Securing Browser Interactions

Lucas Adamski

Mozilla Corporation

Defense in Depth

Biography

Lucas Adamski heads up the software security team at Mozilla, works on security architecture and features, and tries hard to make the Internet a happier and safer place.

My name is Lucas Adamski, and I head up the Security Engineering team at Mozilla. We are focused on securing Mozilla's products and projects, including Firefox and many other projects such Weave and Firefox extensions. I have been working on software security for about 10 years now, at companies such as Macromedia, Adobe and @stake.

Overview

- Korea has some specific requirements for e-commerce and online banking:
- Communication over SEED cypher
- Firewall, anti-keylogging, and anti-malware controls
- Digital certificates for authentication and transaction signing

As you know, Korea has some requirements around online banking and e-commerce. Some of these are quite different from what is practiced in other countries. While other countries are beginning to use digital certificates for authenticating users, most still rely primarily on HTTPS for transport security.

Benefits

- Designed in 1998, the goals of this system were forward looking and commendable
- Digital signing systems can provide a meaningful additional layer of security if implemented carefully
- Separate transaction confirmation & signing in the control UI can mitigate many HTML code injection attacks

When the Korean government started designing this system, 128 bit SSL was not standardized, nor was it legal to export from the United States. In many respects this approach was very advanced for its time, considering the larger key size and the use of client certificates for login and transaction authentication. The use of a separate UI to sign a transaction can help mitigate many common HTML injection attacks.

Weakness

- Transport security and trust
- Control/plugin instantiation
- Content injection
- Over-reliance on part-time anti-malware
- Limitation of client certificates
- Uses trained to trust plugin installs

Unfortunately, the model as currently implemented also presents many weaknesses, which to a large degree cancel out the benefits. The user currently cannot determine the trustworthiness of the content that they interact with, largely because it cannot guarantee the confidentiality and integrity of content delivered from the server. Some of the mitigations provided are also of very little value if a computer has actually been compromised. I will go into some of these weaknesses in more detail.

Transport Security

- The HTML page is loaded over HTTP
- How to tell whether to trust the HTML content?
- Answer: if its loaded over HTTP, they cannot
- An attacker can inject their own content into the HTML, since its sent in cleartext
- Entire HTML UI is untrustworthy

Since the HTML content is actually loaded over cleartext HTTP, then it cannot be trustworthy, since a Man-In-The-Middle (MITM) attacker could inject their content into it before the users sees it. There is no way for the browser to tell if that has happened. Given the user interacts with the HTML content extensively even on banking sites, this is a big problem.

Content Injection

When an attacker injects code, they can:

- Prevent security controls from loading, load a fake one (i.e. built in Flash or Java), or trick the user into installing a different one (malware!)
- Manipulate the configuration of security controls, since they depend on the HTML for a lot of their settings
- Steal any data provided to the HTML, including account numbers, PIN codes, transaction history, etc.

Once an attacker can inject code into the HTML, they effectively control the user interface for that website. They can modify existing content, prevent new content from loading, or inject their own content instead. That includes ActiveX controls and plugins, since they are really just types of content as well.

Since they attacker can inject JavaScript, that JavaScript can monitor the webpage to steal any data that may be presented to the user. The attacker can also present their own UI to request PIN numbers, secret codes, or anything else the want to steal from the user.

In addition, since that JavaScript code has access to read or modify any content in that page, it can silently monitor all HTML forms for changes to data and send them back to the attacker. The firewall will not prevent this since, if the attacker can inject data or view the connection, they could just hide that data in a request back to the bank which the firewall would allow.

Instantiating Controls

- ActiveX controls and Netscape plugins are loaded by an HTML page
- Users trained to install controls by content that cannot be trusted in the first place
- Code signatures aren't a silver bullet: attacker could provide user with an older but validly signed control with known exploits (downgrade attack)

An inherent limitation for using plugins or ActiveX controls for security is that they aren't running until loaded by HTML content. So they need to be started by the very HTML content the user cannot trust. In addition, if they do not already have the required controls, they will be prompted to install them by that very same untrustworthy content.

Thus an attacker can prevent them from loading, or prompt the user to install their own control. Given it looks like the bank is asking the user to install it, they most likely will. Even though controls and plugins can be signed, they are often signed by a company other than the bank itself, so users are familiar with installing controls from publishers they may not recognize or that aren't related to the website itself.

Over-Reliance on Anti

- Anti-malware, anti-keylogging, and similar tools cannot stop malware already running on the system
- A compromised system cannot be made safe
- So it cannot protect private key or passphrase
- Korea is #1 source of malware by some studies, so rates of infection are high.*

* "Korea reigns as king of malware threats", http://www.infosecurity-us.com/view/8547/korea-reigns-as-king-of-malware-threats-/

Malware already knows how to disable antivirus, hide from them, or otherwise avoid detection. They can hook into the browser or OS processes, and prevent antivirus from running or patch it in memory. Part-time anti-malware and firewalls provides little value.

Once a system is compromised, no interaction with that system is trustworthy and no data on that system can be safe (that includes encrypted data if accessed by the user after infection). So you cannot protect a private key or passphrase if used on the system.

Malware is prevalent in Korea, or at least not less so than the rest of the world. As such, many systems must be assumed to be already infected with malware.

Benefits of HTTPS

- Server authentication before any data is exchanged
- Encryption of all data in transit
- Visible identity indicators
- HTTPS-only security features

Supporting HTTPS comes with many benefits. The server is authenticated to ensure the user is talking to the server they think are talking to, before any content is sent or received. All data in transit is encrypted. There are also some additional HTTPS-only security features that are available or currently being implemented.

Also, HTTPS is really a container for an extensible set of crypto algorithms, so a server can usually swap to a different one if a compromise is found.

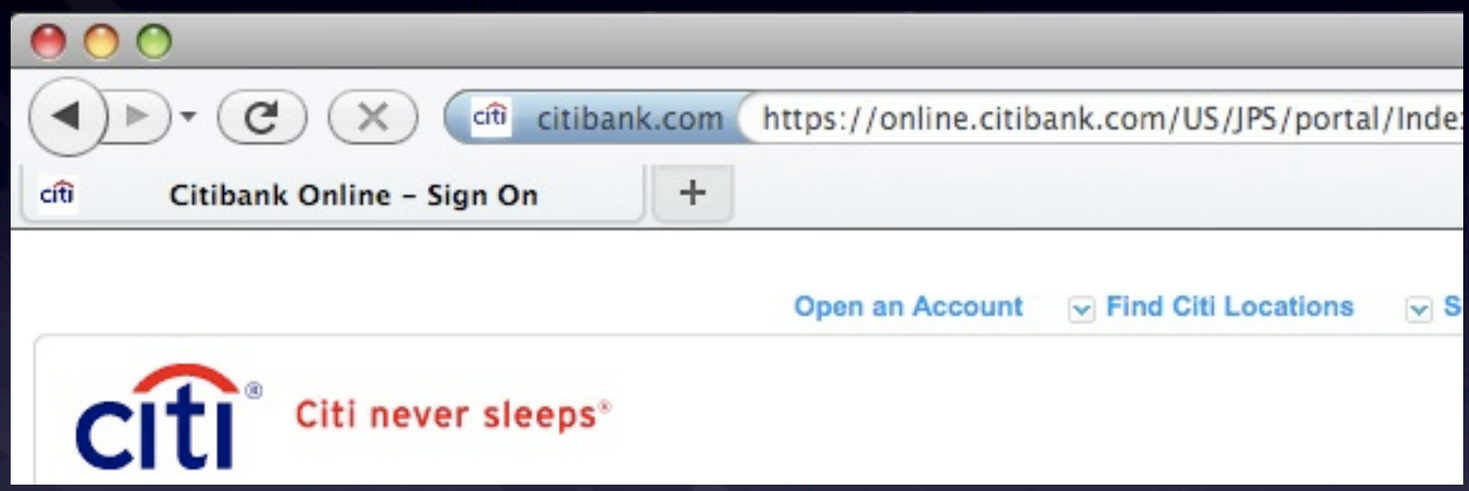
HTTPS Server Auth

- By default the browser will not connect to a server with an invalid or expired certificate
- Certificates can be revoked without code changes
- The identity of the server is clearly displayed to the user
- Extended Validation certificates provide additional information about server's identity

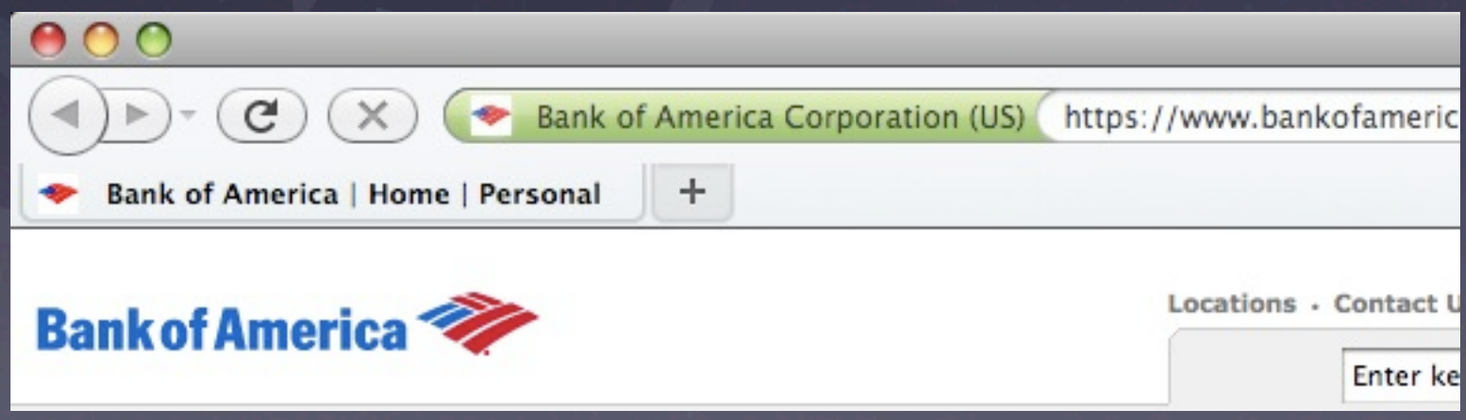
The browser will not normally send or receive any content from a website with an invalid or expired certificate, or if the certificate does not match the server name. This means that there is no opportunity for an MITM injection attack to happen in the first place. Most browsers also provide some additional information in the UI to help the user identify the site they are actually interacting with. Extended Validation certificates provide additional information to the user by displaying the name of the company to the user, rather than just the server name.

Site Authentication

DV Certificates



EV Certificates



Comparison of UI

Comparing the user experience of HTTP vs. HTTPS websites when under attack from a man-in-the-middle (MITM).

I would like to take a moment to compare the user experience of what happens when an HTTP website is attacked with a man-in-the-middle attack, compared to an HTTPS website.

HTTP Normal

This is an HTTP website without a man-in-the-middle attacker.

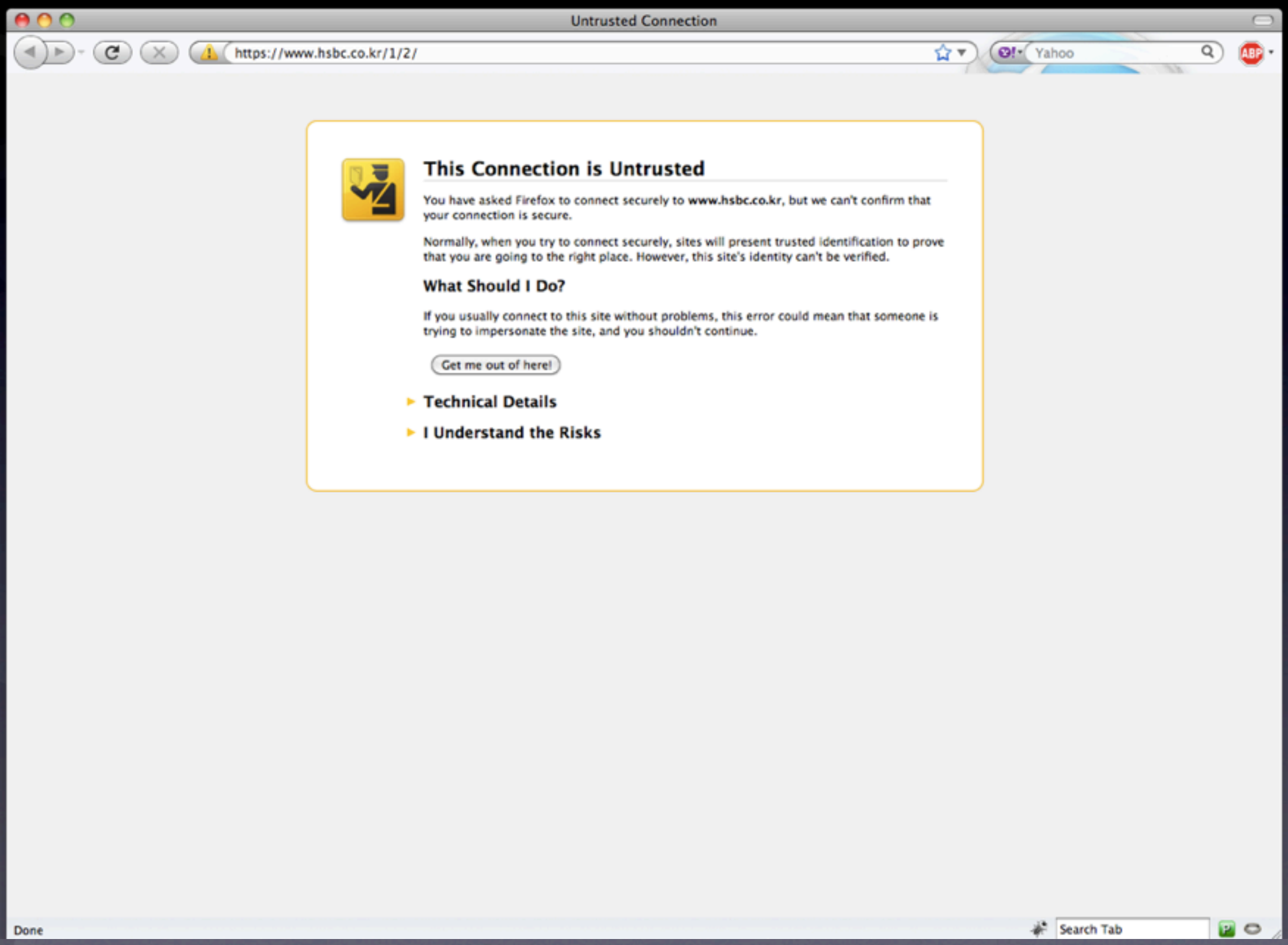
HTTP MITM

This is an HTTP website compromise by a man-in-the-middle attacker. I actually did user a web proxy tool to inspect and modify this page in transit, so it is simulating a real attack.

HTTPS Normal

This is an HTTPS website as it looks normally.

HTTPS MITM



This is an HTTPS website when intercepted by a man-in-the-middle attacker. I used exactly the same technique as I did in the HTTP scenario.

HTTPS-specific features

- Cookies with secure attribute set help guarantee that the browser will never send session ID over HTTP
- Strict Transport Security: Proposed mechanism to let websites allow communication only over HTTPS

When utilizing HTTPS, the server can specify a cookie with the “secure” attribute set, which is only permitted to be sent over HTTPS. So the session identifier will only be sent over an encrypted and authenticated channel.

In addition, a feature currently in development called “Strict Transport Security” allows a server to specify that the browser should only ever talk to it via HTTPS, mitigating the risk that the user could be tricked into try to connect to the same server over HTTP.

Weaknesses of HTTPS

- SSLstrip: prevents transition from HTTP to HTTPS. Not a recent attack; this was never considered safe in the first place.
- SSLstrip cannot intercept users already in HTTPS.
- Mitigated by Strict Transport Security, or simply bookmarking your bank's HTTPS login page! Visual indicators also help.

A researcher named Moxie Marlinspike gained a lot of attention for his SSLstrip tool. This is not a new attack however; it just utilizes the fact that its not currently possible to transition the user from an insecure session to a secure one by clicking a link in insecure content. I even blogged about this very issue back in 2007, so its one of my favorite topics. This very same problem also exists in the current Korean model.

It is important to understand that SSLstrip cannot downgrade a user who is already in an HTTPS session back to HTTP. So a user that bookmarks the bank's HTTPS login page is not at risk from SSLstrip. The Strict Transport Security (STS) feature currently being rolled out will also address this issue.

Browsing History

Most browser now remember the most popular link for a given site. So in cases where the bank redirects from HTTP to HTTPS, they will generally suggest the link to the encrypted home page for the bank.

Certificate Issuance

- Certificate authorities (CAs) have sometimes been sloppy in issuing certificates
- These have been caught and the certificates revoked
- Aggressive governance of the certificate authorities is the long term answer
- Extended Validation certificates help

There have certainly been issues with some certificate authorities who have not been diligent enough when issuing certificates. These have been caught and the issuing CA's have improved their practices, and the relevant certificates revoked. Extended Validation certificates also have much higher verification standards, and should be used by all banks and e-commerce sites.

User Costs of Korean Model

- False sense of security
- Have to use Microsoft OS and IE browser
- Not supported on mobile devices

The current Korean model has some unfortunate costs for the user. It provides the user with a false sense of security about their banking information. In addition, the user is usually limited to using ActiveX controls, which prevents the use of other browser and operating systems, and most mobile devices.

Recommendations

- Controls and plugins are no substitute for comprehensive authentication and encryption
- Signing does provide a meaningful benefit when implemented carefully - but that requires a separate device (out-of-band verification)
- Temporary technological counter-measures make people feel good but are easily defeated
- Crypto trustworthiness is proportional to degree of scrutiny of algorithm and implementation

The current model of untrusted content and (maybe) trustworthy controls is not effectively protecting users. Sites should implement HTTPS, preferably with extended validation certificates. This will provide the minimum measure of confidentiality and integrity necessary to secure users' banking and e-commerce sessions.

Client certificates still provide value, but the current implementation is flawed. They would be far more effective if the verification step happened out of band, on a separate device such as a mobile phone. The current requirements around part-time anti-malware and firewall do very little to actually protect the passphrase and private key. This a problem of non-repudiation: users could easily be blamed on transactions performed by malware that has compromised their system.

Mozilla would be happy to engage in further dialog regarding a solution that could incorporate the benefits of these different models, in a way that allows the users to have a solid foundation of security while maximizing their ability to choose their own browser, operating system and device. I thank you for your time.

Questions?