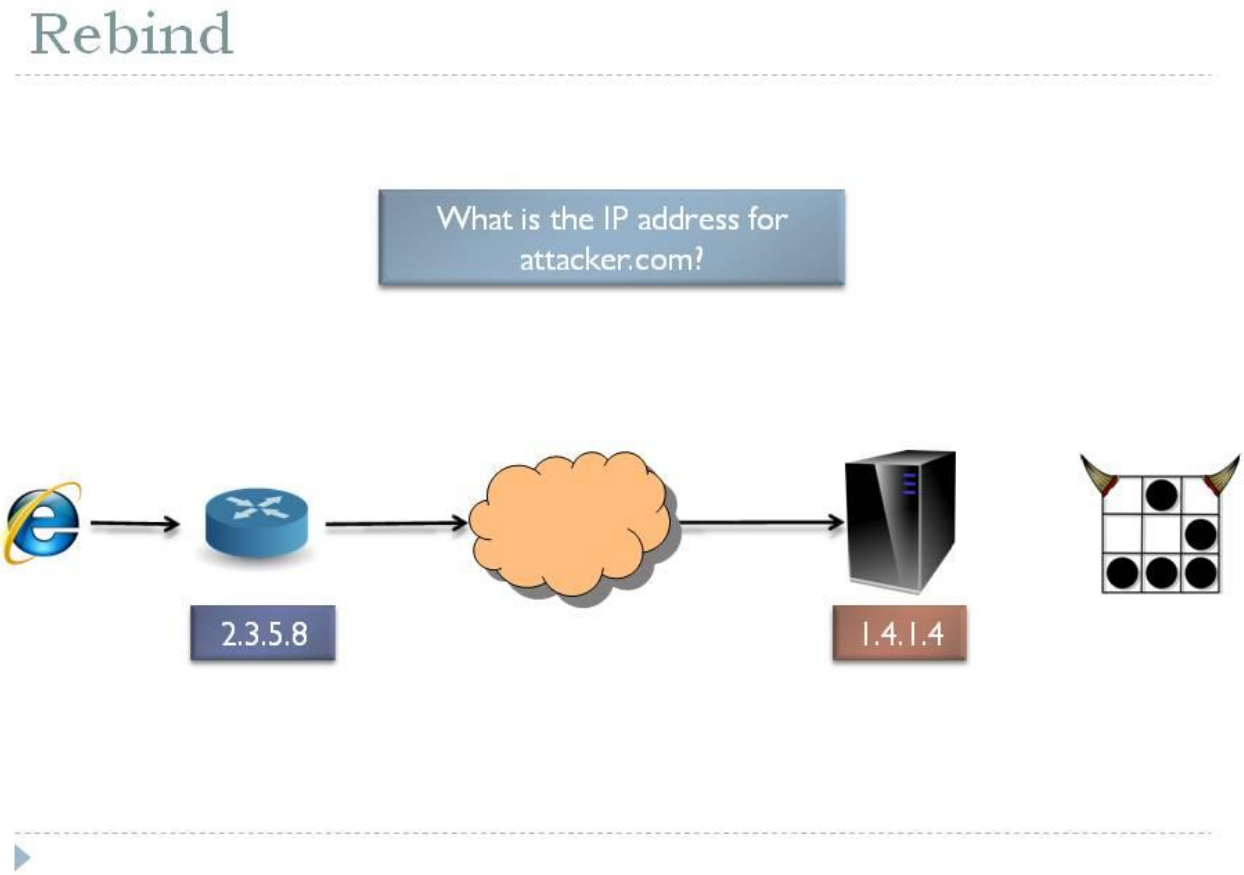


2. **The attacker goes to the registrar where he registered his domain name and registers 1.4.1.4 as a name server named ns1.attacker.com:**

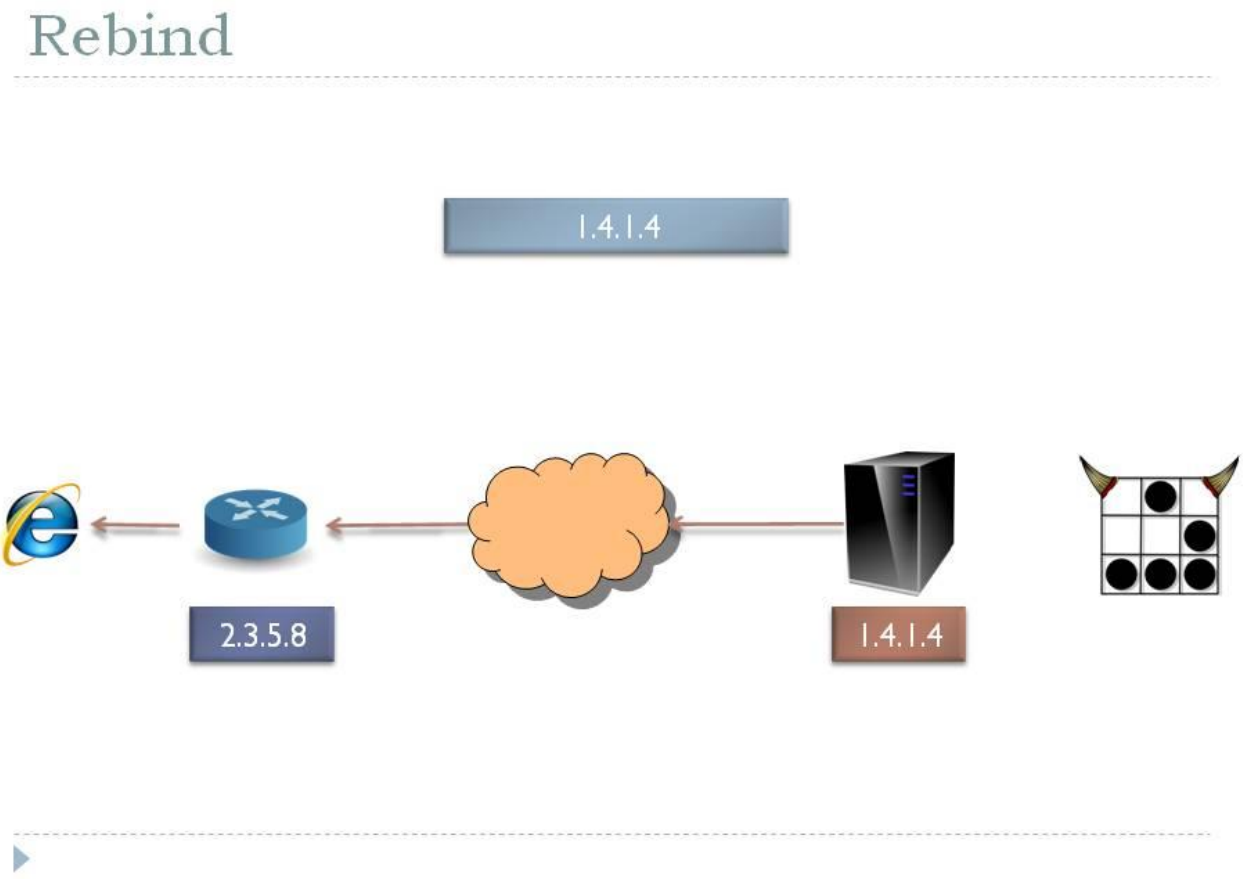
3. **The attacker tells his registrar to use ns1.attacker.com as the primary DNS server for the attacker.com domain:**



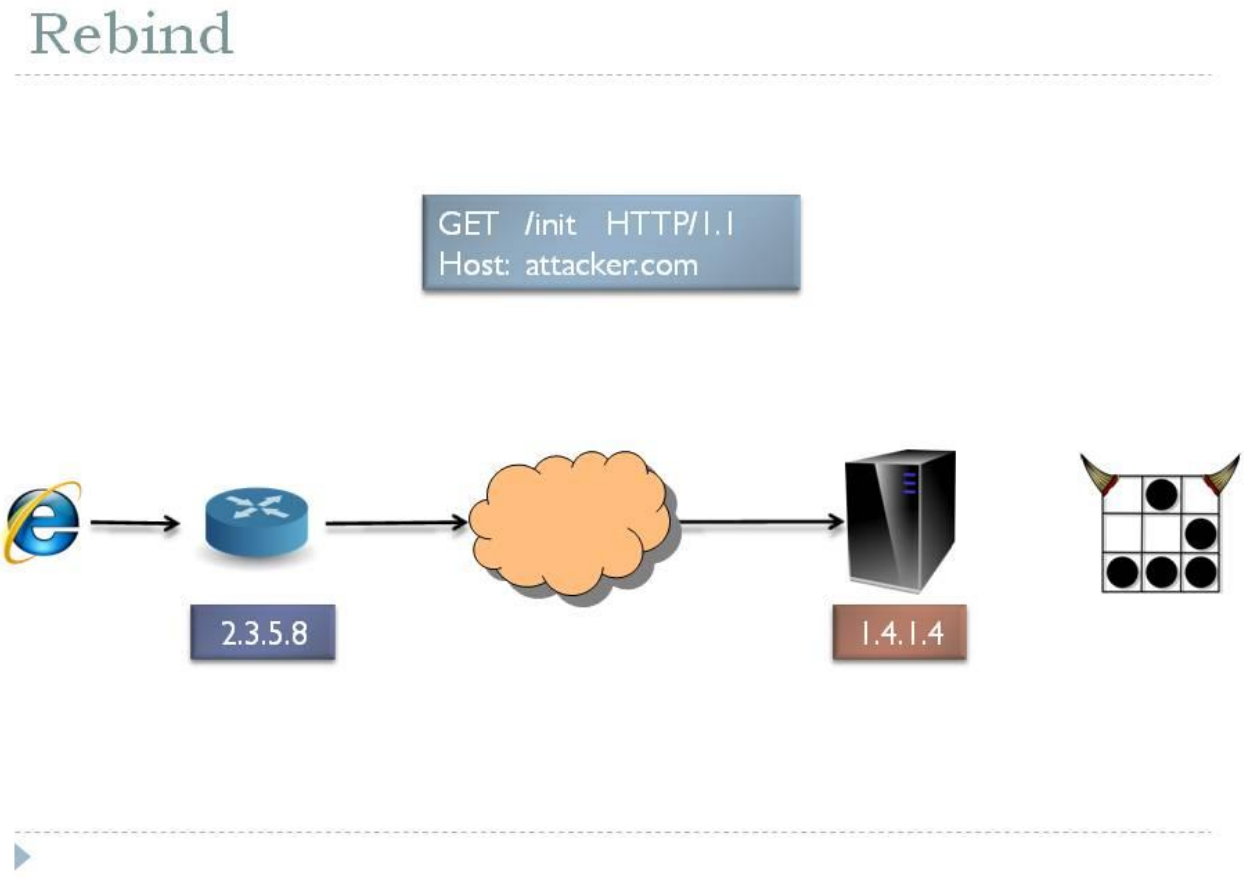
4. **The victim attempts to browse to http://attacker.com/init, and subsequently the victim's browser issues a DNS lookup for attacker.com:**



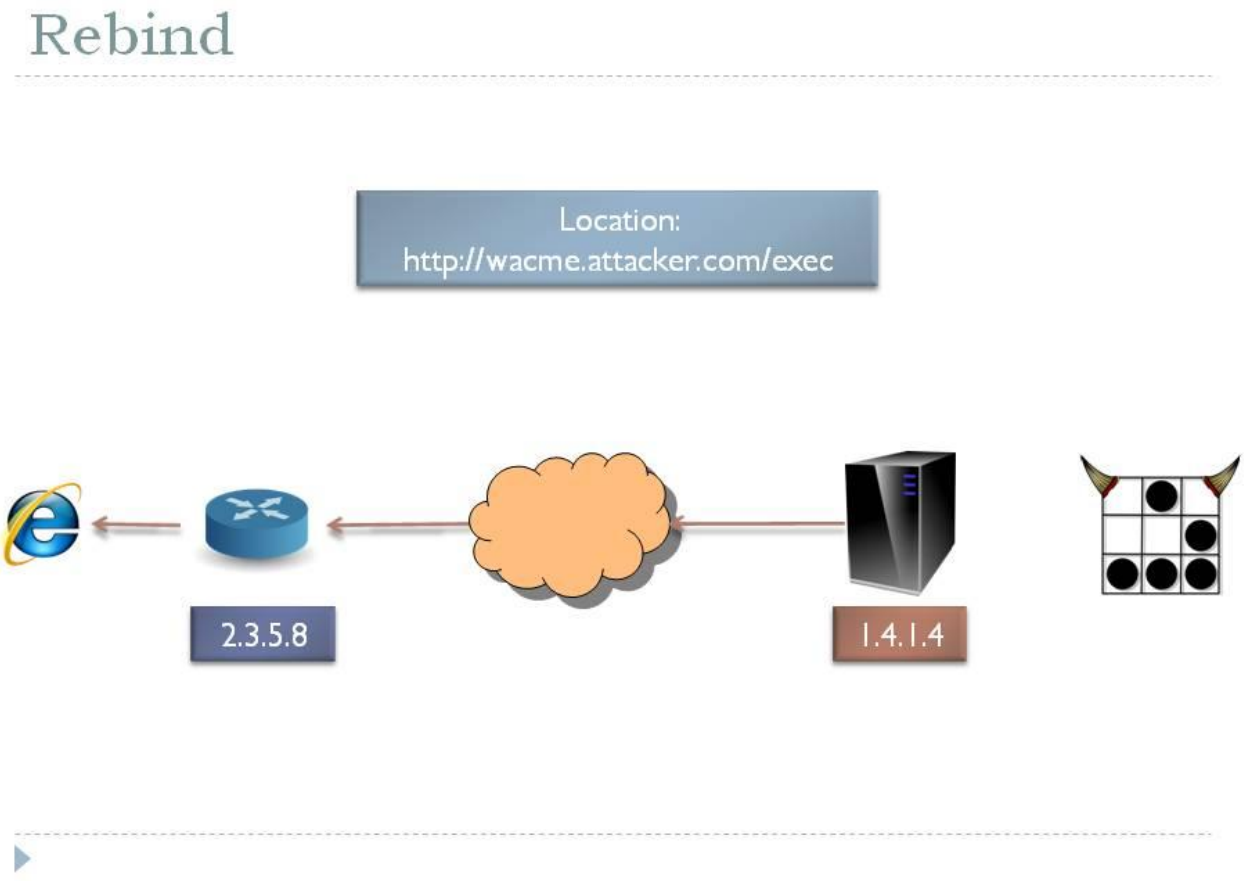
5. **Rebind receives the DNS request and responds with its own IP address:**



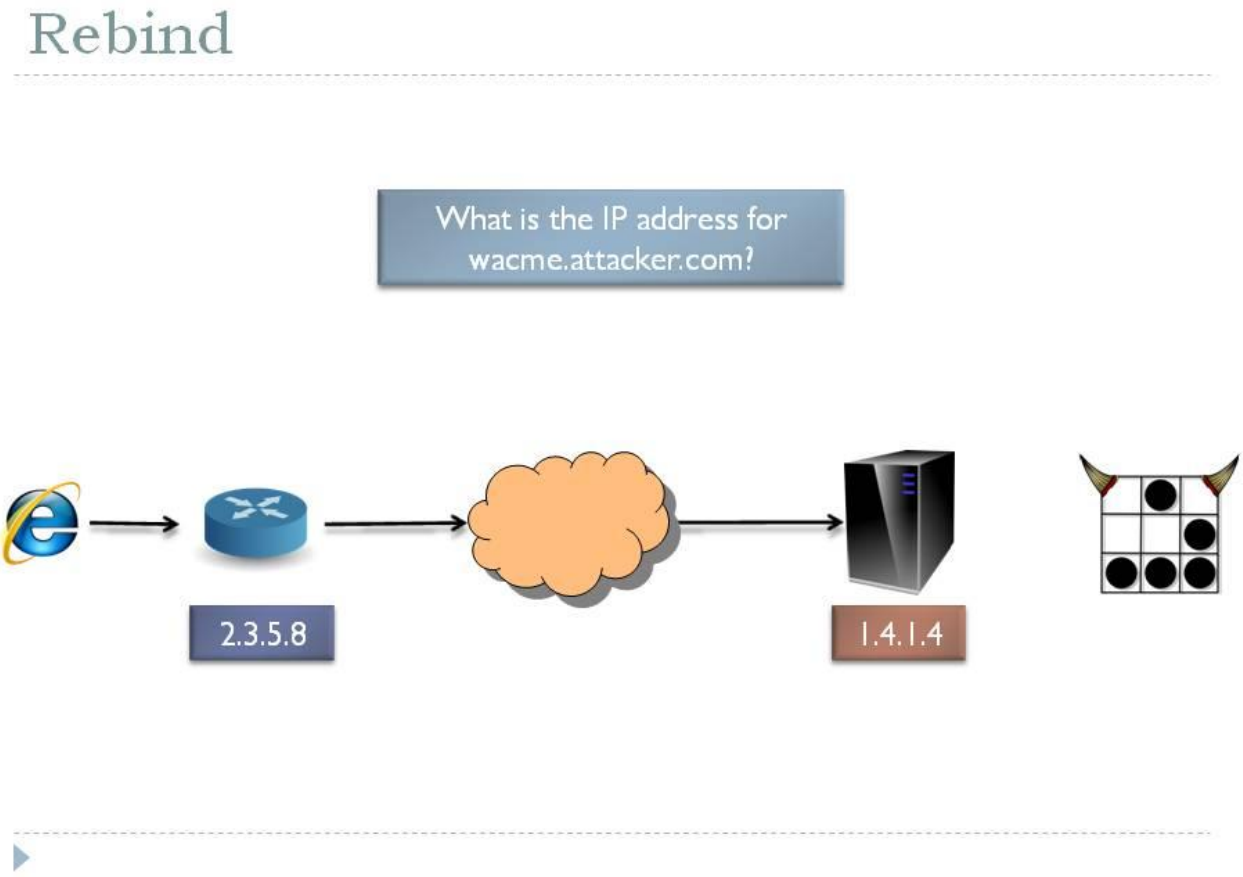
6. **The victim's Web browser requests the /init page from attacker.com:**



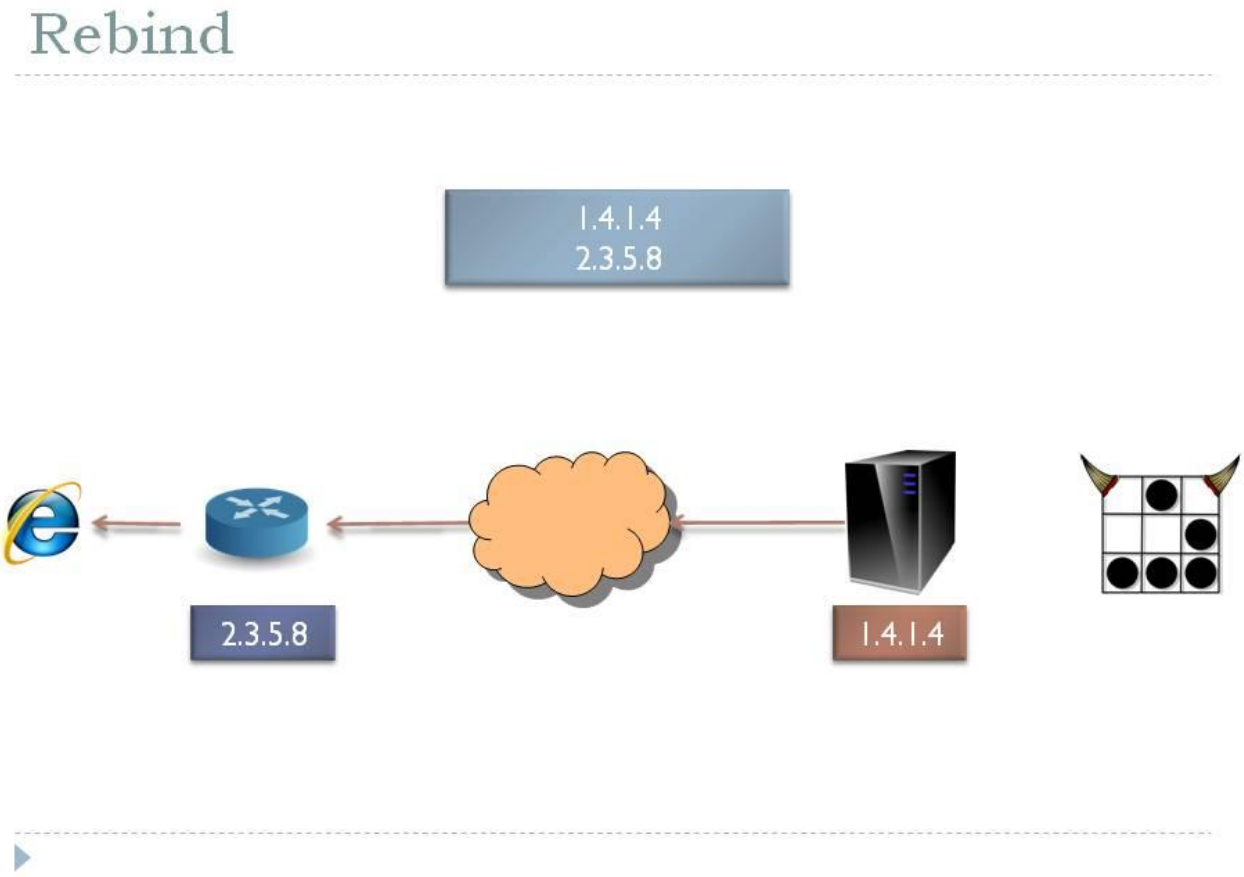
7. **Rebind receives the HTTP request, logs the router's public IP and redirects the victim's Web browser to a random sub-domain of devttys0.com; in this case, wacme.www.devttys0.com/exec:**



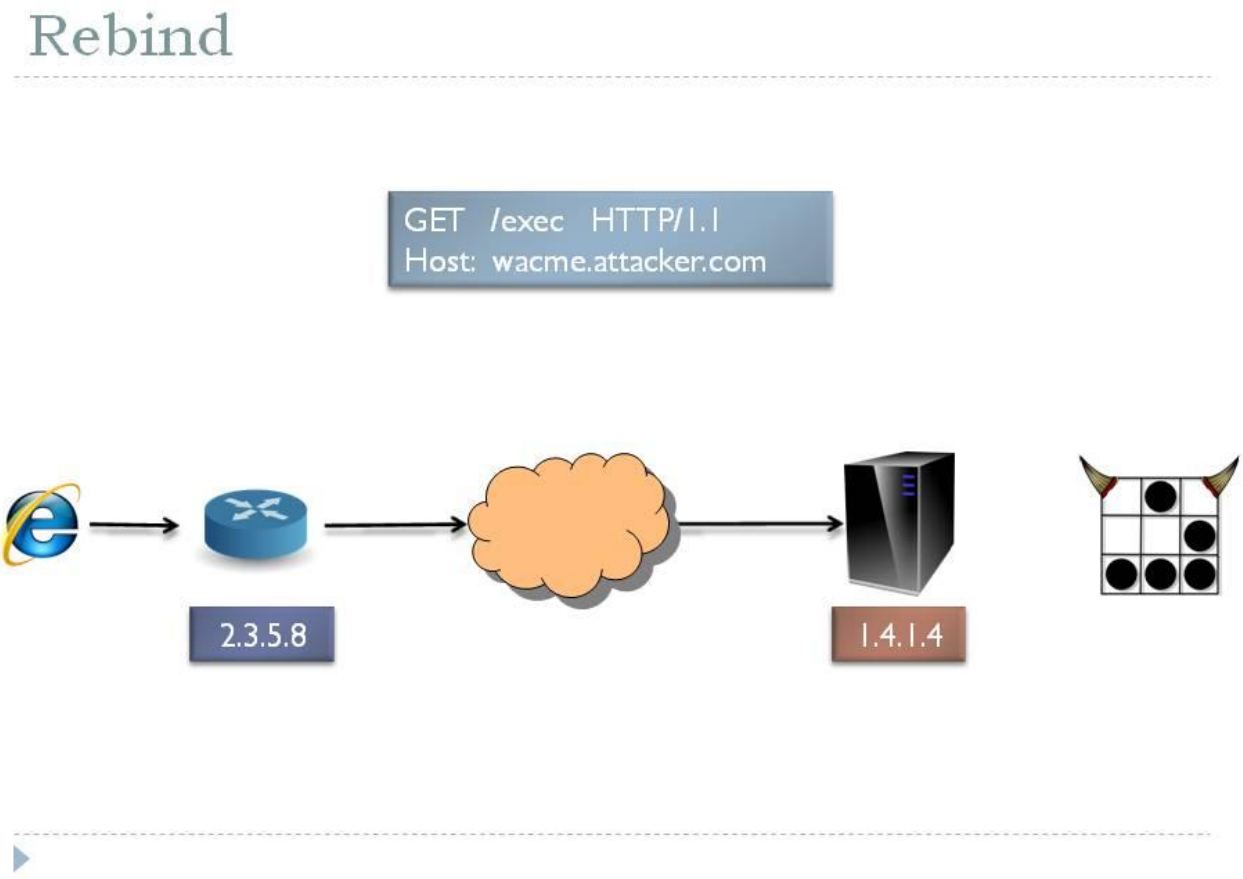
8. **This forces the victim's Web browser to perform a second DNS lookup for wacme.attacker.com:**



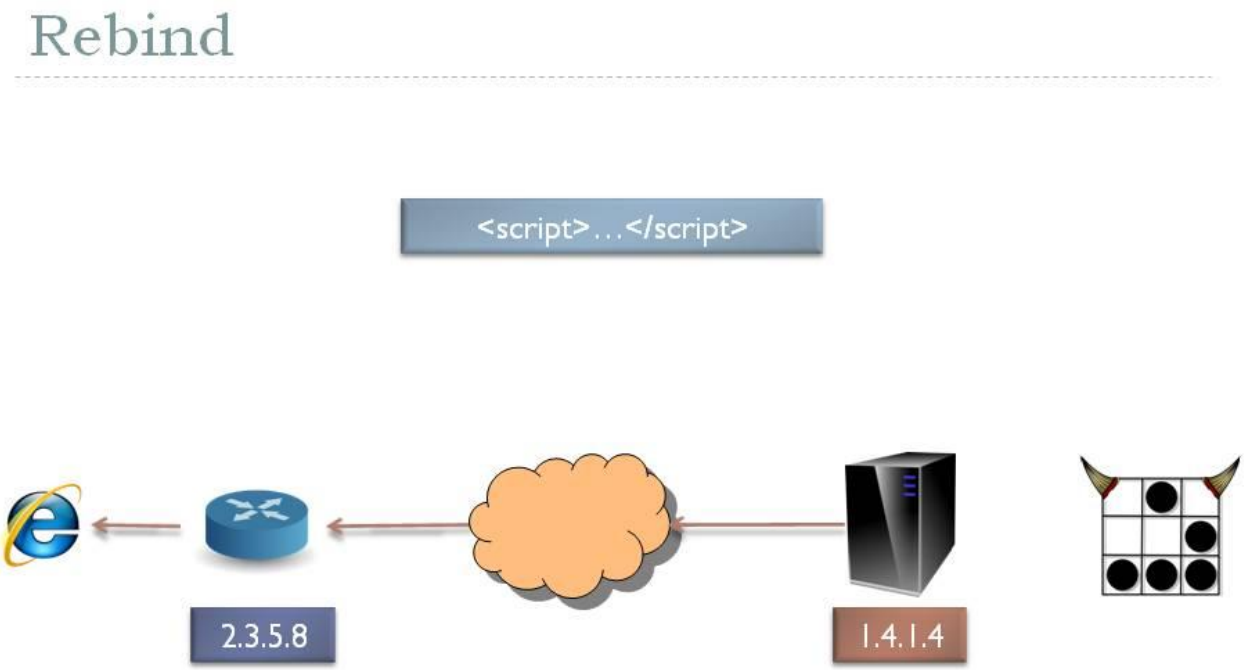
9. **Rebind receives the DNS request and responds with its own IP address, followed by the public IP address of the victim's router which it logged in step #7:**



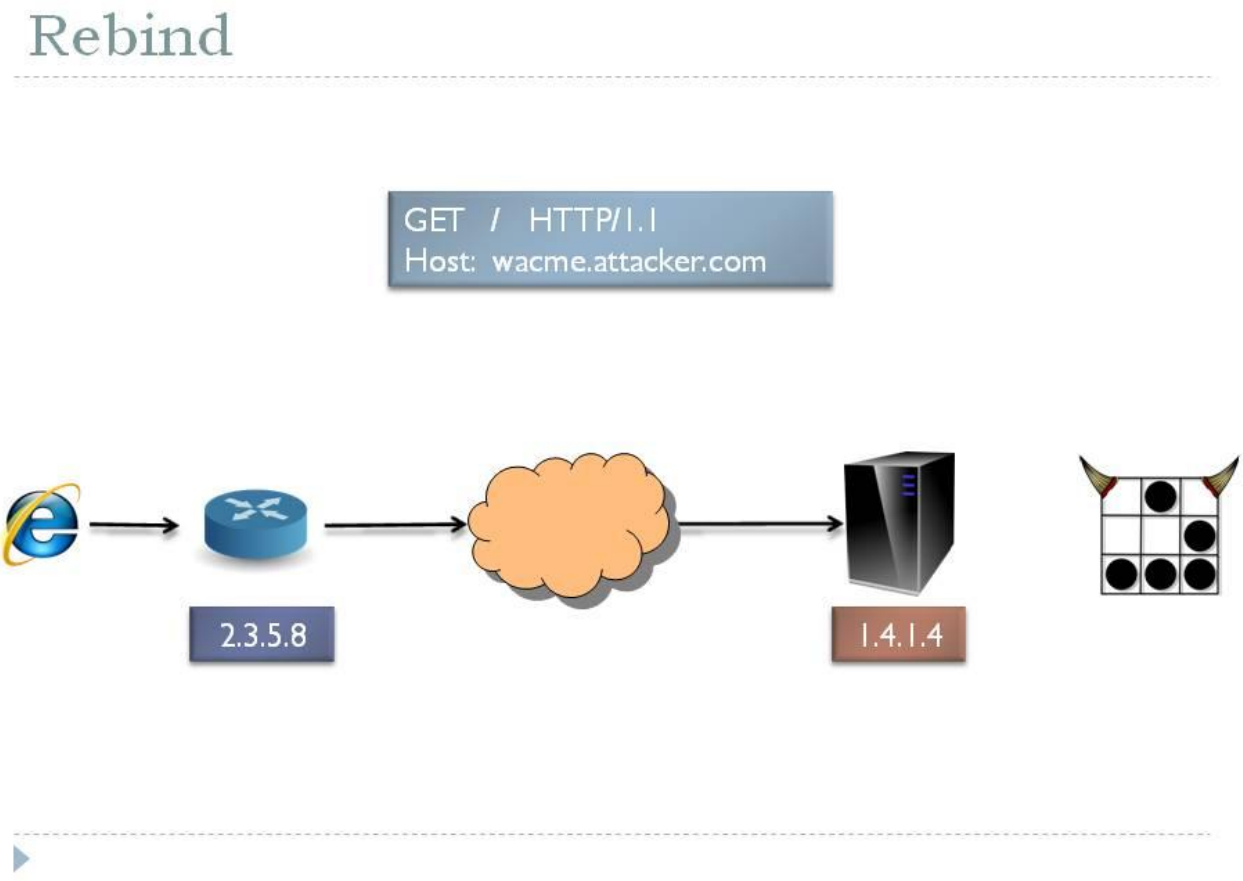
10. The victim's Web browser requests the /exec page from wacme.attacker.com:



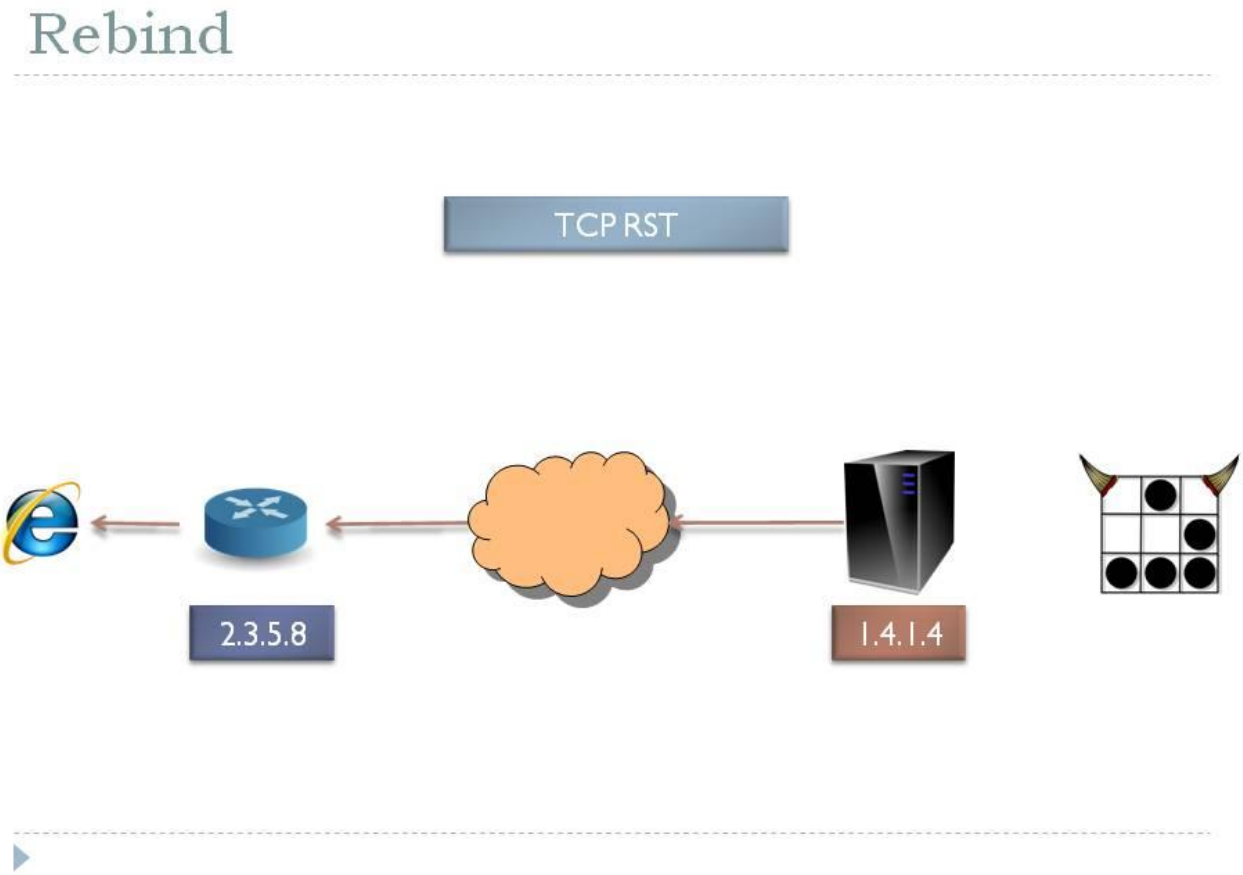
11. **Rebind sends the client its JavaScript code and blocks further connection attempts from the victim to port 80 of the attack box:**



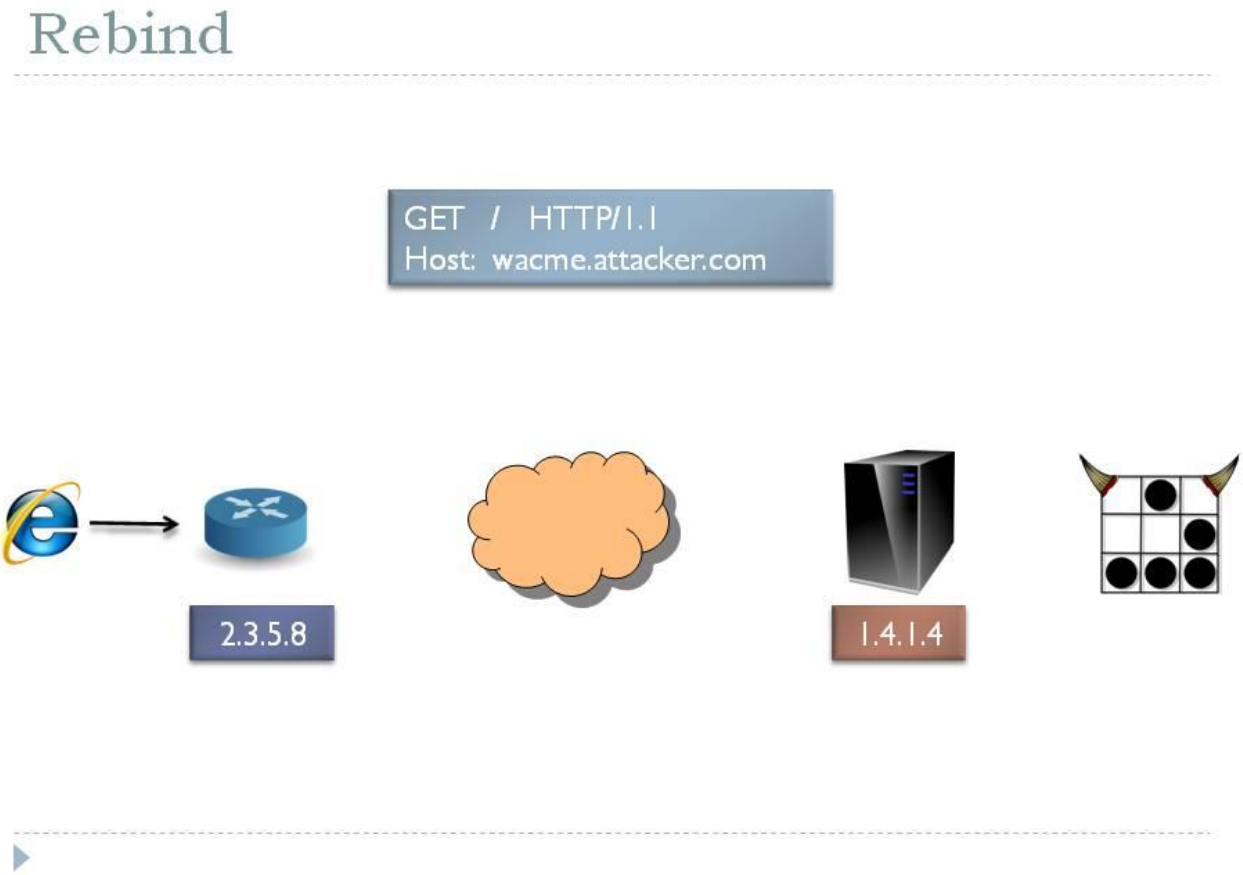
12. The JavaScript code attempts an XMLHttpRequest to wacme.attacker.com:



13. **Since Rebind has blocked the victim on port 80, the connection attempt is refused with a TCP reset packet:**

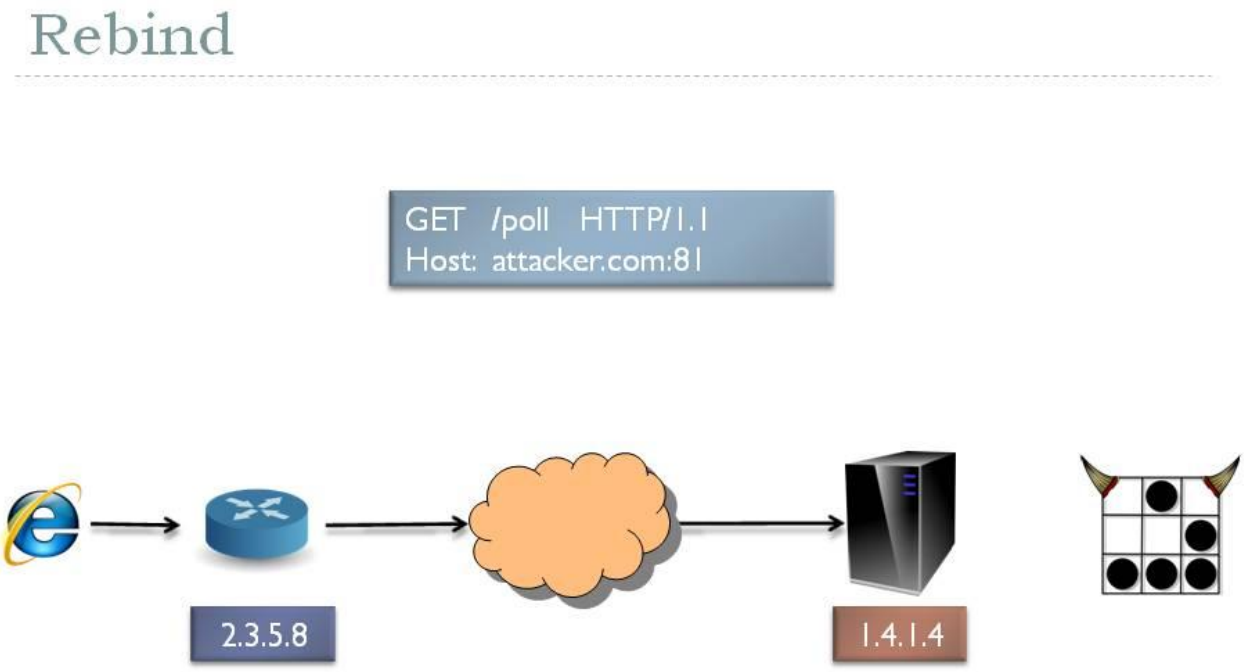


14. The victim's browser automatically attempts to connect to the second IP address for wacme.attacker.com, which is the public IP address of the victim's router:

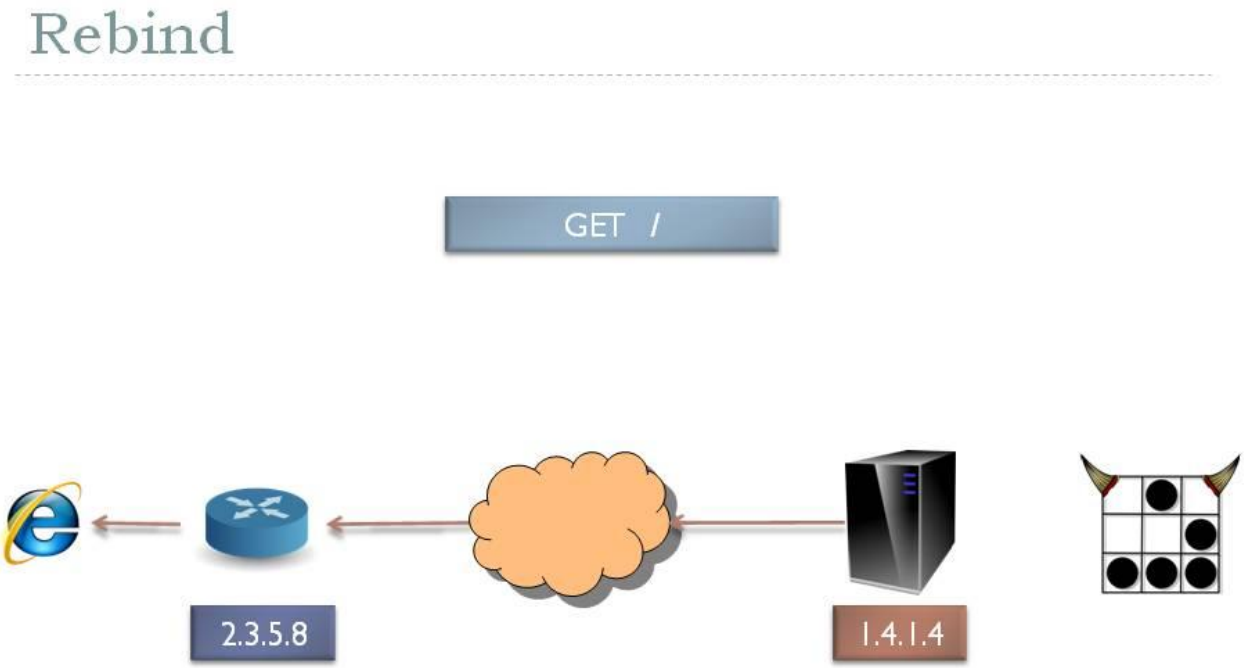


15. **With the sub-domain of wacme.attacker.com now re-bound to the router's public IP address, the external attacker can send requests to the client's router via Rebind's HTTP proxy. Rebind queues these requests into a database and holds the connection to the attacker's Web browser open:**

16. **The JavaScript code begins loading dynamic JavaScript elements from Rebind. Since wacme.attacker.com is re-bound to the router's public IP, the JavaScript elements are loaded from attacker.com. Port 81 is used since the victim is still blocked on port 80:**

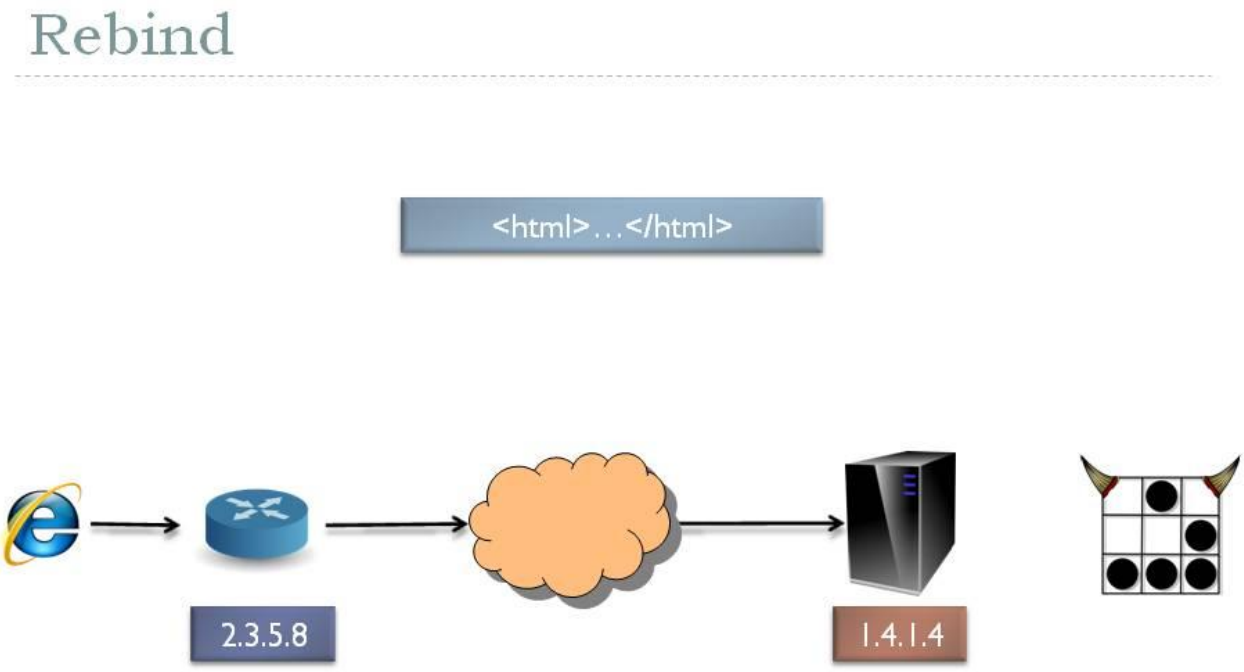


17. **Rebind responds to the JavaScript callback requests with the HTTP request that the attacker made via Rebind's proxy:**

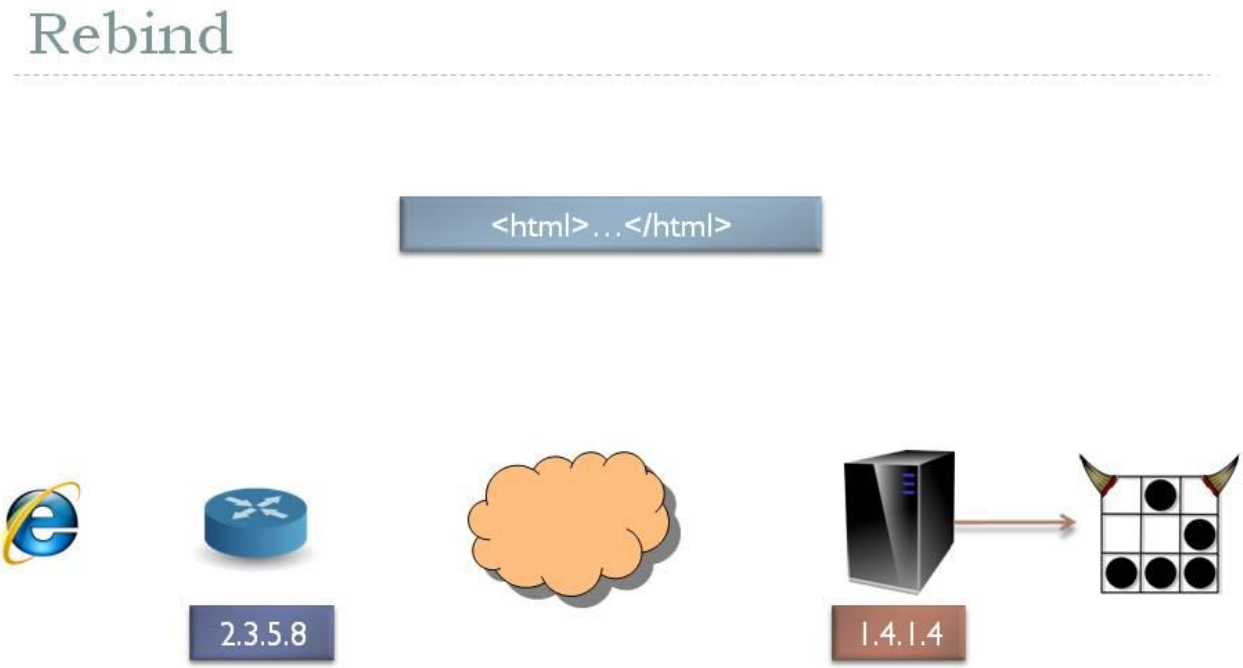


18. **The JavaScript then issues an XMLHttpRequest to wacme.attacker.com to retrieve the requested page from the client's router:**

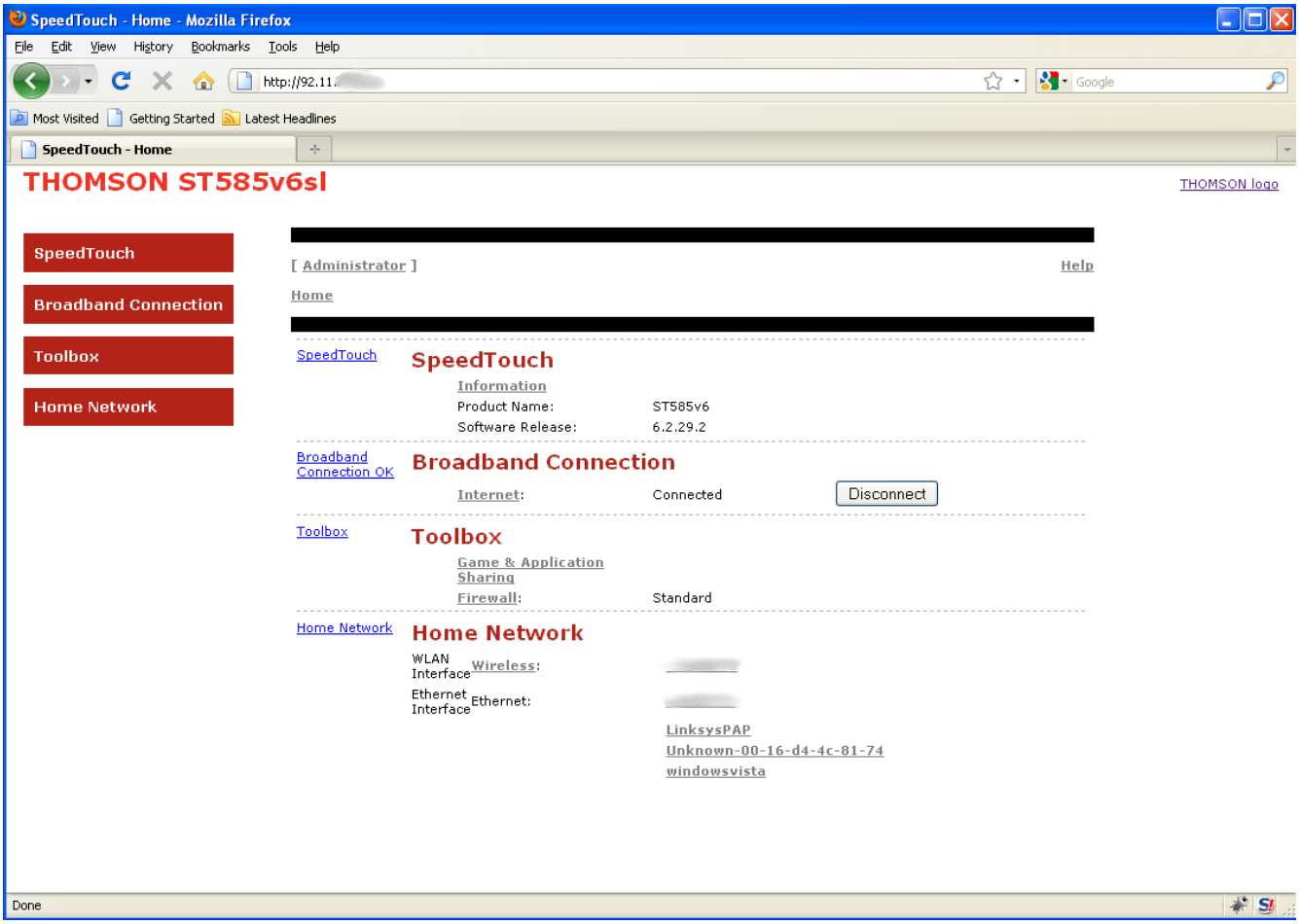
19. **The response headers and text received from the router are then sent back to Rebind. If the browser supports cross-domain requests, they are used; if not, then an IFrame transport is used instead [13]:**



20. **When Rebind receives the response data from the victim's browser, it relays it back to the attacker's browser:**



21. **The attacker is now browsing around in the victim's router as if he were on the LAN himself:**



Impact

Many routers are affected, but perhaps the most notable is the ActionTec MI424-WR. This router is distributed to all Verizon FIOS users, the vast majority of whom never bother to change the default login. As of 2009, Verizon FIOS claimed over 3 million Internet subscribers, providing an attacker with a very large number of potential targets from this one router alone [14].

Additionally, routers from many popular vendors such as Linksys, Thompson, Belkin, Dell, and Asus are known to be affected. The attack has also been successfully tested against routers running third party firmware from DD-WRT, OpenWRT and PFSense.

The following table lists the routers that were tested against Rebind, and whether or not the Rebind attack was successful. Where known and applicable, the hardware and firmware versions of the routers have been included:

Vendor	Model	H/W Version	F/W Version	Successful
ActionTec	MI424-WR	Rev. C	4.0.16.1.56.0.10.11.6	YES
ActionTec	MI424-WR	Rev. D	4.0.16.1.56.0.10.11.6	YES
ActionTec	GT704-WG	N/A	3.20.3.3.5.0.9.2.9	YES
ActionTec	GT701-WG	E	3.60.2.0.6.3	YES
Asus	WL-520gU	N/A	N/A	YES
Belkin	F5D7230-4	2000	4.05.03	YES
Belkin	F5D7230-4	6000	N/A	NO
Belkin	F5D8233-4v3	3000	3.01.10	NO
Belkin	F5D6231-4	01	2.00.002	NO
D-Link	DI-524	C1	3.23	NO
D-Link	DI-624	N/A	2.50DDM	NO
D-Link	DIR-628	A2	1.22NA	NO
D-Link	DIR-320	A1	1.00	NO
D-Link	DIR-655	A1	1.30EA	NO
DD-WRT	N/A	N/A	v24	YES
Dell	TrueMobile 2300	N/A	5.1.1.6	YES
Linksys	BEFW11S4	1.0	1.37.2	YES
Linksys	BEFSR41	4.3	2.00.02	YES
Linksys	WRT54G3G-ST	N/A	N/A	YES
Linksys	WRT54G2	N/A	N/A	NO
Linksys	WRT-160N	1.1	1.02.2	YES
Linksys	WRT-54GL	N/A	N/A	YES
Netgear	WGR614	9	N/A	NO
Netgear	WNR834B	2	2.1.13_2.1.13NA	NO
OpenWRT	N/A	N/A	Kamikaze r16206	YES
PFSense	N/A	N/A	1.2.3-RC3	YES
Thomson	ST585	6sl	6.2.2.29.2	YES

Mitigations

There are several fixes that can be employed by end users to mitigate this type of attack:

1. Add a firewall rule on the router to block network traffic on the internal interface that has a destination IP which matches the IP address assigned to the WAN interface. Note that this rule will have to be updated each time the IP address of the WAN interface changes.

2. Add a firewall or routing rule on each host machine in the network that blocks or improperly routes traffic destined for the router's public IP address. Note that this rule must be applied to any and all current and future hosts on the internal network.

3. Ensure that services on the router are bound only to the LAN and/or WLAN interfaces.

4. Only run Web-based administration over HTTPS.

5. Don't use Web-based administration at all.

If you do not have sufficient access to your router to perform the above mitigations, preventing this attack can be difficult. The best option is to disable JavaScript or limit from which sites you allow JavaScript to run.

There are additional permanent fixes that can be made by firmware authors to prevent this attack:

1. Require a valid host header in all HTTP requests. The host header should always be the IP address of the router or the router's host name, if it has one.

2. Only run services, especially the Web service, on the LAN / WLAN interfaces unless the user has enabled remote administration.

3. Implement the strong end system model in the router's TCP/IP stack.

References

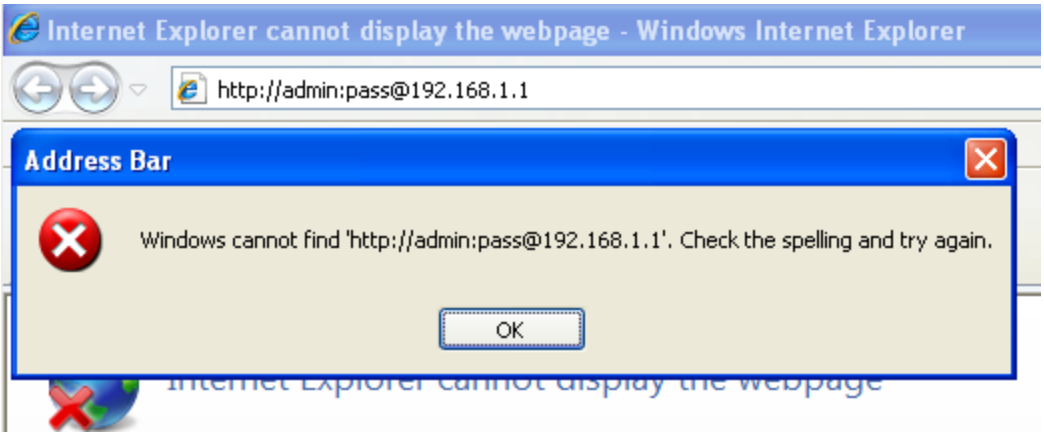
[1] GNUCitizen Router Hacking Challenge
[2] sla.ckers.org Full Disclosure Forum
[3] Security Vulnerabilities in SOHO Routers
[4] Same Origin Policy
[5] JavaScript LAN Scanner
[6] DNS Rebinding
[7] Stealing Information Using Anti-DNS Pinning
[8] Protecting Browsers From DNS Rebinding Attacks
[9] RFC1122 - Requirements for Internet Hosts - Communication Layers
[10] TCP/IP Illustrated Volume 2, p.218-219
[11] Intranet Invasion Through Anti-DNS Pinning
[12] Host-Unlimited
[13] Window.name Transport
[14] Verizon FIOS – Wikipedia

Appendix A

Blocked CSRF Attempts Using Basic Authentication URLs

I) Firefox confirmation warning

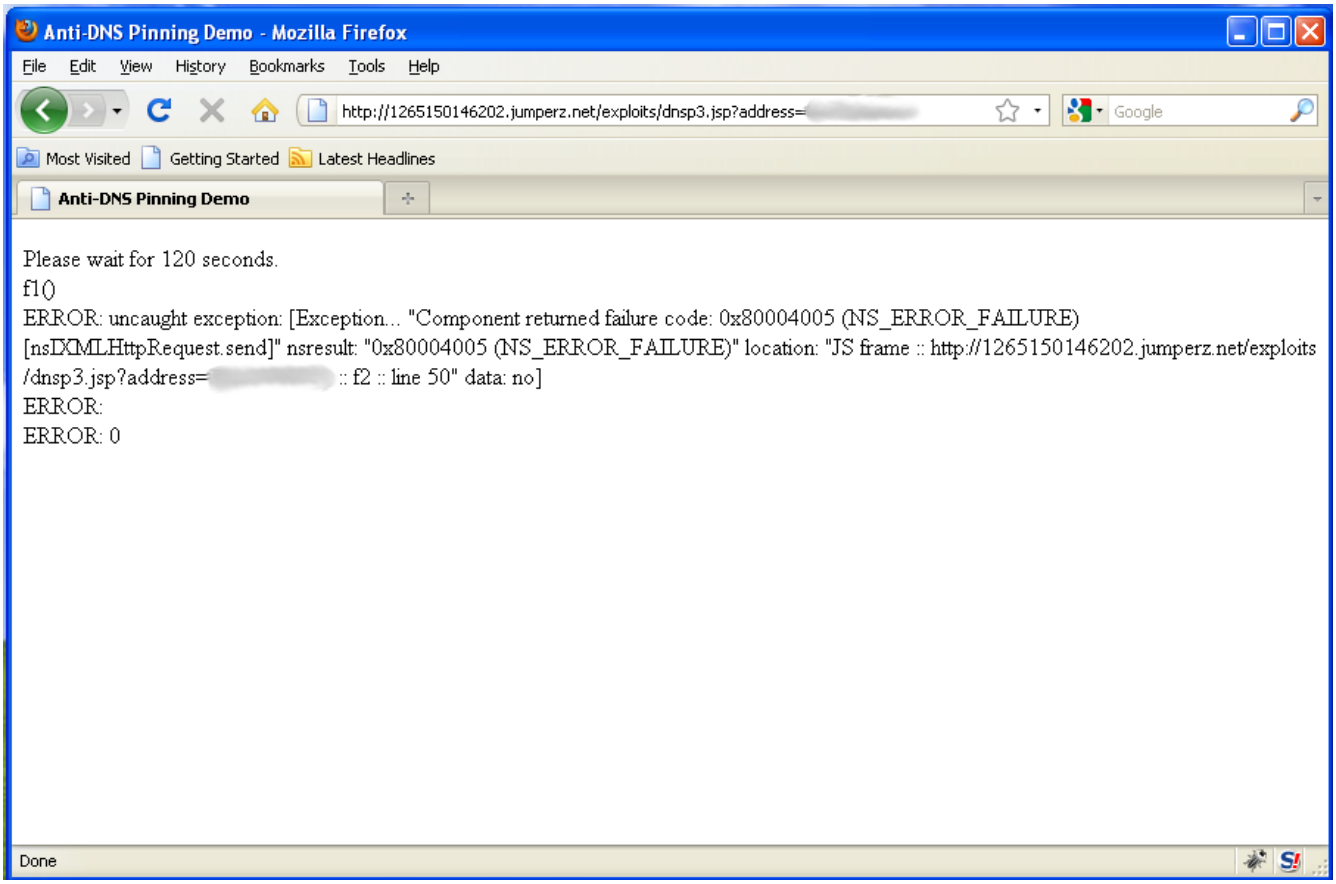
II) Internet Explorer invalid URL warning



Appendix B

Anti-DNS Pinning Attack Failures Against FF 3.5 and IE8

I) Firefox 3.5 JavaScript exception error



II) Internet Explorer 8 data retrieval failure