Avoid Dangers of Wildcard TLS Certificates and the ALPACA Technique

Executive summary

Wildcard certificates are often used to authenticate multiple servers, saving organizations time and money. Wildcard certificates have legitimate uses, but can confer risk from poorly secured servers to other servers in the same certificate's scope.

A new style of web application exploitation, dubbed "ALPACA," increases the risk from using broadly scoped wildcard certificates to verify server identities during the Transport Layer Security (TLS) handshake. Application Layer Protocols Allowing Cross-Protocol Attack (ALPACA) is a technique used to exploit hardened web applications through non-HTTP (Hypertext Transfer Protocol) services secured using the same or a similar TLS certificate. This Cybersecurity Information Sheet details the risks from wildcard certificates and ALPACA, and provides mitigations for both.

Administrators should assess their environments to ensure that their certificate usage, especially the use of wildcard certificates, does not create unmitigated risks, and in particular, that their organizations' web servers are not vulnerable to ALPACA techniques.

Background

Web servers use digital certificates to securely identify themselves to web browsers. They use the certificates to establish a trusted, secure connection within which sensitive information can be shared. Web browsers often indicate to the user that they have properly verified the certificate and established the trusted, secure connection by displaying a lock symbol near the address bar. Once the secure connection has been established, sensitive information that has been stored on the client (e.g., secure cookies) is accessible to code from the server to run on the client. The code is limited to browser data based on the server's identity, which was verified by the digital certificate.

Organizations with multiple public-facing servers often use a “wildcard” certificate[1] to verify server identities during the Transport Layer Security (TLS) handshake when establishing a secure, trusted connection. Wildcard certificates can be used to represent any server with a similar name, where the wildcard—indicated by an asterisk—allows the certificate to represent any subdomain that falls under a base domain name. For example, a wildcard certificate for *.example.com can represent both www.example.com and mail.example.com. Wildcard certificates are typically used to authenticate multiple servers to simplify management of an organization’s credentials, often saving time and money. Common uses include by a proxy representing multiple servers. However, using wildcard certificates to validate unrelated servers across the organization introduces risk.

The well-known risks from using wildcard certificates are based on the compromise of any single server that uses the certificate or a downgrade exploit of a connection to any single server, putting all other servers that can be represented by that certificate at risk. A malicious cyber actor who gains control of the private key associated with a wildcard certificate will provide them the ability to impersonate any of the sites represented, and gain access to valid user credentials and protected information.

In addition to the well-known risks, cybersecurity researchers have recently shown that using wildcard or other certificates that represent both HTTPS and non-HTTPS servers can lead to exploitable web vulnerabilities that don’t depend on TLS weaknesses or private key compromises. Under certain conditions, this technique, dubbed Application Layer Protocols Allowing Cross-Protocol Attacks (ALPACA) [2], allows malicious actors to exploit fault tolerance features of web browsers and servers combined with protocol confusion between HTTP and other text-based protocols protected by TLS to perform arbitrary actions and view sensitive data. While the conditions permitting this complicated technique to succeed are uncommon, ongoing research in this area is likely to identify additional configurations vulnerable to this type of malicious activity.

Administrators should assess their environment to ensure that their certificate usage, especially the use of wildcard certificates, does not create unmitigated risks, and in

[1] A wildcard certificate is a single certificate with a wildcard character in the X509 certificate subject field (ex. CN = *.example.com) or subject alternative name field allowing the certificate to validate the identity of multiple subdomains of the same root domain.

particular, that their organizations' web servers are not vulnerable to ALPACA techniques.

Typical risks from using wildcard certificates

While wildcard certificates have legitimate uses in a trust architecture, there are risks to assigning these certificates to devices representing different applications. Because the wildcard certificate can be used to represent any system within its scope, including masquerading as ones that were not intended, protection of the private key is paramount. Storing a wildcard certificate private key on an application server introduces risk to other application servers within the certificate's scope.

For example, consider the following diagram where a wildcard certificate is scoped at the second-level domain of *.example.com and is stored on a poorly maintained web server.

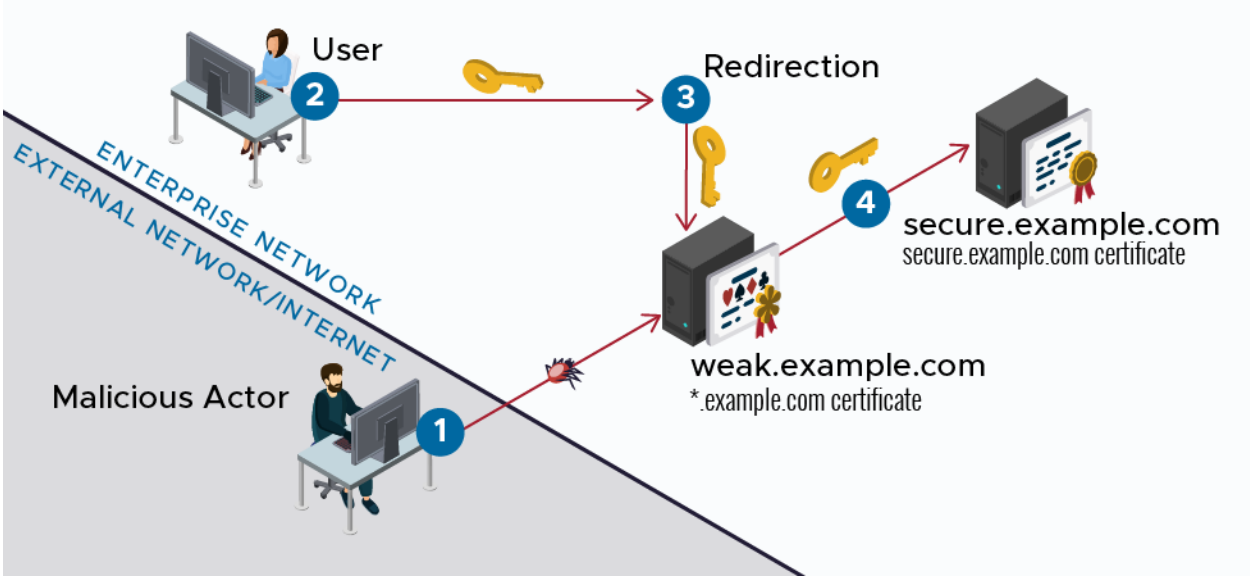

Figure 1: Compromise of an unsecured server leading to credential theft against a secured server in the same certificate scope

1. *A malicious actor compromises the poorly maintained web server (weak.example.com).*
2. *An internal user sends a request to an internal enterprise application (secure.example.com). The request contains sensitive information, such as login credentials.*
3. *Using one of many network manipulation techniques, the actor redirects the request to the weak.example.com server. Because the server presents a valid certificate, the TLS handshake completes and the sensitive data is passed to the server.*
4. *The malicious actor leverages the compromised credentials to authenticate to the secure internal application and access sensitive information.*

This example illustrates how applications within the scope of a wildcard certificate inherit risk from unrelated applications/devices that store the certificate.

Similarly, servers identified by a wildcard certificate using weak/obsolete cryptography confer risk to other servers within the certificate's scope. A well-positioned malicious actor can redirect traffic intended for a secure server to a vulnerable server. If the certificate is valid, then the TLS handshake will succeed allowing sensitive data to be sent by the client. The actor could then potentially exploit cryptographic weakness to obtain the underlying sensitive plaintext sent by the client.

The previous examples differ from two preferred wildcard certificate use cases. First, wildcard certificates can responsibly be installed on a proxy that routes traffic for multiple applications. In this case, weaknesses in the proxy device are presumed to impose risk for proxied applications, so no additional risk is implied. Administrators should limit the scope of wildcard certificates to the proxied applications so as not to increase the risk for other applications within the enterprise.

Second, wildcard certificates can be used responsibly to identify multiple devices that are all dedicated to the same application. In this case, if a compromise occurs, the use of a wildcard would not significantly increase the risk compared to the use of individual certificates representing application components.

Risks specific to ALPACA

ALPACA is a complex class of exploitation techniques that can take many forms. Administrators are encouraged to read the full ALPACA whitepaper [2] for additional details, but the most realistic exploitation scenario requires:

- a target web application that uses TLS,
- another service/application (typically not a web server) that presents a valid TLS certificate with a subject name that would be valid for the targeted web app, such as when wildcard certificates are too broadly scoped,
- a means for the malicious actor to redirect victim network traffic intended for the target web app to the second service (likely achieved through Domain Name System (DNS) poisoning or a man-in-the-middle compromise), and
- an HTTP request that is accepted by the second service that results in at least part of the request being reflected back to the sender.

Under these relatively uncommon conditions, a malicious actor can leverage phishing, watering hole, malvertising, man-in-the-middle techniques, or other means to cause a victim's browser to navigate to the second service. The victim's browser will incorrectly validate that service as the target web application since the service's certificate can represent the target application. From here, the actor will reflect a malicious script using the second service back to the victim's web browser, triggering a cross-site scripting (XSS) vulnerability and executing within the victim's browser in the context of the target web server. In other words, the actor's script running in the victim's browser could expose information to the actor, allowing the actor to impersonate the victim to the target web server. In some circumstances the actor's script running in the victim's browser could be redirected to access the real targeted web server, accessing arbitrary data, and performing actions using the victim's already authenticated session.

The following diagram illustrates the simplest form of this ALPACA exploitation:

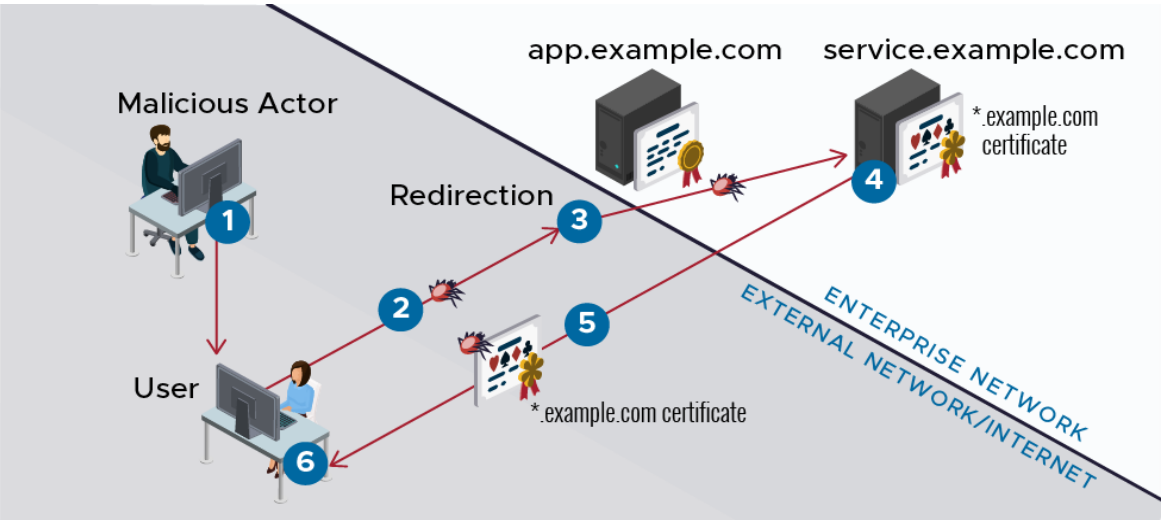

Figure 2: Compromise of a secured application through ALPACA exploitation

1. *The malicious actor induces the user to visit a crafted URL (phishing, malvertising, etc)*
2. *The user sends a request for the URL to app.example.com*
3. *Using one of many network manipulation techniques, the user's request is redirected by the malicious actor to service.example.com instead*
4. *The non-HTTP service.example.com (e.g., a File Transfer Protocol [FTP], Simple Mail Transfer Protocol [SMTP], or other non-web server) attempts to process the HTTP request causing an error that reflects the malicious content into the server's response*
5. *The server's response is signed by the *.example.com certificate*
6. *The user's browser receives the response to their request. Since the request was to app.example.com and the response is authenticated by *.example.com, the browser trusts the response and executes it within the context of app.example.com. This gives the malicious script access to user data and cookies for app.example.com within the browser.*

Mitigating poorly implemented certificates and ALPACA

Ensure responsible use and scope of wildcard certificates:

- Understand the scope of each wildcard certificate used for the organization. Identify all locations where the wildcard certificate's private key is stored and ensure that the security posture for that location is commensurate with the requirements for *all* applications within the certificate's scope.
    - Where possible, rather than having a single wildcard certificate authenticate unrelated applications, restrict the scope of wildcard certificates to servers hosting the same application (e.g., use a unique certificate for each web application and service supporting TLS). For example, the same non-wildcard certificate can be used for load balancers, hosting services, and proxies that are all representing the same application.
- Use an application gateway or Web Application Firewall (WAF) in front of servers, including non-HTTP servers.
    - Application gateways and WAFs often include functionality to filter traffic based on the TLS Server Name Indication (SNI) extension thereby preventing traffic misdirection. Administrators choosing to enforce this mitigation should ensure that SNI filtering is enabled and that each server using the wildcard certificate is behind an application gateway or WAF. WAFs implemented within the server enclave (after TLS is terminated) can provide an additional layer of security by ensuring that transactions to the application server match well-defined criteria and trigger alerts or block traffic when anomalies are detected.
        - This is why most web servers cannot be used as the second server for ALPACA, because most web servers check the SNI and HTTP Host header that the web browser sends in its request and reject requests intended for other servers. In addition, XSS is a well-known web exploitation technique and it is standard practice to harden websites to prevent reflecting scripts back to a client.
- Use encrypted DNS and validate DNS Security Extensions (DNSSEC) to prevent DNS redirection.
    - DNS resolvers should validate DNSSEC to authenticate DNS information from DNS servers, and then use DNS over TLS/HTTPS to provide "last

mile" integrity from the organization's DNS resolver to end clients. Administrators can refer to NSA's recent *Adopting Encrypted DNS in Enterprise Environments* guidance [1] for implementation details.

- Where possible, enable Application-Layer Protocol Negotiation (ALPN).
  - ALPN is a TLS extension that allows the server/application to specify permitted protocols (e.g., HTTP, Internet Mail Access Protocol [IMAP], Post Office Protocol [POP3], and FTP). Requests using protocols not specifically permitted will be dropped before being processed by the application. ALPN was standardized in 2014 for TLS 1.2, but support for ALPN varies across platforms and packages. Administrators should confirm that their environments support ALPN prior to broadly implementing it and should update legacy software to adopt the capability.
- Maintain web browsers at the latest version with current updates
  - To counter the impact of ALPACA, some web browser have started blocking traffic to ports most frequently vulnerable to being used for this type of exploitation, such as TCP ports 465, 587, 989, 990, and several others. While this mitigation is effective for specific high-risk applications, it does not counter the ALPACA threat broadly. Administrators should ensure that organizations are postured to receive automatic browser updates to receive these and other timely mitigations.

Defense-in-Depth

By avoiding or responsibly using wildcard certificates, organizations can harden network identities against malicious actors using masquerade techniques. Additionally, ALPACA mitigations block known protocol confusion exploits and strengthen network posture against potential future issues. Administrators should always seek to apply defense-in-depth approaches that apply to classes of risks/threats in order to counter malicious threat actors.▪

Works cited

[1] National Security Agency, "Adopting Encrypted DNS in Enterprise Environments," 2021. Available: https://media.defense.gov/2021/Jan/14/2002564889/-1/-1/0/CSI_ADOPTING_ENCRYPTED_DNS_U_OO_102904_21.PDF.

[2] M. Brinkmann, C. Dresen, R. Merget, D. Poddebniak, J. Muller, J. Somorovsky, J. Schwenk and S. Schinzel, "ALPACA: Application Layer Protocol Confusion – Analyzing and Mitigating Cracks in TLS Authentication," 2021. Available: https://alpaca-attack.com/ALPACA.pdf.

Disclaimer of endorsement

Purpose

This document was developed in furtherance of NSA's cybersecurity missions, including its responsibilities to identify and disseminate threats to National Security Systems, Department of Defense, and Defense Industrial Base information systems, and to develop and issue cybersecurity specifications and mitigations.

Contact

Client Requirements / General Inquiries: Cybersecurity Requirements Center, 410-854-4200, Cybersecurity_Requests@nsa.gov
Media Inquiries / Press Desk: Media Relations, 443-634-0721, MediaRelations@nsa.gov