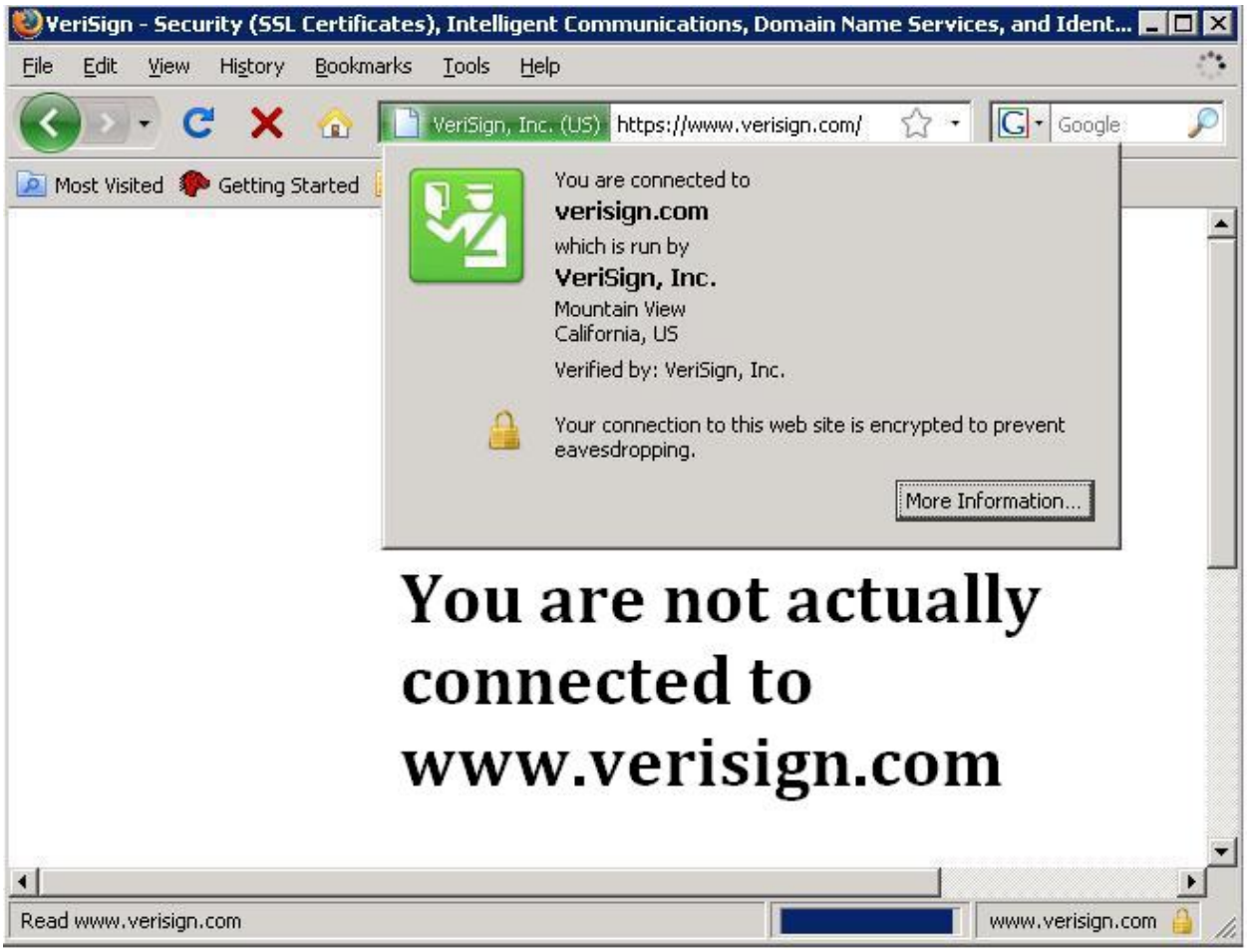
Sub-Prime PKI:
Attacking Extended Validation SSL

By Michael Zusman & Alexander Sotirov

July 2009



Abstract

Jackson and Barth[1], in their paper "Beware of Finer-Grained Origins" (May 2008), describe a number of HTTP session attributes beyond Same Origin Policy(SOP) that are used by web browsers to make security decisions. One of the attributes that the paper discusses is the type of SSL certificate presented by a web server when negotiating a secure connection. Modern web browsers support both domain validated (DV) and extended validation (EV) SSL certificates. EV SSL certificates were created to combat phishing and other web based spoofing attacks which succeed in spite of the DV SSL model for web site identification. The EV SSL model improves upon the DV model in two ways. First, certificate authorities enforce a more thorough off-line validation process in order to more accurately identify the agent requesting the certificate as the owner of the domain which the certificate is for. Second, when a certificate chains up to an EV root certificate authority (CA), EV-capable web browsers display the web sites identity information next to, or in, the address bar. This identity information is usually contained in a green badge, showing what certificate authority Verisign refers to as the “green glow.”

While Jackson and Barth describe a specific flaw in the Same Origin Policy that could allow an attacker to inject arbitrary JavaScript into an EV SSL page, this paper describes two other short comings in browser design that allow an attacker to *silently* Man-In-The-Middle (MITM) EV SSL protected web sites. In addition to these design flaws, this paper introduces a new attack, called SSL Rebinding, and revisits an old attack, browser cache poisoning.

Between SSL rebinding and cache poisoning attacks, this paper will show that the described design flaws in modern web browsers create a “weakest link” scenario, where high-assurance EV SSL certificates are permanently handicapped by their low-assurance DV counterparts. After reading this paper, the reader will see that until our browser vendors address these design issues, the “green glow” of EV SSL is less of an assurance than promised by the marketing materials of EV SSL vendors.

Background

Jackson and Barth disclosed their research at the Web 2.0 Security & Privacy conference on 22 May 2008. This period was the “calm before the storm” for the information security community, as speculation about the yet-to-be-disclosed Dan Kaminsky DNS flaw was about to begin running rampant. When the paper was originally released, a key requirement to launch the attacks described by Jackson and Barth was for an attacker to have a foot-hold between a client and server as a Man-In-The-Middle (MITM). At the time, such MITM attacks were mostly limited to insecure wireless networks and attacker controlled rogue access points. However, when the results of the Kaminsky DNS bug came to light weeks later in July, the long-held security expert assertion that DNS could be easily subverted on a mass scale was shown to be an easily implemented reality. Suddenly, a realistic defense was required for attacks that could be easily launched against large sections of the Internet, and targeted at prominent financial and ecommerce web properties.

With the details of the DNS bug known, security researcher Halvar Flake wrote on his blog that the Kaminsky bug was "why we have SSL,"[2] pointing out that one founding purpose of Public Key Infrastructure was to allow clients to verify the identity of a remote server in spite of an un-trusted domain name resolution system. This comment could be seen as a warning to those with a heavy reliance on SSL PKI. Now that attackers had an easy way to become the MITM for large portions of the Internet, SSL PKI was suddenly exposed as the only defense standing in the way of "perfect MITM attacks." [3]

In August 2008 at the BlackHat conference in Las Vegas, Mike Zusman (a co-author of this paper) disclosed that he received a signed certificate for the Microsoft site Login.Live.Com, for which he was not authorized by Microsoft. Zusman requested the DV SSL certificate from CA Thawte using the email address *SSLCertificates@Live.com* which he registered with the free Live.com webmail service. Since the CA used email as form of identification and a method of authorization for DV certificates, Zusman was promptly granted his certificate. Over the next five months, many more eyes and much scrutiny was cast on the SSL PKI.

Late in 2008, after a number of certificate authority mishaps and compromises made headlines on the Internet, another co-author of this paper, along with a team of international researchers, dealt what was thought to be the final blow to the SSL PKI.

On 31 December 2008, Alexander Sotirov, Jacob Appelbaum and a team of cryptography researchers[4] disclosed that they had used a previously known weakness in the MD5 hashing algorithm to generate a collision between a rogue CA certificate and a certificate legitimately issued by a trusted CA. This feat allowed them to generate their very own signing key, capable of generating SSL certificates that would be trusted by all major web browsers. This rogue certificate authority, coupled with a flaw like that of the DNS bug, could be used to completely undermine the security of e-commerce and banking transactions on the Internet.

At the time, both the researchers and the commercial entities involved in their disclosure thought that EV SSL certificates were immune from this attack. Their logic was that since EV roots cannot use MD5 for certificate signing (by mandate of the CA/B Forum), that the same attack could not generate a rogue CA capable of signing EV certificates. They were right. However, they did not realize that non-EV certificates signed by a rogue CA could be used in attacks like those described by Jackson, Barth, and this paper.

In late February 2009 at the BlackHat conference in Washington DC, SSL PKI received yet another blow. Security researcher Moxie Marlinspike disclosed two methods that he used to defeat SSL in modern day web browsers. Both methods involved spoofing various visual cues that browsers use to indicate a secure session to its human user. Marlinspike's disclosure prompted commercial certificate authority Verisign to issue a press release offering advice on how to protect oneself from such MITM attacks. The press release, titled "VeriSign Offers Recommendations on How to Protect From Man-in-the-Middle Attacks", tells users to "look for the green glow" which would indicate that a web site has presented an

EV SSL certificate. Based on this press release, it seems that between May 2008 and February 2009, the great work done by Jackson and Barth was off the radar of most of the information security community.

New Attacks Against EVSSL Enabled Clients

In the previous section of this paper, it was established that the current PKI is vulnerable to a variety of attacks that result in DV certificates falling into the hands of individuals who are not authorized to have them. The following sections detail new attacks that attackers holding valid DV certificates can launch to silently compromise HTTP sessions protected by an EV SSL certificate.

SSL Rebinding

SSL Rebinding is an attack against SSL involving a rogue server which uses a combination of SSL certificates to manipulate client behavior and bypass security mechanisms. While this paper focuses on attacks against EV SSL, this attack could also be launched against poorly written SSL clients in order to bypass strict validation of DV certificates.

In the case of spoofing EV SSL security indicators, a remote server can switch from "server mode," where it serves a valid DV certificate to the client, to "proxy mode," where it proxies TCP packets and allows the client to handshake with a server hosting a legitimate EV SSL certificate. SSL rebinding attacks allow the attacking server to capture HTTPS requests from the browser containing user credentials, authenticated session tokens, and other sensitive data sent by the client. In response to the captured request, the attacker responds with a redirect and a *Connection: Close* HTTP header. This response causes the browser to replay the same request over a new TCP connection, which the attacker proxies to the real server allowing the client to terminate SSL with the real EV SSL certificate. Since these connections terminate SSL with the real web server, the web browser shows the "green glow" indicating a secure connection with the desired host.

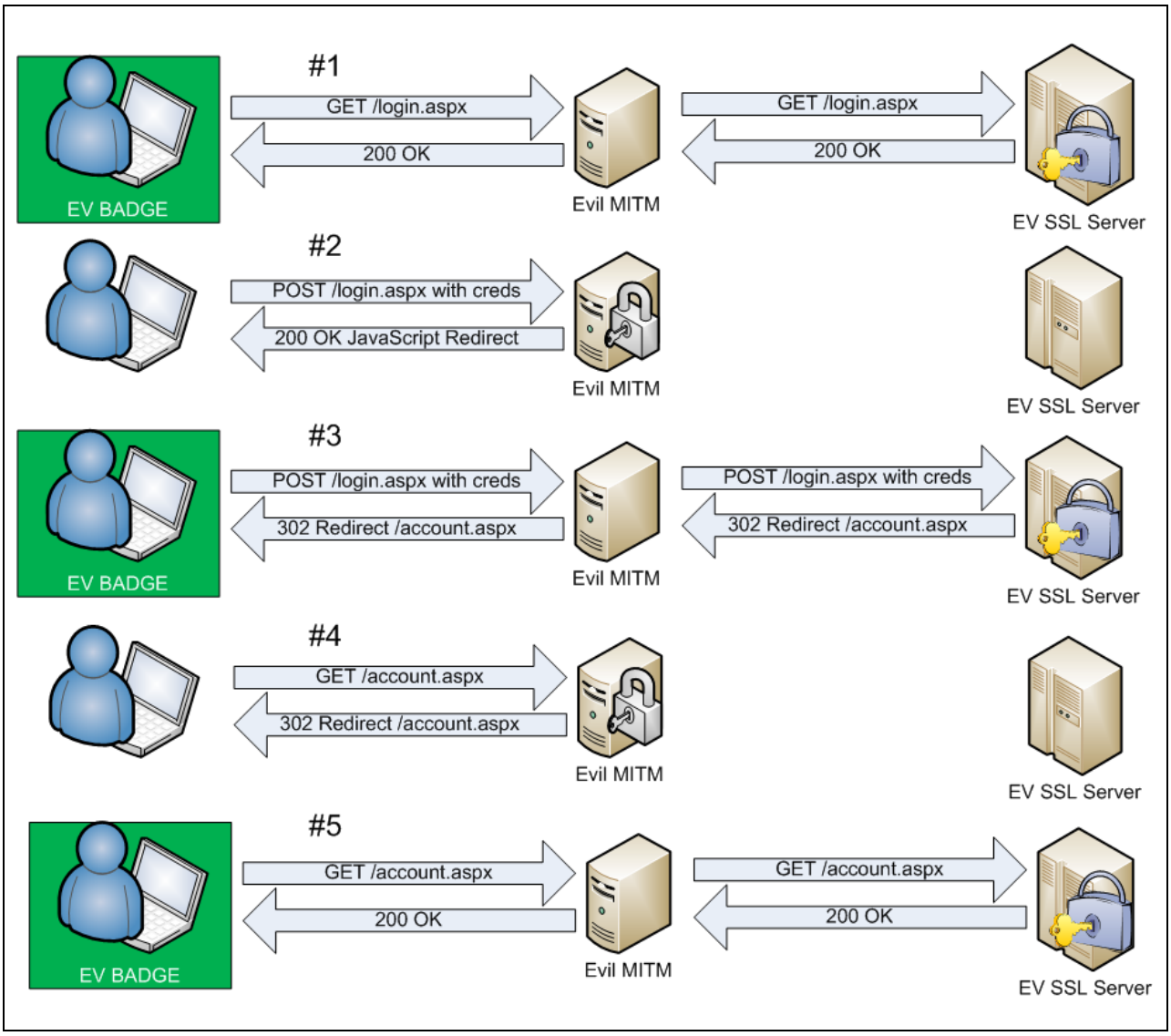


Figure: The request and response flow of an SSL Rebinding attack

Request #1: GET /login.aspx

The MITM proxies TCP packets and lets the client perform an SSL handshake with the real HTTP server and its EV SSL certificate. The HTTP server sends back the login page.

Response #1: 200 OK

The web browser renders the login page, and has the "green glow" indicating an EV SSL session.

Before the user enters credentials, the MITM closes the TCP connection between itself and the client, and itself and the HTTP server.

Request #2: POST /login.aspx

When the client submits their credentials, the browser must first open a new TCP connection to the MITM. For this connection, instead of forwarding the connection to the real HTTP server, the MITM terminates SSL with a non-EV domain validated SSL certificate. This certificate is valid, so no browser warnings are generated. The browser connects and sends the HTTP request along with user credentials to the MITM.

Response #2: 200 OK

Once the MITM has captured the HTTP request, he must build an HTTP response to send to the client. This response must cause the browser to do two things:

1. Close the TCP connection to the MITM

2. Replay the exact same request as request #2.

Request #3: POST /login.aspx HTTP/1.1

The last response from the MITM has told the browser to replay request #2. This response can be in a number of formats, which will be discussed in detail later in this paper. However, part of the response was a "Connection: Close" HTTP header. This header causes the browser to close the TCP connection with SSL terminated at the proxy, and open a new TCP connection. The proxy again forwards this new TCP connection all the way to the real HTTP server, allowing it to terminate SSL with the EV SSL certificate.

Response #3: 302 Redirect

Now that the browser again has completed the SSL handshake with the real HTTP server, the browser shows the "green glow" and has sent the credentials to the real server for authentication. The real server now generates the real HTTP response, and sends it to the client via the MITM. The MITM cannot see the contents of the response.

Request #4: GET /account.aspx

Since the MITM does not allow persistent TCP connections to last very long between the client and the real HTTP server, the browser will eventually open a new TCP connection to the MITM. Again, the MITM will terminate SSL using a good domain validated SSL certificate and intercept the browsers request. Since the user is now authenticated, the MITM now has captured a valid session cookie for that user.

Response #4: 302 Redirect

Again, the MITM directs the browser to close its TCP connection with the MITM, open a new connection, and replay request #4.

Request #5: GET /account.aspx

The browser now replays request #4 and the MITM lets it terminate SSL at the real HTTP server

Response #5: 200 OK

The HTTP server now responds to request #5 and sends the client the expected data. Once again, the "green glow" is restored.

The "Every Request" Problem

The common methodology for proxies that intercept SSL encrypted traffic is to intercept every connection and always terminate SSL. For EV SSL, this idea that we must intercept every SSL connection does not work. The problem is that since some communication has SSL terminated at the real web server, the proxy does not have access to all HTTP data going between the client and the server; it can only see what it terminates itself. Because of this, the proxy has no easy way to correlate an intercepted HTTP request with the resulting replay of that request over a new TCP connection and EV SSL channel. If the proxy were to intercept every TCP connection and terminate SSL, the browser would get stuck in an endless loop.

To solve this problem, the proxy must attempt to correlate EV SSL encrypted connections with their preceding intercepted HTTP request. While it may be possible to use TCP connection characteristics such as sequence number or source port, or even timing of connections, to correlate two arbitrary TCP connections, there are two much simpler methods.

Method #1: TCP Connection Counting

The MITM can keep a running count of all TCP connections from each client. By doing a simple modulus operation on the running total, the MITM can alternate between terminating SSL and not terminating

SSL for each connection. In this method, the capture of a specific HTTP request is not guaranteed. However, in testing, this method was found to be simple and effective.

Method #2: Web Browser TCP Connection Exhaustion

Most web browsers have a maximum number (N) of simultaneous connections that they will make to a given web site. It is possible for a MITM proxy to accumulate and hold open N-1 connections, such that the web browser can only send HTTP requests over the one remaining connection. Since there is only one connection available, the proxy can simply alternate between terminating and not terminating SSL.

The Flicker Effect

When a web browser switches from a non-EV connection to an EV SSL connection, and ultimately back to a non-EV connection, the "green glow" in the address bar will quickly appear and disappear, creating a "flicker effect." This behavior can be seen in the wild on Microsoft's Live.com web site. When a user requests the login page for Live.com, it is served over HTTP, so no green glow is displayed. However, the login form POSTs to https://login.live.com, which uses an EV SSL certificate. When the user POSTs their credentials, they quickly see the address bar turn green. After a successful authentication request, the browser is redirected by a 302 response to https://acounts.live.com, which does not use an EV SSL certificate. The address bar now turns white again.

The flicker effect is not a necessarily a technical problem, but it does pose a question for the user: what should the user expect to see, and when?

JavaScript Not Required

The finer-origins paper says that a MITM can inject JavaScript into a browser taking part in an EV SSL protected session. While true, this is not the most efficient way to leverage SSL rebinding from an attackers perspective. Injecting JavaScript into a browser does not necessarily make for a persistent attack. The use of pop-up and pop-under windows is required to allow an attackers code to remain running as the main browser window renders new HTML served over an EV SSL connection. This adds complexity to the attack, as the attacker must find a way to evade pop-up blockers built into modern browsers. Additionally, FireFox plug-ins such as NoScript can prevent JavaScript execution in the web browser, rendering any key logging, pop-up, or redirection code useless.

Exploring HTTP Redirects

While a silent SSL rebinding attack does not need JavaScript, it still needs to redirect browser requests, which include both HTTP GETs and HTTP POSTs. The redirect of an HTTP GET without JavaScript is trivial. This can be done using both the META-REFRESH tag as well as an HTTP 302 response. Both methods allow the attacker to redirect the browser and have any data sent in the query-string re-sent as well.

An HTTP POST, on the other hand, is not so simple. A 302 response to a POST request will cause the browser to send a GET request to a MITM controlled URL. While the MITM can embed the intercepted POST data into the new URL as query-string data which the browser will send, the web application will most likely disregard the query-string data. The MITM cannot rely on the web application to read from both the query-string and POST data.

For browsers such as Internet Explorer, in which users have no easy way to disable JavaScript, the easiest way for a MITM to redirect a POST is to generate an HTML page. Sent to the client as an HTTP 200 response, the attacker-generated HTML will contain a hidden form with all of the intercepted POST data from the original request. Along with the form, the MITM sends one line of JavaScript to automatically re-play the same POST request. This POST is sent over an EV SSL connection, and once again the “green glow” is restored.

However, this does not help the attacker if the victim is running Firefox 3 with NoScript. The page will render in the browser, but he has no way of triggering the automatic re-POST. The user will end up staring at a blank screen - with no "green glow."

The HTTP 307 Response Code

http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

The 307 response functions almost identical to the 302 redirect - except for the fact that a browser receiving a 307 in response to a POST will replay the POST request to the URL specified in the Location header, and send the POST data. It is important to note some differing behaviors between Microsoft Internet Explorer 7 and FireFox 3. In most cases IE7 will violate RFC2616 by replaying the POST request and sending the data without confirming with the user that this is desired behavior. FireFox 3, on the other hand, will present the user with a simple dialogue explaining that the page is being redirected and asking them to confirm that this is okay. Additionally, the authors of this paper noticed a bug in Internet Explorer 7, which fails to send the POST data along with the replayed request if the URL specified in the Location header is identical to the original POST URL. However, this bug is easily worked around by appending a hash-sign (#) to the end of the URL in the 307 response Location header.

Proxy Request Forgery

In an SSL rebinding attack, the attacking proxy only sees HTTP requests coming from the client. Since all communications with the real web server must terminate SSL with the real server, the proxy cannot see any HTTP response data. However, since the proxy can capture authentication credentials and valid session tokens for an authenticated session, there is nothing stopping the MITM from silently replaying these requests.

EV Cache Poisoning

SSL rebinding requires a number of conditions to be true. First and foremost, the attacker must have a foot hold as a man-in-the-middle. However, just because an attacker is ready to MITM a particular web site, he still needs the victim to browse to that web site and perform a transaction. In the case of a compromised gateway in an Internet café, some users might be savvy enough to avoid performing any sensitive transactions over the un-trusted network. On the other hand, these same users may feel completely safe performing a simple Google search.

Due to the failure of modern browsers to properly implement both same origin policy and an SSL mixed content policy, it is possible for the MITM to leverage plain-text HTTP transactions to poison the browsers cache with spoofed SSL content.

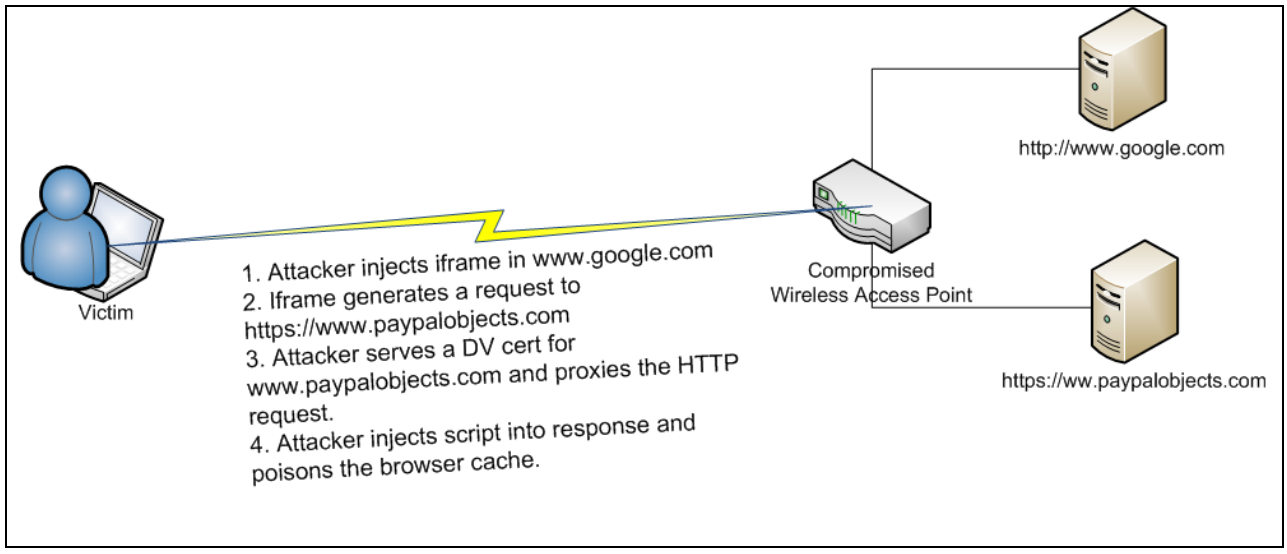


Figure: The victim gets their cache poisoned on the un-trusted network.

The above diagram shows how a cache poisoning attack might unfold. The victim, knowing that the network is not to be trusted, avoids checking their PayPal account. However, they still perform a Google search through the compromised gateway.

The attacker injects an iframe in one of the plain-text HTTP responses from Google. This iframe causes the browser to request a script (foo.js) from *https://www.paypalobjects.com*. Since the attacker holds a valid DV SSL certificate for *www.paypalobjects.com*, the browser makes this request without warning the user. The attacker proxies this request, and embeds his JavaScript payload into the script before sending it to the client.

In addition to the injected payload, the attacker also adds the following HTTP headers to ensure that foos.js is cached:

Cache-Control: Public
Expires: Friday, 30 Oct 2011 14:21:21 GMT

When the victim returns to a trusted network, and points their browser to *https://www.paypal.com*, the poisoned foo.js is pulled from their cache. At this point, the browser shows the "green glow," yet the attacker specified JavaScript has compromised the browser. If the browser contacts the original site to check if the content has been updated, the If-Modified-Since header of the request will be set to 2011 and the server will return a 304 Not Modified response.

Browser Design Flaws

The problem with Mixed Content

As was previously stated, modern web browsers support both DV and EV SSL certificates. However, the advent of high-assurance EV SSL has not resulted in degradation in the trust of DV certificates by these web browsers. This means that browsers trust content equally, whether it is served over DV SSL or EV SSL. Subsequently, these browsers allow DV and EV SSL protected content to co-exist *on the same webpage*, while still showing the "green glow." This behavior is counter intuitive to the ideas behind EV SSL; users are supposed to trust EV SSL *more* than standard DV SSL, while the browser itself treats them equally.

One example of a site that uses mixed EV and DV SSL protected content is PayPal. The site *www.paypal.com*, which uses an EV SSL certificate, pulls JavaScript files from the domain *www.paypalobjects.com*, which uses a DV SSL certificate. Even though *www.paypal.com* pulls content from a site that is less trusted by the EV SSL model, recent versions of Microsoft Internet Explorer, Mozilla Firebox, and Apple Safari web browsers all show the EV identity information, also known as the "green glow."

Creating a Mixed Content Policy

Web browsers need a policy that disallows a web site protected with an EV SSL certificate from loading content from non-EV sites. In a case like PayPal, the browser should fail closed and not show the "green glow" and EV identity information. A policy of this sort would be a boon for certificate authorities, as it would pressure sites who host content on EV protected sites to adopt EV themselves. A failure to adopt EV would drive their customers toward solutions that allow them to retain their EV status in the browsers of their visitors.

A perfect example of such a web site is Google Analytics, which hosts JavaScript linked to by millions of other web properties. If such a policy existed, and Google Analytics did not use an EV certificate, a company that relies on both EV certificates and the service would be forced to make a decision: use Google Analytics, or benefit from EV SSL. If Google would not migrate to EV, the company would risk losing subscribers.

Unfortunately, the easiest solution for solving the mixed content policy is not a feasible one. The best solution would be to no longer trust the low-assurance DV certificates. Such a move by browsers would "break the Internet", since most businesses and SSL enabled web sites rely on DV certificates being trusted, and would not be willing to spend the larger amounts of money required for EV.

Fixing Same Origin Policy

In addition to the creation of a mixed content policy governing the use of the EV SSL indicators in web browsers, the original problem described by Jackson and Barth is dangerous. Browsers do not use the type of SSL certificate served to distinguish between different content origins. An easy solution might be to treat different certificates for the same host name as being from different origins, regardless of whether they are EV or DV. Unfortunately, some organizations have architectures that depend on multiple, distinct key pairs for a single domain name.

Jackson and Barth recommend implementing a new protocol, such as httpev in addition to https, since the protocol is one of the attributes used in determining origin. While such a change would serve the purpose, it would also cause problems for sites that rely on hard coded https links. The authors of this paper feel that such a parameter can be treated as an internal flag by the browser, with no need for any new external representation.

Conclusion

The state of the SSL PKI is dismal. EV SSL has altered the model for trust and identification on the Internet by introducing tiers of trust and assurance. Proper adoption of such a drastic change goes much deeper than just browser chrome. While the user interface enhancements showing web site identity information are a good thing, at this point they only give users a false sense of security. For the potential of EV SSL to be realized, and to make our PKI scalable enough to handle a day when EV SSL can no longer be trusted, web browsers, SSL clients, and SSL client APIs must provide the mechanics to plainly communicate to users the difference between tiers of SSL trust and assurance.

References

1. Jackson, C. and Barth, A., "Beware of Finer-Grained Origins", in proceedings of Web 2.0 Security and Privacy (W2SP 2008), 2008. http://crypto.stanford.edu/websec/origins/fgo.pdf

2. Flake, H., Internet blog posting http://addxorrol.blogspot.com/2008_07_01_archive.html. July, 2008.

3. Nigg, E., "~~Un~~trusted Certificates", Internet blog posting https://blog.startcom.org/?p=145. December, 2008.

4. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D., and de Weger, B., "Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate", in proceedings of CRYPTO 2009, also available at http://eprint.iacr.org/2009/111.pdf

Trademark Information