ALPACA Attack

ALPACA Attack - Marcus Brinkmann, Christian Dresen, Robert Merget, Damian Poddebniak, Jens Müller, Juraj Somorovsky, Jörg Schwenk, Sebastian Schinzel, alpaca-attack.com.

- Published: date not stated
- Original: <https://alpaca-attack.com/>
- Preserved from: https://alpaca-attack.com/ (live) on 2026-08-10
- 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.

ALPACA Attack

- Paper
- Q&A
- How to ALPN/SNI
- Updates!

News

- A big reevaluation of TLS libraries, TLS application servers, and a new internet scan by Jannik Hölling is now available in the Updates section!
- ALPACA will be presented at Black Hat USA 2021, USENIX Security Symposium 2021, and Real World Crypto Symposium 2022!
- Recommended articles: Ars Technica (Dan Goodin), Golem (Hanno Böck; German)

Introduction

TLS is an internet standard to secure the communication between servers and clients on the internet, for example that of web servers, FTP servers, and Email servers. This is possible because TLS was designed to be application layer independent, which allows its use in many diverse communication protocols.

ALPACA is an application layer protocol content confusion attack, exploiting TLS servers implementing different protocols but using compatible certificates, such as multi-domain or wildcard certificates. Attackers can redirect traffic from one subdomain to another, resulting in a

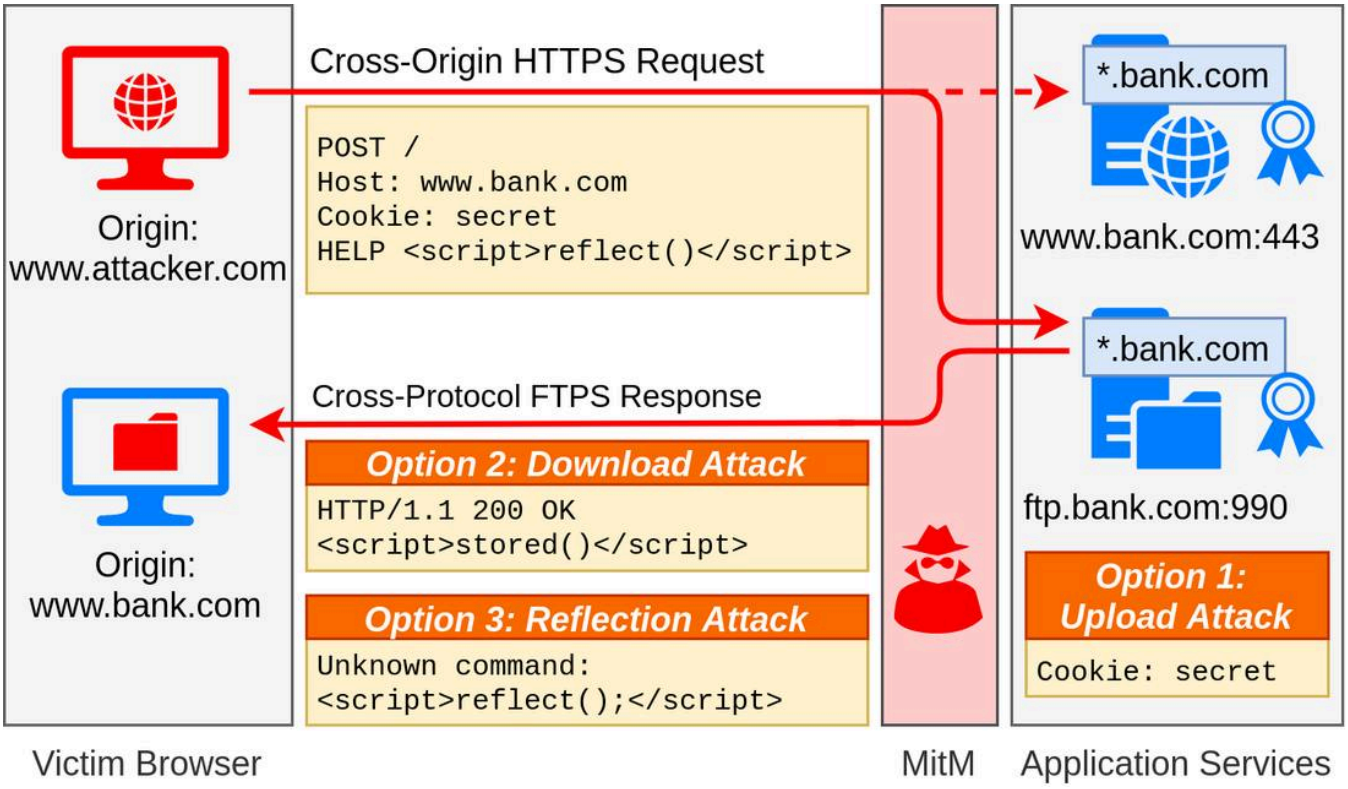
valid TLS session. This breaks the authentication of TLS and cross-protocol attacks may be possible where the behavior of one protocol service may compromise the other at the application layer.

We investigate cross-protocol attacks on TLS in general and conducted a systematic case study on web servers, redirecting HTTPS requests from a victim's web browser to SMTP, IMAP, POP3, and FTP servers. We show that in realistic scenarios, the attacker can extract session cookies and other private user data or execute arbitrary JavaScript in the context of the vulnerable web server, therefore bypassing TLS and web application security.

We evaluated the real-world attack surface of web browsers and widely-deployed Email and FTP servers in lab experiments and with internet-wide scans. We find that 1.4M web servers are generally vulnerable to cross-protocol attacks, i.e., TLS application data confusion is possible. Of these, 114k web servers can be attacked using an exploitable application server. As a countermeasure, we propose the use of the Application Layer Protocol Negotiation (ALPN) and Server Name Indication (SNI) extensions in TLS to prevent these and other cross-protocol attacks.

Although this vulnerability is very situational and can be challenging to exploit, there are some configurations that are exploitable even by a pure web attacker. Furthermore, we could only analyze a limited number of protocols, and other attack scenarios may exist. Thus, we advise that administrators review their deployments and that application developers (client and server) implement countermeasures proactively for all protocols.

Attack Overview



The image shows three possible ways for an attacker to use cross-protocol attacks against webservers, exploiting vulnerable FTP and Email servers: In the Upload Attack, the attacker exfiltrates authentication cookies or other private data. In the Download Attack, the attacker

executes a stored XSS attack. In the Reflection Attack, the attacker executes a reflected XSS in the context of the victim website.

Full Technical Paper (last update: 2021-06-09)

ALPACA: Application Layer Protocol Confusion-Analyzing and Mitigating Cracks in TLS Authentication, Marcus Brinkmann, Christian Dresen, Robert Merget, Damian Poddebniak, Jens Müller, Juraj Somorovsky, Jörg Schwenk, Sebastian Schinzel.

The artifacts are available at GitHub.

FAQ

I am an admin, should I drop everything and fix this?

Probably not. For the ALPACA attack to succeed, many preconditions need to be fulfilled. The generic attack requires a MitM attacker that can intercept and divert the victim's traffic at the TCP/IP layer. However, if you run application servers such as FTP and email on non-standard ports that are not blocked by browsers, you should make sure that you are not vulnerable to the web attacker variant of ALPACA that can affect users of Internet Explorer.

What can the attackers gain?

For the specific attacks on HTTPS described in the paper, the attacker can potentially steal cookies or perform a cross-site scripting attacks.

However, the potential consequences to the general ALPACA attack are dependent on the interactions of two unknown protocols, so any number of undesirable behaviors may be possible.

Who is vulnerable?

This is difficult to answer. The general flaw behind ALPACA is within the server authentication of TLS, so potentially all TLS servers are affected that have compatible certificates with other TLS services. In regards to that, all those servers have to be considered vulnerable. However, for practical purposes, this definition is not very useful, as the flaw is exploitable only in some cases. We therefore distinguish between vulnerable servers and exploitable services. Our analysis was limited to only a few protocols and a small number of implementations, so we can really only make clear statements for those. From our analysis, the following is generally true:

- Sharing certificates between a Webserver and an FTP server is almost always dangerous if an attacker has write access to the FTP server. It is sometimes dangerous if the attacker has no write access.
- Sharing certificates between a Webserver and an SMTP/POP3/IMAP server is sometimes dangerous, depending on the exact behavior of the server.

Here is a list of analyzed implementations in regards to their vulnerability (see Table 3 in the paper): Sendmail SMTP allowed reflection attacks that work in Internet Explorer when used over STARTTLS. Cyrus, Kerio Connect and Zimbra IMAP servers allowed download and reflection attacks

that work in Internet Explorer. Courier, Cyrus, Kerio Connect and Zimbra allowed download attacks that work in Internet Explorer. Microsoft IIS, vsftpd, FileZilla Server and Serv-U FTP servers allowed reflection attacks that work in Internet Explorer. And the same FTP servers allowed upload and download attacks that work in all browsers.

But even then there are interactions with analyzed and not yet analyzed protocols which makes a risk estimation difficult, since we believe that there is a large number of yet undiscovered vulnerabilities in this area.

Browsers are generally affected by the vulnerability, but they are not responsible for the flaw. We found that some browsers are more vulnerable than others because of how they react to non-HTTP responses.

So how practical is the attack?

Most attacks require an active Man-in-the-Middle attacker, that means some way for an attacker to intercept and modify the data sent from the victim's browser to the web server. This is difficult on the Internet, but can be a plausible attacker model on the local network. Also, some attack variations do not require a Man-in-the-Browser, and thus are more dangerous. In particular, if you are still using Internet Explorer, we recommend you update to the latest version from June 8th, 2021.

Is my website/ftp-server/mail-server vulnerable?

It might be. If you are hosting several TLS-enabled application servers on the same hostname, or if you use multi-domain certificates, or if you use wild-card certificates, you may be vulnerable to the general confusion attack. If one of the application servers you are hosting has an exploitable upload, download, or reflection vector, this may negatively impact the security of your webserver.

Is my browser/client vulnerable?

Internet Explorer and Edge Legacy (i.e., those not based on Chrome) are "more" vulnerable than other browsers, because they block fewer ports and perform content-sniffing. Content-sniffing is dangerous, because it enables JavaScript code execution in server responses that are noisy due to error messages by the application server that implements a protocol different from HTTP. This means that the pure web attacker variant of ALPACA is more dangerous for users of such browsers than for other users.

However, no browser protects the user against all possible ALPACA attacks. In particular, all browsers can be compromised by a Man-in-the-Middle attacker who has write-access to an error-tolerant FTP server presenting a certificate compatible with a target web server under attack. Although the FTP server can in theory protect against this particular attack by detecting HTTP POST requests and/or terminating the connection after a small number of errors, this attack variant shows that this is not a bug in the browser, the web server, or the application server, but an emergent property of the TLS landscape.

Is this a new attack?

The ALPACA attack is not fundamentally new. Cross-protocol attacks on HTTP were first described by Jochen Topf (2001), and Jann Horn presented the first attack on a TLS-secured HTTP connection in 2014 involving ProFTPD. We did the first systematic study for cross-protocol attacks against the browser exploiting popular SMTP, IMAP, POP3, and FTP servers, performed an internet-wide scan to estimate the number of affected web servers, and generalized the attack away from a browser-specific issue to a general property of misconfigured TLS servers. We think that this new perspective is useful in focussing countermeasures on a limited number of effective options, rather than patching application servers one at a time as more exploits are found.

Why does TLS not protect the TCP connection endpoints?

The ALPACA attack is only possible because TLS does not protect the source or destination IP and port address of the TCP connection. As is stated in the TLS RFC, TLS is application layer independent. However, this gap in protection gives the attacker the flexibility to redirect traffic from one server to another. If the presented certificate of the substitute server is compatible with that of the intended server, the general content confusion attack is possible (although it depends on the server and client behavior if it can actually be exploited).

Can TLS mitigate these attacks at all?

Two extensions in TLS can provide some protection to the application layer protocol: SNI and ALPN. With SNI, the client can let the server know about the hostname it wants to connect to, which is useful in virtual hosting configurations. Sadly, SNI is often misconfigured with an insecure fallback to a default server, allowing content confusion attacks (for HTTP, these were analyzed by Delignat et al. in 2015, and Zhang et al. in 2020). However, the SNI standard allows the server to terminate the connection if the hostname does not match the expected hostname of the server, which would prevent some ALPACA attacks in practice. Unfortunately, this strict behavior is rarely implemented, even among web servers.

For application servers, which commonly lag behind web servers in feature completeness with regards to TLS, the situation is even more dire. With ALPN, the client can let the server know about the intended protocol, which is used to demultiplex between HTTP/1.x and HTTP/2 connections to a web server without requiring an additional roundtrip. Here the standard mandates strict behavior, so a server supporting ALPN should terminate the connection if no supported protocol is requested by the client. Unfortunately, this strict behavior is commonly not implemented, and many application servers do not even support the ALPN extension at all.

Why is the attack called "ALPACA"?

We initially were interested in special properties of the HTTP, FTP and email protocols that make cross-protocols practical. However, we eventually realized that the ALPACA attack is generic, and that the authenticity of the TLS connection is already compromised before any application layer data is exchanged. So, the original acronym ("Application Layer Protocols Allowing Cross-Protocol Attacks") was not a good fit anymore, because it is not the ALP allowing the attack, but the insufficiency of TLS to protect the TCP connection endpoints. Still, the name stuck, and we managed to squeeze the letters in the title in the following way: "Application Layer Protocol

Confusion - Analyzing and mitigating Cracks in tls Authentication". Tortured, we know. But ALPACAs are still cute. :)

How have vendors responded to this vulnerability?

Many vendors have updated their application servers to remove exploitation vectors or add countermeasures in the application layer and/or TLS implementation. TLS library maintainers have reviewed the ALPN and SNI implementations and updated their code and documentation to allow easy implementation of countermeasures by developers. To prevent the attacks in the pure browser attacker model, browser vendors have blocked more standard application ports and disabled content-sniffing in more scenarios.

Specific responses are listed below (please contact us if you have more info!):

- Microsoft Internet Explorer blocked more non-HTTP server ports and disabled content sniffing for HTTP requests to non-standard ports (CVE-2021-31971).
- Sendmail fixed a bug to detect HTTP requests when STARTTLS is used, and since Sendmail 8.17 there are additional countermeasures at the application layer to block HTTP requests.
- Courier 5.1.0 implemented support for ALPN.
- FileZilla implemented countermeasures at the application and TLS layer.
- Vsftpd 3.0.4 implemented countermeasures at the application and TLS layer.
- Nginx 1.21.0 implemented mitigations at the application layer in the mail proxy.
- crypto/tls (Go) now enforces ALPN overlap when negotiated on both sides.

How is this attack related to other TLS attacks?

ALPACA uses the same attacker scenario as other TLS attacks, i.e. it assumes a Man-in-the-Middle attacker who can lure a victim to an attacker-controlled web site. However, in the ALPACA attack, we do not try to attack the cryptographic protections of TLS directly. Instead, we exploit defects in the configuration of TLS services, who often share certificates to save costs, reduce administrative work, or enable reverse proxy deployments where several services share a single, terminating TCP endpoint. In contrast to other TLS attacks, the attacker never compromises the confidentiality of the TLS connection. However, due to misconfiguration, authenticity and integrity are affected, allowing the attacker to inject some dangerous data into the connection, while the victim remains oblivious to the attack.

What about other protocols?

The ALPACA attack is generic, i.e. it describes the preconditions under which TLS traffic from the client to one server implementing one protocol can be redirected to another server implementing a different protocol, which can lead to any number of undesirable behavior or security vulnerabilities. We only looked at the combination of a HTTP client with an SMTP, IMAP, POP3, or FTP application server. We did not investigate any of the other hundreds of possible cross-protocol scenarios possible with current TLS enabled applications and servers. In addition, as TLS is more widely deployed, more protocols will be added to the TLS landscape, increasing the possibility of ALPACA attacks quadratically with the number of protocols and applications.

If you find application layer protocol confusion attacks in other protocols, let us know! We are of course very interested in hearing about other affected protocols and applications.

If my clients and servers verify the ALPN and SNI parameters of the TLS handshake, will I be secure against this attack?

We do not think that it is feasible for clients or servers to enforce the use of ALPN and SNI for a long time, because doing so will exclude legacy clients and servers that have not been updated yet, and it is unlikely that this will be accepted by users or service providers. However, if both client and server support ALPN, and make sure that an acceptable protocol and hostname is negotiated, they will protect connections to all other servers with compatible certificates by the same client from almost all content confusion attacks.

However, there still is some room for content confusion attacks even with ALPN and SNI fully deployed. If two services implement the same protocol on the same host, but on a different port, connections to one server can be redirected to another by an attacker. This can enable same-protocol, same-host context confusion attacks similar to those described by Delignat et al. (2015) and Zhang et al. (2020) in very specific scenarios.

Is this vulnerability really serious enough to deserve a name, a logo and a web page?

ALPACA is not a simple software bug that can be fixed with an update to a single library or component. Instead, clients and servers need to be updated to protect the connections to other (seemingly unrelated) servers. This means we need to raise awareness of the issue across all TLS-enabled applications and protocols, which is a huge effort. We expect that the general ALPACA attack will stay with us for many years, so we have a cute animal to keep us company while we help clients and servers to adopt the suggested countermeasures!

How can I contact you?

You can reach us via mail or twitter:

- Marcus Brinkmann, Ruhr University Bochum, @lambdafu, [](https://alpaca-attack.com/cdn-cgi/l/email-protection)
- Christian Dresen, Münster University of Applied Sciences, @dr4ys3n, [](https://alpaca-attack.com/cdn-cgi/l/email-protection)
- Robert Merget, Ruhr University Bochum, @ic0nz1, [](https://alpaca-attack.com/cdn-cgi/l/email-protection)
- Damian Poddebniak, Münster University of Applied Sciences, @dues, poddebniak@fh-muenster
- Jens Müller, Ruhr University Bochum, @jensvoid, [](https://alpaca-attack.com/cdn-cgi/l/email-protection)
- Juraj Somorovsky, Paderborn University, @jurajsomorovsky, [](https://alpaca-attack.com/cdn-cgi/l/email-protection)
- Jörg Schwenk, Ruhr University Bochum, @JoergSchwenk, [](https://alpaca-attack.com/cdn-cgi/l/email-protection)

- Sebastian Schinzel, Münster University of Applied Sciences, @seecurity, schinzel@fh-muenster

Responsible Disclosure Timeline

- 2020-10-20: Initial contact with Eric Rescorla (author of TLS standard, CTO of Mozilla)
- 2020-12-03: Initial contact with OpenSSL.
- 2021-02-02: Initial contact with other TLS library maintainers.
- 2021-02-20: Initial contact with all affected application servers (FTP, Email).
- 2021-03-25: Initial contact with nginx and Apache.
- 2021-06-09: Public disclosure.
