FDD | CCTI Center on Cyber and Technology Innovation

Research memo

Is the Padlock on Your Browser Bar Giving You a False Sense of Security?

How Trust is Managed (and Mismanaged) on the Internet

David Wu, Ph.D., Douglas Wood, Ph.D., and Michael Hsieh, Ph.D.
Transformative Cyber Innovation Lab

July 6, 2020

How Do You Know Whom You Are Really Talking to on the Internet?

When users connect to a website such as Gmail, how do they know that they are actually talking to Gmail and not, say, a malicious website pretending to be Gmail? Today, trust on the internet is managed through a system called the public-key infrastructure (PKI), which associates real-world identities with digital entities. Within PKI, trusted organizations called certificate authorities (CAs) play a central role in connecting real-world identities with digital assets, such as websites, devices, or systems, by vouching for the identity of websites on behalf of their owners.

When a user initiates a connection to a website, the website responds with a certificate from the CA that identifies the owner of the website to the user. Based on the certified identity of the website owner, the user can decide whether to continue with the connection or not. However, this is far from a foolproof system. While the technological basis of PKI is sound, the soft underbelly of PKI has primarily been its human layer – specifically, badly-designed security rules and policies that present openings for attackers to circumvent the otherwise sound security properties of PKI. This research memo will explore how this system has been successfully compromised in the past, and will propose recommended practices to prevent or defend against these attacks.

Why Does It Matter?

CAs serve as a root of trust on the internet. If attackers can coerce or trick a CA into issuing them a certificate identifying them as the owners of a legitimate service (such as Gmail), then the attackers can both impersonatethe legitimate service as well as intercept and read all communication between unsuspecting users and that service. In this setting, users believe they are talking to the actual service, when in reality they are talking to the attacker.

In this type of "man-in-the-middle" (MITM) attack, users may unwittingly expose all of their communications (such as login credentials, sensitive data, et cetera) to the attacker. Importantly, by compromising the CA, the attacker is able to defeat all of the cryptographic protections used to ensure the confidentiality and integrity of communications on the internet.

David Wu is an assistant professor of computer science at the University of Virginia. Douglas Wood is chief scientist of Ursa Space Systems and a board adviser for FDD's Center on Cyber and Technology Innovation (CCTI). He previously led extremely large distributed computing system architecture at the Defense Information Systems Agency. Michael Hsieh is executive director of TCIL, a project of CCTI.

This is not just a hypothetical problem. In fact, we have recently seen multiple high-profile cases:

- Internet service providers in Kazakhstan required customers to add a new government CA to their trusted list of CAs.[1] This enabled the Kazakh government to eavesdrop on user communications with anysite.
- The French Treasury issued a fake certificate for Google[2] and used it to eavesdrop on employees' Gmail conversations.
- The Turkish government issued a fraudulent certificate for all Google domains[3] to enable it to eavesdrop on user communications on Google platforms.
- Iranian state actors were able to obtain certificates for domains owned by major technology companies (such as Google, Yahoo, and Microsoft) from Comodo, a legitimate and trusted CA.[4] This allowed them to impersonate those services and eavesdrop on all communications between users and those platforms.

In each of these cases, the attacker was able to obtain a certificate for a major web service from a CA that was trusted by users' browsers. In the first three examples, government agencies were themselves a trustedCA and thus had the ability to issue certificates for other domains. In the last case, malicious actors compromised internal systems of a third-party CA to obtain fraudulent certificates. As a result of these compromises, several CAs (such as DigiNotar[5] and TurkTrust[6]) are no longer trusted by most major browsers.

How Is Trust Managed on the Internet Today?

Suppose Ted owns the CNN news website. As the website owner, Ted would like to assure his visitors that they are indeed viewing hiswebsite and not a fraudulent website set up by a malicious third party. To do so, Ted would register the domain of his website (cnn.com, for example) with a CA. After the CA confirms that Ted is indeed the owner of cnn.com and has rights to the CNN website, the CA issues Ted a digital certificate asserting that the domain cnn.com belongs to Ted. The certificate also includes a "digital signature" from the CA.

A digital signature scheme is a cryptographic mechanism for guaranteeing the authenticity and integrity of digital messages.[7] In some sense, it functions as a digital analog of a physical signature. For instance, when Ted sends a message to Alice, another user, Ted can include his signature on the message. When Alice receives the message together with the signature, she can check for Ted's signature on the message. If Alice sees a valid signature from Ted, then she is convinced that Ted sent the message (a property often called "authenticity") and, moreover, that the message from Ted was not tampered with in transit (a property often called "integrity").

1. "Kazakhstan's new online safety tool raises eyebrows," (UK), July 22, 2019. (https://www.bbc.com/news/technology-49071222)
2. John Leyden, "French gov used fake Google certificate to read its workers' traffic," (UK), December 10, 2013. (https://www.theregister.co.uk/2013/12/10/french_gov_dodgy_ssl_cert_reprimand/)
3. Sean Gallagher, "Turkish government agency spoofed Google certificate 'accidentally,'" , January 4, 2013. (https://arstechnica.com/information-technology/2013/01/turkish-government-agency-spoofed-google-certificate-accidentally/)
4. "Iran accused in 'dire' net security attack," (UK), March 24, 2011. (https://www.bbc.com/news/technology-12847072)
5. Kim Zetter, "DigiNotar Files for Bankruptcy in Wake of Devastating Hack," Wired, September 20, 2011. (https://www.wired.com/2011/09/diginotar-bankruptcy/)
6. "Revoking Trust in Two TurkTrust Certificates," Mozilla, January 3, 2013. (https://blog.mozilla.org/security/2013/01/03/revoking-trust-in-two-turktrust-certficates/)
7. For a heuristic explanation for how digital signature schemes work, see the Appendix.

What Are the Limits of Current Approaches to Managing Trust?

Ultimately, secure communication over the internet is predicated on CAs performing their due diligence when issuing certificates to domain owners (as well as on timely verification and processing of certificate revocation requests). Notably, the tasks of issuing and revoking certificates are both entirely non-technical and rely on a human-based verification process. Yet at the same time, both of these components are fundamental to securing communication on the internet. Thus, a key limitation of the existing system is that it is often easier to fool humans than to fool computers.

This issue is further compounded by the fact that there are over 100 trusted CAs in popular operating systems and browsers,[8] and a compromise in any single one of them compromises all trust on the internet. Essentially, users' trust on the internet requires them to simultaneously trust all of these different CAs. If an eavesdropper Eve is able to convince even a singletrusted CA to issue a fraudulent certificate for a domain such as cnn.com, she can employ an MITM attack to eavesdrop on every user's communication with CNN, while remaining essentially undetectable to the unsuspecting user or to the Internet service provider.[9]

To illustrate how an adversary might mount an MITM attack, consider again the previous example of Alice and CNN. Suppose Eve has a certificate from a trusted CA that binds the domain cnn.com with hersignature verification key. To mount an MITM attack against Alice and CNN, Eve first intercepts Alice's initial request to CNN. Eve then sets up her own connection with CNN (pretending to be Alice). At the same time, Eve completes the connection setup with Alice using her own certificate for CNN. From Alice's perspective, she has received a valid certificate for CNN, and now believes that she is talking to CNN.[10]

Importantly, Alice's browser would notsuggest that anything suspicious has occurred. However, in reality, both Alice and CNN are communicating with Eve. With this setup, Eve can passively eavesdrop on the communication by relaying messages between Alice and CNN. And if she so chooses, Eve can also maliciously change or inject new messages of her choosing on behalf of either Alice or CNN.

How Can Trust on the Internet Be Improved?

There are several simple approaches for mitigating some of the problems with the existing public key infrastructure. Many of these approaches restrict the set of entities that are trusted, and these approaches can be implemented at the system-administrator level.

Proposal Rule 1: Certificate Pinning. Certificate pinning is a way for the client to verify the server's certificate by comparing it to a local copy of the certificate. The authors refer to the locally stored copy of the certificate as a "pinned" certificate. In this case, there needs to be some (possibly separately arranged) setup between the client

8. For instance, there are roughly 150 default trusted CAs in Mozilla Firefox, over 170 in macOS, and over 100 in Android. "Included CA Certificate List," , June 29, 2020. (https://ccadb-public.secure.force.com/mozilla/IncludedCACertificateReport); "List of available trusted root certificates in iOS 12, macOS 10.14, watchOS 5, and tvOS 12," ,accessed June 29, 2020. (https://support.apple.com/en-us/HT209144); "Security with HTTPS and SSL," ,accessed June 29, 2020. (https://developer.android.com/training/articles/security-ssl)

9. While MITM attacks are often nefarious in nature, there are some legitimateapplications in enterprise environments. For instance, an MITM might be deployed on a corporate network to intercept and decrypt incoming web traffic for virus and malware scanning. These types of applications are sometimes referred to as "blue coat" technologies. (See, for example: Blue Coat Systems, a security company that has developed MITM tools for enterprise scenarios.) Of course, this same type of technology can be used to violate user privacy and eavesdrop on sensitive communications, and thus, any such use should be carefully vetted and controlled.

10. For a detailed explanation on the mechanics of the cryptography of certificate signing, see the Appendix.

and the server whereby the client obtains a copy of the server's certificate. For example, Google is able to embed the certificates for Google services within the Chrome browser. Then, when a client visits a domain for which it possesses a pinned certificate, the client compares the certificate it receives from the server against the version the client already has, and proceeds only if the two certificates match (that is, if they identify the same domain and signature verification key). This way, the client no longer relies solely on an external CA to vouch for the identity of the server. If an MITM adversary were to obtain a certificate for a domain with a pinned certificate, the client would no longer be fooled, since the adversary's certificate would not match the pinned version.[11]

For sensitive and mission-critical systems, certificates for authorized peers should be pinned to the client and not authenticated by external third-party CAs. Moreover, for systems intended to be isolated from the external internet, the onlypermissible connections should be with servers identifiable by a pinned certificate or by certificates issued by internal CAs. There should be no reliance on external (and possibly unreliable) roots of trust in these scenarios.

Proposed Rule 2: Banning Foreign CAs in Sensitive Networks. On sensitive and critical networks such as those employed by parts of the U.S. federal government, the set of trusted CAs should be carefully curated. A possible starting point is to exclude all foreign CAs from the set of trusted CAs. If for some reason it is infeasible to exclude all foreign CAs, the default behavior should be to reject certificates issued by such CAs and allow exceptions on a case-by-case basis.

Ideally, on sensitive networks, the network administrator would systematically review each potential CA (foreign or not) and decide whether the CA should be included in the set of trusted CAs. In doing so, the security of the network relies on a much smaller number of roots of trust. Of course, if certificate pinning is an available option, then the systems should be configured to accept connections only from an trusted set of peers and not to rely on external CAs at all as a root of trust.

The recently released final report by the congressionally mandated Cyber Solarium Commission urged Congress to "direct the U.S. government to develop and implement an information and communications technology industrial base strategy to ensure more trusted supply chains and the availability of critical information and communications technologies."[12] While not specifically recommended in the Commission's report, banning foreign CAs in sensitive networks is a necessary step to reshape the cyber ecosystem toward greater security.

Proposed Rule 3: Banning the Hosting of Commercial Sites on Government Enclaves. The general practice within the U.S. federal government is to require government-issued certificates for government websites. This is to ensure compliance with proper security requirements on any equipment associated with a government website. However, in certain cases, it is technically feasible for a negligent or malicious individual (for instance, a contractor) to deploy a web server within a government network that is authenticated by commercial (non-governmental) certificates. Such systems are far more likely to be compromised or vulnerable than properly vetted systems, and as such should be strictly prohibited. All websites hosted on government infrastructure should be authenticated by government-issued, non-commercial certificates.

11. For example, pinned certificates (for Google services in Chrome) played an important role in revealing the compromise of the certificate authority DigiNotar. Josephine Wolff, "How a 2011 Hack You've Never Heard of Changed the Internet's Infrastructure," , December 21, 2016. (https://slate.com/technology/2016/12/how-the-2011-hack-of-diginotar-changed-the-internets-infrastructure.html)

12. Cyberspace Solarium Commission, "United States of America Cyberspace Solarium Commission Report," March 2020, page 88. (https://drive.google.com/file/d/1ryMCIL_dZ30QyjFqFkkf10MxIXJGT4yv/view)

Proposed Rule 4: Banning the Acceptance of Self-Signed Certificates. "Self-signed" certificates are inherently dangerous and should be banned outright.[13] As the name suggests, a self-signed certificate is one that is authenticated by the same entity the certificate identifies. As self-signed certificates are not issued by CAs, anyone is able to generate a self-signed certificate for any domain of his or her choosing. There is no binding between the entity identified by a self-signed certificate and the public signature verification key the certificate identifies. As such, self-signed certificates provide absolutely zero security assurances about the identity of the holder and should be treated accordingly.

Conclusion

The basis of trust on the internet rests in the hands of CAs that are responsible for authenticating the identities of the many services and domains on the web. However, this process of validating user identities and issuing certificates is an innately human-driven process and thus can be fooled, subverted, and compromised. The security of sensitive, mission-critical networks should ultimately notrely on the integrity of externalauthorities.

Appendix: How a Digital Signature Functions

A digital signature scheme consists of three algorithms: a key-generation algorithm, a signing algorithm, and a verification algorithm. Standard software implementations of these algorithms are typically shipped as part of the browser (to verify the identities of remote servers on the Internet) and the operating system (to verify the authenticity of software updates). The key-generation algorithm in a digital signature scheme is responsible for generating a pair of cryptographic keys: a verification key and a signing key. Both of these keys are, in practice, a sequence of binary values. The signing algorithm takes as inputs the signing key and a message, and it outputs a signature on the message. The verification algorithm takes the verification key, a message, and a signature, and it outputs either "accept," to indicate that the provided signature is valid for the message, or "reject," to indicate that the provided signature is invalid.

Typically, the signing key is kept secret so only its owner can use it to sign messages, while the verification key is made public so anyone can verify signatures. When Alice receives a message from Ted that contains a valid signature from Ted (that is, with respect to Ted's public verification key), then she knows that only someone who knows Ted's signing key could have produced that signature. As long as Ted's signing key has not been compromised, the only person who could have signed the message is Ted. Note, of course, that this requires Alice to know that a particular verification key indeed belongs to Ted and was not, for instance, generated by a malicious party pretending to be Ted. This is not a trivial problem, as a verification key is just a sequence of bits and contains no information about its owner. The primary role of certificates in solving this problem is to provide this critical link between the sequence of bits that represent a verification key and the concrete real-world identity associated with that key.

Now, before Ted registers his domain cnn.com with a CA, he first runs the key-generation algorithm for a digital signature scheme to obtain a verification key and a signing key. When Ted subsequently goes to the CA to register his domain, in addition to providing evidence of his identity, proof of ownership for the domain cnn.com, and the rights to host the CNN website, Ted additionally provides the verification key he generated earlier (recall that this is just a binary string). After validating the information Ted provided, the CA issues Ted a certificate that includes information on Ted's identity as well as his public verification key.

13. The authors are referring, of course, to self-signed certificates issued by an entity that is not a trusted root CA. In particular, the (self-signed) certificates of trusted root CAs are pinnedto the client's local trusted certificate store.

When a CA issues a certificate for a domain, the certificate includes a signature from the CA on the contents of the certificate. When a user's browser receives a certificate from a domain, the browser checks that the certificate contains a signature from a CA that it trusts. This sequence of steps of course assumes that the browser knows the public verification key of the CA that issued the certificate. Today, browser and operation system distributions typically ship with a default set of trusted CAs already in place. The public verification keys for each of these trusted CAs are bundled with the software distribution for the browser or the operating system. The user also has the flexibility to add or remove CAs to the default set of trusted CAs.

After Ted receives the certificate from the CA, he posts the certificate on his web server. In addition, Ted stores the signing key for the signature scheme on his web server. When a client such as Alice initiates a connection to cnn.com, Ted's web server responds with the certificate identifying cnn.com as well as some additional messages that will be used as part of a cryptographic protocol to secure the rest of the communication between Alice and the server. Ted's web server signs all of these additional messages using Ted's signing key.

When Alice connects to cnn.com, she obtains a copy of Ted's certificate, which allows her to verify that the domain cnn.com is owned by Ted and, moreover, that the certificate is issued by a CA that she trusts.[14] She also learns Ted's signature verification key, which she will use to verify that all of the subsequent messages that Ted sends contain a valid signature from Ted. Here, the digital signature establishes a link between the messages received by Alice and Ted's signature verification key, while the certificate establishes a link between Ted's verification key and his real-word identity.

After Alice (or her browser) has verified the server's certificate, then she is confident that she is indeed talking to Ted's CNN website. Moreover, she also knows Ted's public verification key. This means that whenever Alice receives a message from the server, she can check the signature to be assured that the message indeed came from Ted and was not tampered with in transit. (Recall that the digital signature scheme ensures both message authenticity and message integrity). With this guarantee, Alice's browser and Ted's server can rely on standard cryptographic techniques to ensure confidentiality and integrity for all of her subsequent communications with Ted's server. At this point, Alice is assured that she is communicating with Ted and, moreover, that all of her communications with the server are visible only to Ted. (Observe, however, that if Ted's secret signing key has been compromised, then anyone who knows Ted's signing key can impersonate Ted and eavesdrop or tamper with communications between Ted's web server and Alice.)

At a later point in time, if Ted discovers that some other certificate for cnn.com has been issued (binding to some other public verification key), he would have to convince the issuing CA that a mistake was made and have the CA revoke the certificate in question. Alternatively, he could sue the CA for improper issuance of a certificate, or he could seek to expose the CA for fraudulent activity and have major software vendors remove the rogue CA from the list of trusted CAs.

14. In the case of the actual CNN website, the certificate is issued to a company called Fastly, Inc., located in San Francisco. Fastly is a cloud service provider that operates websites on behalf of clients like CNN. The certificate authority vouching for CNN is GlobalSign, a Belgian corporation and major global CA. When Alice connects to cnn.com, her browser first checks that the certificate presented by the server is for the cnn.com domain. Then the browser checks that the certificate was issued by a CA that it trusts (that is, the certificate was issued by an authority in the browser's list of trusted CAs and, moreover, that the certificate has a valid digital signature from the corresponding CA). Finally, the browser checks that the CA has not previously revoked the certificate. However, this entire sequence of steps is predicated on the assumption that the CA (GlobalSign) properly validated the identity of Fastly and confirmed that Fastly is indeed the owner of the domain cnn.com. Thus, if Alice trusts the security and integrity of her connection to cnn.com, she is implicitly trusting that GlobalSign performed due diligence when issuing CNN's certificate.