Browser Security Model*

Horst Görtz Institute for IT Security
Ruhr University Bochum, Germany

Michael Nordhoff
michael.nordhoff@rub.de

February 8, 2008

Abstract

Attacks against web authentication, known as rebinding attacks and dynamic pharming, are described. It works by hijacking DNS and infecting the victim's browser with malicious JavaScript or other active content, which then exploits the name-based same-origin policy to hijack a legitimate session after authentication has taken place. As a result, the attack works regardless of the authentication scheme used. Dynamic pharming enables the adversary to eavesdrop on sensitive content, forge transactions, key log secondary passwords, etc. To counter dynamic pharming attacks, several long-standing and recent proposals are identified and discussed. Also browser's plug-ins with specific vulnerabilities and their potential elimination are treated. Smarter pinning, finer-grained origins or policy-based pinning are some kinds of countermeasures. Recently two locked same-origin policies for web browsers were published. In contrast to the legacy same-origin policy, which regulates cross-object access control in browsers using domain names, the locked same-origin policy enforces access using servers' X.509 certificates and public keys. This policies help existing web authentication mechanisms resist dynamic pharming attacks.

1 Introduction

The well-known social engineering attack called *phishing* makes the credulous Internet user to disclose confidential information to the attacker, although he usually would direct these information only to a trustworthy communication partner he *thinks* to communicate with. The attack leads to an identity theft, the web visitor reveals his login credentials, e.g. personal identification numbers, bank account transaction numbers or credit card numbers. In a more advanced attack the adversary subverts the domain name systems (DNS). The attacker can force the DNS system to resolve a victim's site domain to an attacker-controlled IP address. This can be achieved by techniques like DNS poisoning and DNS response forgery. Nowadays DNS systems can be more and more manipulated in wireless network environments which are mainly still not secured properly. Often a wireless LAN router can be compromised easily and either the whole software can be swapped or manipulated or only DNS settings can be changed and might then point to a DNS server the attacker owns. This attack is also known as "Drive-by pharming" [1]. Additional to this deceptive (static) pharming attacks, there is a new, stronger attack called *dynamic pharming*. During this kind of attack, the attacker first delivers a web document containing malicious content (e.g. JavaScript code) to the victim, and then forces the victim's browser to connect to the legitimate server in a separate window, frame, table field, etc. The adversary waits for the victim to authenticate himself to the legitimate server, and then uses the malicious JavaScript to hijack the victim's authenticated session. Dynamic pharming can be used to compromise even the strongest web authentication schemes currently known, e.g. passwords, authentication cookies or client-side SSL [4]. It is a special kind of a DNS rebinding attack. These attacks

*This is a seminar work presented at the Chair of Network and Data Security, Ruhr University Bochum, Germany

need to cheat the client's same-origin policy, because the malicious script - coming from the adversary - needs access to the objects (web documents, authentication cookies, etc.) related or belonging to the honest web site. The usual same-origin policy fails because browsers consider two objects to have the same origin if the host and the protocol are the same. Actually the host and domain name of the malicious script and the honest web site are equal but the IP addresses, the actual and technical origins, are different. To face the problems of rebinding attacks, there are defenses by using smarter pinning, policy-based pinning or finer-grained origins. A recent proposal is using more sophisticated policies which not only binding to URLs but also to properties of the SSL, X.509 and public key system. More details are given later.

1.1 Preliminaries

For further analysis we define three underlying threat models: *phishers*, *pharmers* and active *man-in-the-middle attackers*. We assume a phisher has the following capabilities:

- complete control of a web server with a public IP address (but not the same domain name as the target domain)
- the ability to send emails or instant messages to potential victims
- mount application-layer MITM-attacks, representing a legitimate server to the victim and proxying input from the victim to the real server as needed.
- complete control of authoritative DNS server for resolving DNS names of the own domain

Pharmers are attackers which have all the abilities of a phisher, plus

- the ability to change DNS records for the target site, such that the victim will resolve the target site's name to the attacker's IP address.

We assume that the server under the pharmer's control does not have the same IP address as the victim's server and cannot receive packets destined to the victim's IP address.
Man-in-the-middle (MITM) attackers have all the abilities of a pharmer, plus

- the ability to control the Internet routing infrastructure and re-route traffic destined to particular IP addresses
- eavesdrop on all traffic.
- mount active, network-layer, MITM attacks

Furthermore we assume that users will ignore browser's certificate warnings that appear e.g. when a self-signed certificate is used or the certificate chain cannot be validated toward built-in root certificates. Studies have shown that users routinely ignore and dismiss such warnings [2].

1.2 Principle of Dynamic Pharming Attacks

In [4] the authors describe the so-called dynamic pharming attack. During usual (static) pharming attacks the adversary makes the victim's name service to answer queries for the target domain to always return the adversary's IP address. But during *dynamic* pharming attacks the adversary causes DNS to answer queries with either his IP address or the legitimate server's IP address - depending on the situation, the state of the attack. This can be achieved by a special form of rebinding attacks explained in chapter 4.3.
The reminder of this paper introduces object access mechanisms in chapter 2. The resulting DNS rebinding vulnerabilities are described in chapter 3, leading to different attack scenarios explained in chapter 4. In chapter 5 potential countermeasures are shown and discussed in the final section.

2 Object Access in Browsers

Today's web browsers are – including a plurality of plug-ins – highly integrated applications which leads to the multipurpose usability, specially for modern web applications. To face the security risks of untrusted content trying to execute harmful code and trying to get unauthorized access to other objects, the sandbox principle is used for the browser security model: each object using the browser's environment and its DOM model[1] has only access to resources of the same sandbox. Resources are mainly accessible through the network, but also history, cookies and

[1]The Document Object Model (DOM) is a platform- and language-neutral interface allowing scripts to dynamically access and update content, structure and style of documents. Standardized by W3C, usually used in browsers by JavaScript[3].

other objects need to be focused. The so-called sandboxes are generated for each origin of any object.

2.1 Access Within Same Origin

HTML and browser scripts can read and write network resources using the HTTP protocol. Plug-ins like Flash player or Java have additionally the opportunity to establish direct socket connections on other ports. While Java has no port number restrictions for the socket access, the Flash player needs to find a suitable XML policy file at the destination's site to connect to port numbers below 1024.
Some browsers block network access to prevent distributed denial of service attacks or cross site scripting even within same origin. The Internet Explorer 7 does not allow connections to e.g. FTP, SMTP, POP3, NNTP or IMAP. Otherwise the simple HTTP form protocol attack could be applied. Firefox 2 even blocks dozens of well-known server ports.[5]

2.2 Access Between Different Same Origins

In general web content and scripts have no authorization to access DOM elements of other origins. Focussing on network access, browser scripts may send HTTP requests to any destination, but if the destination differs from the script's origin the responses cannot be read. Additionally the Flash player can read content from other sources and even make direct socket connections if the destination provides a suitable XML policy file.
Furthermore some types of web content like CSS are assumed to be public libraries and can be included across domains [6].

2.3 Origin Definition

Different parts of the browser use different origin definitions. For network access three parts of the Uniform Resource Locator (URL) are used: the scheme (e.g. http or ftp), the host name (like www.example.com.au) and the port number (e.g. 8080). Cookies for example just use the host names as origin. [7]

3 DNS Rebinding Vulnerabilities

Common browsers' same origin policies (SOP) refer (beside to protocol and port number) always to the host names, but actually the network access is performed with the help of IP addresses. When the browser starts loading network content, the host name first gets resolved by the DNS system and then the request is sent with the IP address as technical and routable destination. The origin of the content will be still determined as DNS host name. The SOP can only be effective until there is no mismatch between the DNS host name and the technical IP address (apart from protocol and port number). And this is a crux of the SOP.

3.1 Standard Rebinding Vulnerability

The simple standard rebinding attack uses JavaScript, Java Virtual Machine (Java VM, JVM) or Flash player to connect to different IP addresses with the same host name.

3.1.1 Multiple A Records

This kind of vulnerability was already published in May 1996 [8]. An *A record* is a specific type of record in the DNS database. It associates a DNS host name with a single IP address. Multiple A records with the same host name are used to send the DNS-requesting client a complete list of associated IP addresses. Although it is not standardized, the DNS client may cyclically run through the list and use the next IP address for resolving. The so-called round-robin mechanism provides load balancing and failover.
With multiple A records of an attacker's host name in the DNS system the Java VM could be confused: while the malicious Java applet is loaded from the first IP address, the applet itself starts connecting to the same host name. Because of the round-robin usage of A records, this second connection is established to the second IP address, which was chosen by the attacker to address the victim target. The Java VM permitted this attack, because the target's IP address belongs to one of the origins A records. With Java applets this specific type of attack is not possible anymore because the current Java SOP focuses directly on the network layer: connections can only be established to IP addresses the applet comes actually from.
With JavaScript a very similar attack is still possible: the malicious JavaScript code needs to initiate a second connection to the attacker's server. The attacker only has to refuse this connection request by sending a TCP RST packet[2] and the browser is forced to connect to the second IP address. This is not an

[2]A TCP reset packet (TCP RST) is commonly sent by a host when a client (here: browser) tries to establish a TCP connection to a closed port, i.e. a port which no service (here: HTTP or HTTPS) is listening on

unspecified bug of browser software, it is the idea of round-robin DNS to obtain failover. Thus the browser continues working with the new IP address masked by the origin host name, and the malicious script loaded from the attacker's IP address can connect to the victim's IP address. [9].

3.1.2 Time-Varying DNS

Instead of multiple A records the attacker can also provide the authentic DNS record with a minimal time-to-live (TTL). If this field is set to zero, the resolving IP address is discarded immediately after the first usage.
Like described above JavaScript tries to reconnect to the origin but the browser has to re-resolve the DNS host name. In the meantime the attacker has modified the DNS record and swapped the authentic IP address to the victim's IP address. Now the malicious script can use the XMLHttpRequest or uses frames to connect to the victim, because the SOP is not violated regarding the DNS host name.

3.1.3 Weak Countermeasure: Browsers' Pinning

Browsers' developers have basically realized the DNS rebinding vulnerabilities and have introduced the today's commonly used countermeasure: the *pinning* mechanism. Most of the current browsers perform pinning of IP addresses to host names. This should prevent referencing to multiple IP addresses when scripts access to a single host name. But these pinning policies are not strictly applied, because round-robin DNS and dynamic DNS (DDNS)[3] need still be supported by the browsers. Sidestepping the pinning mechanism is easily: the current Internet Explorer 7 or the prior version usually pins IP addresses to host names for 30 minutes. But when there are multiple A records and the server is not available at the current used IP address, it will use the next IP address after one second. Firefox 1.5 and 2 pin IP addresses to DNS entries for 60 to 120 seconds. With JavaScript it is possible to forecast the exact expire time. With multiple A records it is also possible to reduce this time to one second. Opera 9 pins for about 12 minutes, but with connecting to a closed port the pin releases after a few seconds. Safari 2 only pins for one second. In this case forcing HTTP to closing the connection prevents the browser from reusing the existing TCP connection to the first IP address. (The duration details are taken from [6]).

[3]Dynamic DNS (DDNS) can be used to resolve DNS host names to dynamic allocated IP addresses. Every time the ISP allocates a new IP address to a host, this host can initiate an DNS record update. Ensuring the propagation of updates the time-to-live value of these records are short.

3.1.4 Behavior of Flash Player Plug-in

The current Flash players allow the active content in the form of small web format(SWF) files (or SWF "movies") to establish TCP sockets to any host. The only condition to fulfill is the destination holds and serves an XML policy file which allows the access by specifying the movie's origin. Up to the Flash player version 9.0.48.0 which was released about mid of 2007 the SWF rebinding vulnerability can be used as follows:

1. The web browser loads (unnoticed) an embedded SWF file from the attacker's site.
2. The SWF movie opens a socket to the attacker's site, bound to the authentic IP address of the attacker's server and requests the policy file.
3. The attacker's server responds with the XML policy file e.g. like this:

```
<?xml version="1.0"?>
<cross-domain-policy>
<allow-access-from domain="*" to-ports="*"/>
</cross-domain-policy>
```

4. The attacker changes his site's A record and substitutes his real IP address with the victim's IP address
5. The malicious SWF movie connects can connect to any port number of the victim, because the policy seems to come from the same host.

These versions of flash players perform no pinning and the cross-domain policy method is also vulnerable to rebinding attacks, as we can see. The only restriction is that it depends on the port number of the policy file service: in case the policy is served on port numbers below 1024, the Flash player can even connect to ports below 1024. Otherwise only ports with numbers starting at 1024 can be used for socket connection. Note that the above described attack can be performed even with the limited capabilities of a

phisher.
Recently (December 2007) Adobe reacted and released the Flash player 9 update 3, version 9.0.115.0. Several security enhancements were made and also this kind attack is not possible anymore: the SOP applies now to the IP addresses, i.e. the source IP address of the policy file must match with the one used for socket connections. In despite of this improvement, most of the browsers currently might work with the Flash player 9 without Update 3. But the auto-update notification checks for updates every 30 days (by default) when the Flash player is running and notifies the user when new updates are available.

3.2 Multi-Pin Vulnerabilities

There are other vulnerabilities to enable rebinding attacks: while the browsers and the plug-ins usually maintain their own and separated DNS database they use for pinning, an attacker can use the lack of replication between the browser and the plug-ins. Generally the browser allows JavaScript accessing the plug-ins and vica versa. This results in the potentiality of accessing objects from different origins. Concerning this matter every plug-in has its own peculiarity.

3.2.1 Java

The Java VM itself is not vulnerable. It maintains its own pinning separated form the browser. While with Java it is possible to use TCP sockets directed to the applets origin, the SOP is related to the actual source IP address of the applet and not to a DNS host name. This avoids rebinding attacks. However problems occur with LiveConnect, which can be seen as the glue between JavaScript and the Java VM. The Java standard library can be used by scripts without loading an applet. While JavaScript uses the browser's pinning and connects to the attacker's server, the Java VM accessed by LiveConnect does a second DNS resolve. Corresponding to the other rebinding attacks the attacker is able to change the DNS entries in the meanwhile.
Another weakness is the combination of applets with the usage of HTTP proxies. The Java VM loads the applet via a proxy with the help of the host name. In this case the proxy performs the DNS resolution. Once the applet is started and tries to connect to a socket, the Java VM performs another DNS resolution. Also during this scenario a usual rebinding attack is possible.
A third vulnerability occurs when a Java applet is embedded with the attribute "MAYSCRIPT" by a relative path. In this case the HTML page is first loaded from the target server, the browser pins to the target server. Then the Java VM loads the malicious applet and performs a second resolve which points to the attacker's IP address. Now the applet can make the browser via JavaScript to start XMLHttpRequests to the target's address.[6]

3.2.2 Flash

Basically the Flash player is vulnerable to multi-pin attacks. It does not pin DNS bindings at all, but however it would not lead to more security: the SWF movies are first downloaded by the browser which pins the DNS name to its resolved IP address. Then the browser delivers the movie and additionally its origin in the form of the host name. When the SWF movie tries to make a socket connection, it resolves the host name a second time.
Since the new Flash player 9 update 3 a stricter cross-domain policy handling is enforced. Cross-domain policy files need to be served by the socket connection destinations. To avoid the procedure of a rebinding at a moment when the policy file has been received and the socket connection is not yet initiated, the Flash player allows socket connections only to IP addresses that already have served a suitable policy file.

3.2.3 Other Plug-Ins

The Adobe Acrobat plug-in restricts general network communication to the SOAP protocol but allows access by document origin. "often" ([6]) the user is informed before accessing the network.
Microsoft's Flash plug-in pendant Silverlight also permits network access but uses the browser to make the request. That is why it also uses the browser's DNS pinning.

4 Attacks

The above mentioned vulnerabilities can be used by attackers to perform several kinds of attacks. Generally they can be put into three categories: firewall evasion, IP hijacking and dynamic pharming.

4.1 Firewall Evasion

The firewall's assignment is to restrict traffic between different trust zones. A usual scenario is the protection of an internal net against connections coming from the Internet. Firewall evasion tries to bypass the firewall by using an outgoing connection which was established by a user of the inner network and then uses the free access to the whole intern network by utilizing the client's browser (or plug-ins) as proxies (see figure 1). The attacker can e.g. spider the inner network. He needs to guess the internal IP address of interesting hosts. This is sometimes easy, but also depends on the network structure and the size of the IP network (class). However it is also possible to let the used client resolve guessable host names to the internal IP addresses. The attacker only has to make the client rebinding to a CNAME record[4]. Although this variant is not possible with exploiting the multi A record vulnerability, the time-varying DNS vulnerability is usable for this variant. Once the attacker got response from e.g. a port number 80, he can request the root document and follow all links and hyperlinks and filtrate all interesting documents being meant for internal use only. Many confidential content exists on intranet web servers of company networks without any additional security, assuming they are protected by the firewall.

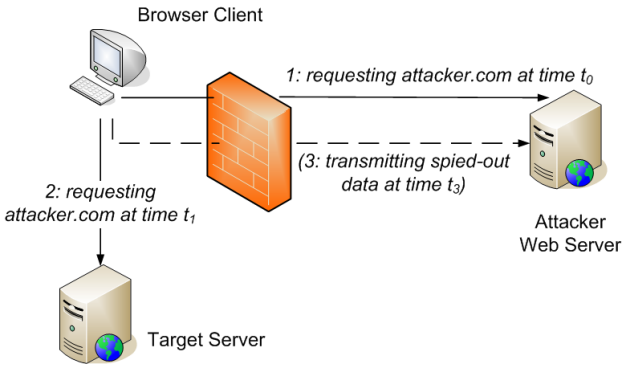

Figure 1: Basic firewall evasion

Additionally, attackers can try to compromise unpatched machines of the inner network. Often security updates are not performed at hosts that seem to be protected by the firewall. With direct socket connections toward these unpatched machines known vulnerabilities of e.g. the operation system can be exploited. Especially the attacker can try to attack the client's machine itself, because attacks coming from the localhost origin are often not avoided by software firewalls and other security mechanisms. Once an exploit was successful, the attacker might have permanently the opportunity to access the inner network, independently of the client's browser.
Another risk is the abuse of internal open services. Usually printer ports are open for free access from the intranet or file shares are readable and writable, so the attacker can exhaust the printers or use file shares for storing illegal content. Also network components with standard passwords can be used for reconfiguration. Even the recent security advances at routers against vulnerabilities called cross-site scripting (XSS) and cross-site request forgery (XSRF) do not prevent from rebinding attacks using socket connections.
Firewall evasion can be achieved via rebinding attacks by attackers having solely the restricted capacities of a phisher.

4.2 IP Hijacking

The second attack category includes attacks against hosts on the public Internet. The attacker abuses the implicit or explicit trust public services have in the client's IP address. One example is performing click fraud. While web advertisements are often paid per click, the publisher is interested in gaining as much clicks as possible at the advertisements' sites. The clicks are counted by requests coming from different IP addresses. So-called click-bots exist and make the members of their bot nets to send requests to the advertisements' site. Defending these flick fraud nowadays it is not enough to just send a simple request. The attacker first has to read a nonce – generated for each response separately – from the advertisement impression, coming with the first HTTP response from the site. The click request needs to contain this nonce. But also this assumed security advance allows bot nets, which use rebinding techniques, to perform click fraud, because the attacker can buy a high amount of advertising impressions and can convert them into clicks.
Another possibility for the attacker is sending spam. While many e-mail servers maintain blacklists of spam-sending IP addresses, the attacker can hijack hosts with IP addresses not listed by them. While most browsers do not allow JavaScript sending on port 25, it is possible to use clients with the older version of the Flash player to make socket connections. Like descried above the current update for the Flash player with the new strict cross-domain policy mech-

[4] A CNAME record is a name entry in the DNS system that points to another record, typically to an A record referenced by another host name.

anism prevents this attack. Nevertheless the current Java VM still allows this kind of attack. Additionally the attacker may use the client's mail relay and might also be already authenticated, because often the client runs the mail user agent at the same time and polls e-mails from the POP3 server – which authenticates for SNTP usage.
Also IP hijacking can be performed with the minimum of phishers' capacities.

4.3 Dynamic Pharming

A dynamic pharming attack affects some security goals very hardly: the attacker is able to eavesdrop on sensitive content, forge transactions, key log secondary passwords and so on, regardless of the authentication mechanism between client's browser and server. During pharming attacks the adversary causes DNS to answer queries with either his IP address or the legitimate server's IP address – depending on the situation, the state of the attack. The typical "time-varying rebinding" technique is used. The attack can be described in six phases (see also figure 2):

1. The client browser tries to connect to the victim's page, e.g. a login page of an online bank. Because the IP address is not known (not in the cache) the client sends a DNS request to the name system the adversary controls. The response is a DNS record with the right domain name but the adversary's IP address and an information that the client should not cache this result (TTL=0).

2. The client loads the malicious web page from the adversary server. If the adversary implemented an SSL web page, the user gets a warning because the certificate cannot be proper but may be self-signed or signed by unknown root CAs. Usually the user ignores these warnings.

3. Meanwhile the pharmer can update his DNS entry to the real IP address of the victim's server.

4. Forced by the malicious web page (see listing 1) the client wants to load the web page into the unnoticeable iframe. Because the domain name is not in the cache, a new DNS query provokes an answer with the real IP address of the victim's web page.

5. The client loads the origin web page of the server and the well-known web page appears in front of the user as it usually does – just in an iframe. The user will ignore this detail and enter his password or the browser even will automatically login with a client-side SSL certificate or password from the password manager.

6. Now the malicious JavaScript can hijack the session with the real server, because the same-origin policy is not offended.

The additional task of the attacker (compared to firewall evasion or ip hijacking) is that the client's user need to be fooled. He has to believe he is visiting the familiar web page of a trusted application, e.g. the online bank or other sensitive systems. The above described attack cannot be performed with capabilities of a phisher, because initially the user has to receive the attacker's page when he is requesting the trusted page. For manipulating the client's DNS response the attacker needs pharmer's capabilities. Another vector could be the typical phishing mechanism to get the user to the attacker's malicious web site. In this case step 1 can be left out but the browser now keeps the attacker's host name as origin of the page. Subsequently the rebinding mechanism could bind the attacker's host name to the victim's server IP address. Further on the real web page could be loaded in an iframe, also having the attacker's host name as origin. Again the SOP is met. The URL in the browser's address field is now the attacker's one, but history has shown that many users cannot realize this, if the attacker's address is chosen trickily. This modified attack is also performable by attackers with only phishing capabilities. There is no need to manipulate a foreign DNS resolver anymore.

```
<html>
<body>
<script>
    ---MALICIOUS JAVASCRIPT CODE---
</script>
<iframe src="https://www.mybank.com/index.html">
</iframle>
</body>
</html>
```

Listing 1: Structure of malicious HTML page

The above described dynamic pharming attack uses the time-varying rebinding vulnerability in phase number 1. Alternatively, exploiting the multiple A record vulnerability is possible. In both cases the browser's pinning can be released by answering with a TCP RST message, if the browser sends the second request again to the attacker's server in phase number 4. Furthermore other variants of this attack are possible by exploiting the multi-pin vulnerability. For example the functionality of LiveConnect can be used

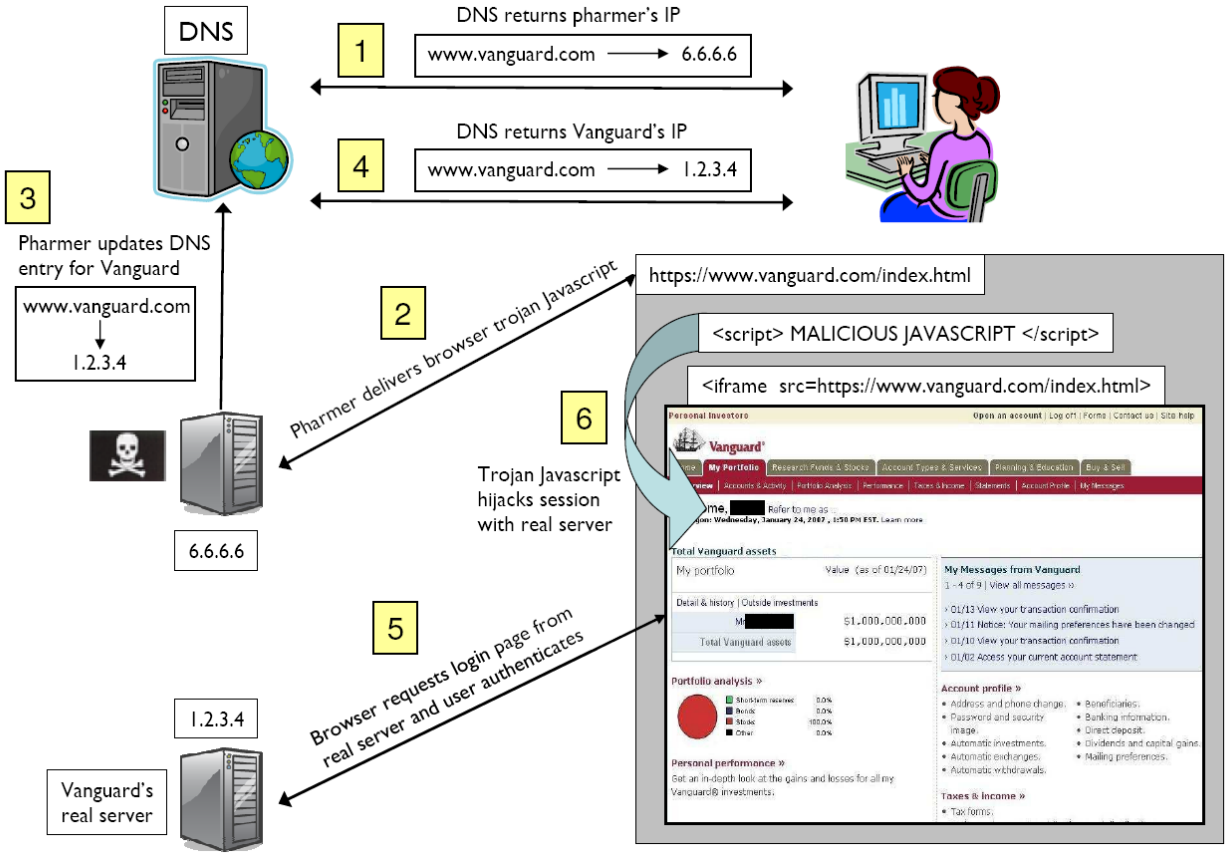

Figure 2: Dynamic pharming attack against www.vanguard.com [4]

as well as a Java applet, if the client's browser uses an HTTP proxy (as already explained in chapter 3.2).

5 Defenses

Defenses for above mentioned attacks can be applied at different points: at the browser, it's plug-ins, at the DNS resolver together with the firewall and at the servers.

5.1 Fixing at the Firewall

For preventing firewall evasion by rebinding attacks the firewall must not allow outbound traffic on port 53 to avoid resolving DNS host names by DNS servers not controlled by yourself. A second step is to prevent your DNS servers to resolve external names to internal IP addresses. DNSWALL is a small program that runs as a daemon at hosts with BIND servers and enforces this policy. By default it categorizes the standard private IP addresses as internal addresses. Some router-firewall appliance vendors might already have implemented a kind of this policy. For instance my up-to-date home DSL router is not vulnerable to rebind attacks as long as my intern hosts are forced to use this device for resolving.
Note that these methods do not prevent from IP hijacking or dynamic pharming, only firewall evasion can be stopped.

5.2 Fixing at the Plug-Ins

We have seen that socket-level network access permitted by plug-ins like Java VM and Flash player basically creates effective rebinding vulnerabilities. These weaknesses can be faced with changes of the plug-ins' behavior.

5.2.1 Flash

Since the Flash player 9 has got the update 3 since December 2007 and Adobe is heading to fully im-

plement the new strict cross-domain policies in the next update, the former massive vulnerabilities – using socket connections with action script 3 – seem now to be cleared out. Uploading and using manipulated policy files are hard to perform due to a standard cross-domain policy port, meta-policies, and the requiring of a policy from every IP address for socket connections.

5.2.2 Java

The Java LiveConnect vulnerability could be solved by installing a custom class that will handle the DNS resolution for the LiveConnect-used JVM. This class could have a native method for resolving DNS using the browser's DNS resolver. Actually this class must be installed into browsers, in this case the Java VM stays unchanged. Though performing this change the multi-pin vulnerability of LiveConnect in connection with the browser is removed.
Fixing the problem with Java VM behind a HTTP proxy other measures are needed to face the multi-pin attacks. Many used Java applets expect that it is allowed to use socket connections. A saver method would be the use of the CONNECT method to obtain a proxied socket connection to an external host, because proxies usually only allow CONNECT on HTTPS port. An advanced measure would be the usage of the HTTP host header for communicating the IP address between browser and proxy. This measurement would require complementary changes in the browser's and proxy's implementations and is hard to enforce.
Another approach could be the usage of cross-domain policies similar to the Flash player. This would also need a deployment of multiple steps but might be more successful and practicable as the other approaches.

5.3 Fixing at the Browsers and/or Servers

There are several long standing and recent approaches to prevent rebinding attacks by patching the browsers. Essential for their success is – like it is for the approaches fixing the firewalls or the plug-ins – the robustness, the effectiveness and the deployability.

5.3.1 Checking Host Header

HTTP 1.1 requires host header in the HTTP requests. These headers are used by proxies and web servers hosting more than one site's web pages. Because XMLHTTPRequest[5] is restricted from spoofing the host header, a server-side defense could be the rejection of requests with unexpected host headers.
This measurement is only leading to the desired result without allowed socket connections.

5.3.2 Smarter Pinning

Pinning parameters have to balance between robustness and security. For example a short pinning time might improve the robustness when multiple A records provide a round robin failover mechanism, unlike a long pinning time results to better security. But there are other, smarter parameters which also can be used: for example the IP address heuristic can be used: assuming multiple failover servers which have the same DNS host name have similar IP addresses. For example it might be a good trade-off if a rebinding within the same class C network (which holds IP addresses matching at the first three octets) is allowed after using the old address a very short time, while rebinding between different class C network addresses is not possible or only after a duration of several minutes. The Firefox extension NOSCRIPT might implement this feature in current releases.
Another, easier heuristic mechanism is avoiding of rebindings between public and private IP addresses. This measurement is only useful against firewall evasion in case the firewall protects a private IP network. It is already implemented in the Firefox extension LocalRodeo that performs JavaScript pinning.

5.3.3 Policy Based Pinning

The cross-domain policy mechanism can also be suggested for authorization of HTML and JavaScript content to access different network destinations. The policy files would be accessible in a standardized file at the root path of each web server. Like described above this technique is already used by Flash players.

[5] XMLHttpRequest is an API that can be used by JavaScript to transfer XML and other data between browsers and servers.

5.3.4 Host Name Authorization

Another approach is the authorization of host names. The owner of an IP address can use DNS PTR[6] records to release a white list with all host names that can be resolved to this IP address. Every owner of a server and it's IP address only adds honest host names to this white list and does not include the attacker's domain.

5.3.5 IP Address Based Origins

To gain more security by finer-grained origins a straightforward suggestion is to define origins by IP address instead of host names. One problem would be the behavior during failover situations, when today's long-lived applications are not able anymore to connect to new failover IP addresses. A second problem occurs when the browser is situated behind a proxy: the browser is not aware of the IP address and thus cannot determine and bind the IP address origin.

5.3.6 Locked SOP and Public Key Extension

In [4] the authors propose the update of the browser's some-origin policy. Two different stages are proposed. The "*weak* locked same-origin policy" distinguishes locked and non-locked web objects. Locked web objects are retrieved from legitimate servers via SSL. The browser uses for every object the tuple (scheme, domain name, port, validity bit). The validity bit is set when there was no problem with the certificate chain at all. On the other hand self-signed certificates and domain mismatch avoid setting the validity bit. This is independent of the user reaction of possible warnings when problems occur. Objects retrieved over HTTP have also not set the validity bit.
The weak locked SOP only allows access between locked web objects if the corresponding tuples exactly match. The attacker has no possibilities to perform a rebinding attack, because he has no valid certificate for the target domain and his validity bit is not set. But there is one significant exception: if the target domain uses an invalid X.509 certificate, the weak locked SOP provides no additional protection compared to the legacy SOP. Naturally also sites that are completely not SSL-protected are not protected.
The so-called "*strong* locked same-origin policy" goes on step further: a locked web object has only access to another one, if the origins' X.509 certificates have the same public keys. The tuples (scheme, domain name, port, public key) are compared. The strong locked SOP resists phishing, pharming and active attacks against locked web objects as long as the adversary does not know the corresponding private key of the victim.

[6]The DNS PTR record is normally used for reverse lookups to ask for a corresponding host name of a known IP address.

6 Deployability and Conclusion

Above mentioned concepts can only be considered as solutions if they are deployable without many changes of the software and the protocols. Changed software also must be compatible to the legacy versions.
To eliminate vulnerabilities of the plug-ins, a kind of strict cross-domain policy mechanism – which is already tracked by the Flash player developers – could be developed and advanced especially for plug-ins that allow direct socket connections. Probably a cross-domain policy *standard* for socket-connecting plug-ins could be drafted. Independently firewall evasion can be more simply avoided by strict control of the firewall and the DNS resolver (e.g. with DNSWALL). This would also mitigate the vulnerabilities which occur when browsers and plug-ins inter-operate. The alternative would be a common pin database for browsers and all plug-ins, which still does not solve the proxy problem and seems also hard to deploy.
The weak locked same-origin policy seems to be easy to deploy: only small changes have to be made at the browser's side to change the SOP. The authors have made an HTTPS server survey to make sure the new policy will not "break the web". Results have shown that only 0.0005 % of the SSL domains would make problems due to administrative faults of the servers and certificates. Web servers and the HTTP specification can remain unaffected. The survey shows different results for the strong locked SOP: an order of magnitude more servers would cause problems with this policy. And – unlike problems mentioned above – these ones are not the result of misconfiguration. For example, different objects of the same domain may have different public keys, key updates can be caused by certificate expiration, content may come from different servers, etc. To face this problem the authors propose policy files for supporting multiple keys and key updates. These files could be send by the server like they send the favicon.ico file. In this file all allowed public keys for the domain could be listed and signed for security reason. Subdomain object sharing and key revocation are also deployable with the

approach of the strong locked SOP with policy files. In this case also small changes have to be made at server's side, but the HTTP and HTTPS specifications need not to be changed.
These approach could mitigate the dangers of dynamic pharming by changing the browser security model. However, arguable is their ability to save today's and future web applications summarized by the term "web2.0": the trend of web sites goes to combining web content originated from absolutely different sources. These sources can sometimes not be trusted. The approaches like the locked SOP are based on SSL and put the focus on the connection between browser and web server. This assumes that the user has to trust the web server he connects to – including all the content it is actually providing. Unfortunately web2.0 servers may not be able to guarantee the harmlessness of its content.

References

[1] Stamm, S., Ramzan, Z., Jakobsson, M. "Drive-By Pharming", December 2006, http://www.cs.indiana.edu/pub/techreports/TR641.pdf

[2] Dhamija, R., Tygar, J.D., Hearst, M. "Why Phishing Works", 2006, http://people.seas.harvard.edu/~rachna/papers/why_phishing_works.pdf

[3] "W3C Document Object Model", http://www.w3.org/DOM/

[4] Karlof, C.K., Shankar, U., Tygar, D., Wanger, D. "Dynamic pharming attacks and the locked same-origin policies for web browsers" May 2007.

[5] "Mozilla Port Blocking", http://www.mozilla.org/projects/netlib/PortBanning.html

[6] Jackson, C. et al. "Protecting Browsers from DNS Rebinding Attacks" October 2007.

[7] "The Same Origin Policy", http://www.mozilla.org/projects/security/components/same-origin.html

[8] Dean, D., Felten, E.W., Wallach, D.S. "Java security: from HotJava to Netscape and beyond" in IEEE Symposium on Security and Privacy. Oakland, California, USA May 1996.

[9] "A small contribution to the current 'hacking the intranet with JavaScript' theme", http://shampoo.antville.org/stories/1451301