Concepts against Man-in-the-Browser Attacks

Philipp Gühring <pg@futureware.at>

Date: 2006-06-16

Update: 2006-09-12

A new threat is emerging that attacks browsers by means of trojan horses. The new breed of new trojan horses can modify the transactions on-the-fly, as they are formed in in browsers, and still display the user's intended transaction to her. Structurally they are a man-in-the-middle attack between the the user and the security mechanisms of the browser. Distinct from Phishing attacks which rely upon similar but fraudulent websites, these new attacks cannot be detected by the user at all, as they are using real services, the user is correctly logged-in as normal, and there is no difference to be seen.

The WYSIWYG concept of the browser is successfully broken. No advanced authentication method (PIN, TAN, iTAN, Client certificates, Secure-ID, SmartCards, Class3 Readers, OTP, ...) can defend against these attacks, because the attacks are working on the transaction level, not on the authentication level. PKI and other security measures are simply bypassed, and are therefore rendered obsolete.

Acknowledgements

Thanks to quintessenz.at for originating the discussion and for feedback on the paper. Thanks also due to the Anti-Fraud Coffee-Room (http://lists.cacert.org/) for feedback and FinancialCryptography.com for peer reviewing the paper. Many thanks to the various individuals who freely gave ideas and insights but prefer to remain nameless.

1 The Problem

1.1 Trojan Technology

The new trojan technology is technically more advanced than prior generations by way of combining Browser-Helper-Objects, Browser Extensions, and direct Browser manipulation techniques.

1.2 Browsers

As of time of writing, it is known that (2006-09-12) Firefox, Internet Explorer and Opera are successfully targeted, on the Windows, Linux and MacOS X Platform. It is currently not known yet, whether Konqueror, Safari, Lynx and other browser are affected too.
The trojans can do the following:

- Modify the appearance of a website before the user sees it.
- Modify the data entered by the user before it is encrypted and sent to the server. This modification is not visible / detectable by the user or the server.
- Change the appearance of transactions returned by the server back to the version that the

user expects.

1.2.1 Points of Attack

- Browser Helper Objects – these are dynamically-loaded libraries (DLLs) loaded by Internet Explorer / Windows Explorer upon start-up. They run inside IE, and have full access to IE and full access to the DOM tree, etc. Developing BHOs is very easy.
- Extensions – similar to Browser Helper Objects for other Browsers such as Firefox (hereafter, both will be referred to as extensions). Developing Extensions is easy.
- UserScripts – Scripts that are running in the browser (Firefox/Greasemonkey+Opera). Developing UserScripts is very easy.
- API-Hooking – this technique is a Man-in-the-Middle attack between the application (.EXE) and the DLLs that are loaded up, both for application specific DLLs such as extensions and Operating System (OS) DLLs. For example if the SSL engine of the browser is a separate DLL, then API-Hooking can be used to modify all communication between the browser and the SSL engine. Developing API Hooks is difficult.
- Virtualisation – running the whole operating system in a virtual environment, to easily bypass all the security mechanisms. Developing Virtualisation attacks is difficult.

1.2.2 Method of Attack

An attack may progress thusly:

1. The trojan infects the computer's software, either OS or Application.
2. The trojan installs an extension into the browser configuration, so that it will be loaded next time the browser starts.
3. At some later time, the user restarts the browser.
4. The browser loads the extension.
5. The extension registers a handler for every page-load.
6. Whenever a page is loaded, the URL of the page is searched by the extension against a list of known sites targeted for attack.
7. The user logs in securely to for example https://secure.original.site/.
8. When the handler detects a page-load for a specific pattern in its targeted list (for example https://secure.original.site/account/do_transaction) it registers a button event handler.
9. When the submit button is pressed, the extension extracts all data from all form fields through the DOM interface in the browser, and remembers the values.
10. The extension modifies the values through the DOM interface.
11. The extension tells the browser to continue to submit the form to the server.
12. The browser sends the form including the modified values to the server.
13. The server receives the modified values in the form as a normal request. The server cannot differentiate between the original values and the modified values, or detect the changes.
14. The server performs the transaction and generates a receipt.
15. The browser receives the receipt for the modified transaction.
16. The extension detects the https://secure.original.site/account/receipt URL, scans the HTML for the receipt fields, and replaces the modified data in the receipt with the original data that it remembered in the HTML.
17. The browser displays the modified receipt with the original details
18. The user thinks that the original transaction was received by the server intact and authorised correctly.

1.3 Email Clients

In principle, a trojan employing the same attack methods described above could be effective against clients for Email and Instant Messaging. Transaction details could be changed by much the same filtering process, sending the fraudulent transactions on to a server, and displaying the original.

As of the time of writing, such trojans have not been spotted in the wild as yet.

1.4 Authentication

The trojans are working on a different part of the transactions cycle to authentication. They could still work to effect the authentication, and that might be useful for harvesting account details, but fundamentally they circumvent the standard authentication concepts.

Which authentication systems are circumvented? All authentication systems that use the PC as the single channel for transaction data to the server are circumvented:

- Username+Password
- PIN+TAN
- PIN+iTAN
- Client certificates
- SecureID Tokens
- One Time Pad Tokens
- Biometry authentication
- SmartCard authentication
- SmartCard + Class3 reader authentication (with client certificates)
- Bürgerkarte / SecurityLayer (when used for Login only, which is currently often done)
- (likely also Digital Signatures with SmartCards and Class3 readers)
- No authentication (information portals)

In effect, if you are relying on any Authentication of the user (or the machine), to authorize transactions, the trojan horses can simply change the transactions without affecting the authentication.

2 Risk assessment

2.1 Current situation

Unfortunately the necessary technology for trojans is widely available now. It is not high-tech and high-priced any more. The time-to-market for active attacks is known to be less than 1 day.

Every potential target has to worry about the risk of being attacked with this technology now.

The technology is easy enough that everyone with webdesign experience can craft a successful attack.

2.2 Future situation

It is expected that this attack will be productised into widely available *construction kits.* The first generation of construction kits and compilers have already been seen in the wild. Such kits are making the attack viable for any target, with or without any financial motivation.

In the middle-term of 1 year, all client-server applications outside the browser are expected to be at risk too. (Email, Instant Messaging, ...)

3 Solution concepts

This section summarises the design and early implementation discoveries of financial institutions that have worked over the last six months to address the attack.

3.1 Secure Client

The first attempt by many is to attempt to harden the otherwise insecure clients.

3.1.1 Hardened Browser

All browser vendors should make sure that Userscripts, Extensions and BHOs can´t be easily run on SSL protected websites.

To create a secure browser, we need to disallow any extensions / browser helper objects, and compile it into one static binary.

A hardened browser could be constructed that fulfils the following security requirements:

- No Extensions possible (No BHO´s, No Active-X, No Java, No Plugins, No Extensions)
- Statically compiled
- Stripped (no compiler symbols available to guide the attack)
- No outside access (no DOM externally available)
- Tightly coupled to cryptography subsystem (no modularity between browser core and TLS engine)
- Additional binary-protection methods (encrypted executable, packing, ...)
- Do not allow Userscripts to be run on SSL protected websites

Additional properties that might be helpful:

- HTTPS only, HTTP not compiled into the browser any more
- Good security-enhancing tools included such as TrustBar, Petnames, Google Toolbar
- Implement a zero-knowledge password proof (TLS-SRP or TLS-PSK) with a UI that is tightly integrated into the browser and not easily forgeable.

An additional option for the future is the personalisation of the binaries, such that there is a distinct compiled binaries for every user, customised with their selected but hardened extensions.

Pros:

- Could be made available on every Desktop additionally to the user's normal insecure browser, so as to be used in parallel when high security sites are visited.
- Better usability than a Secure Live-Distribution
- Only few changes are needed to current industry standard browsers, mostly to strip them down and hard-compile them.

Cons:

- No reliable way for the server to identify the use of a hardened browser, and differentiate between secure and insecure client access.
- There is substantial work in creating parallel browser distro.

3.1.2 Unwriteable Distribution

An evolving solution from the open source world is that of Live-Distributions, being distributions of client operating systems running from read-only media such as CDROMs or DVDs.

- Knoppix (Linux) http://www.knoppix.org/
- Freesbie (FreeBSD) http://www.freesbie.org/
- BartPE (Windows) http://www.nu2.nu/pebuilder/

This allows the client to bootup securely, and to reboot securely any time it is needed.

Pros:

- A client achieves a very high security grade, and can likely withstand all currently validated threats.

Cons:

- It is a severe usability problem for the users to reboot their computer. For many users, losing from one to three minutes, and interrupting their entire workflow completely will be unacceptable. It is currently seen as impossible to ask the users on the large scale to reboot their computer for every small transaction they want to do. Will users need more than one computer?
- There are significant costs in production and distribution of media, and potential for unreliability.
- The approach assumes that there is no trojan in the platform BIOS, something that is likely to be challenged if the approach takes off.
- Printer drivers: A huge problem at the moment with Live-Distributions is the non-availability of printer drivers due to printing companies not making them available.
- BartPE potentially has licensing complications with its use of Windows platform.

3.1.3 Virtual Machine

A suggested variation for that usability problem is to run the secure operating system in a virtual machine. The risk on the other hand is that attacks that are used against the browsers in the local environment on the host system can also spread into the guest system by affecting the live system or the harddisk image. Two possible products are available in this category:

Qemu - http://www.qemu.org/

VMWare - http://www.vmware.com/

Pros:

- It raises the cost of the attack
- Usability is better than a Secure Live-Distribution
- Virtual machines are enjoying something of a surge in popularity at the moment, such a strategy may come with other benefits unrelated to security.

Con:

- Usability is not as good as a Hardened Browser
- It will probably not be secure for long. If sufficiently popular, it can be challenged and broken easily as VM-manipulation techniques are already quite sophisticated both for positive and negative purposes. It is likely that local attacks can be automatically applied to guest-applications from outside the VM.

3.1.4 Secure Signature Client

The Austrian security community has invented a secure electronic digital signature concept, called SecurityLayer, which defines the system requirements for a „Security Capsule“, and the interface so that any application (local or web-application) can use the Security Capsule to have a document securely displayed to the user and signed by the user afterwards.

The definition includes a secure XHTML subset (no dynamic elements, guaranteed readability, ...), the demand for a secure viewer which can display those secure documents to the user before the user can sign the document, to make sure that the user really sees what is being signed/authorized.

At the moment, demonstration Security-Capsules which are implemented in Software are available from several vendors. Their security level can be compared to Hardened Browsers.

It is hoped that a vendor will take the challenge to implement a Security Capsule in Hardware.

Pros:

- A potentially secure solution to the problem
- The demonstrations are promising

Cons:

- There are no rumours of a planned hardware implementation yet, so it is expected to take least 1-2 years until they could be on the market.

3.1.5 Class 4 SmartCard reader

A theoretical solution for a secure client would be the Class 4 SmartCard reader. Although not available, the device characteristics and security profile are relatively well understood, and appropriate.

Class1 Readers are just interfaces without any security mechanisms, Class2 Readers have a keyboard, but no display, Class3 Readers have a small display, Class4 Readers would have to have a secure viewer

2 possible implementations were heard of, but haven´t appeared on the market yet:

- FINREAD http://www.finread.com/
- Mastercard CAP (using the transaction signature mechanism)

It would be interesting to see Class4 Readers which implement the SecurityLayer specification in Hardware, and just send forward the whole request to the device, implementing the secure viewer and the signature system in the Class4 Reader.

Pros:

- It is based on secure hardware.

Cons:

- There is no sufficiently powerful and cost effective display available on the market as yet. Arbitrary transaction, image or document formats would have to be displayed correctly and securely.
- It is an additional device to the PC.
- It needs more space on the user's desk.
- There are no such devices available on the market yet, and time to market would be at least a year, assuming it were a routine production.
- Drivers would need to be installed on the computer.
- Price is likely to be high.

3.1.6 Trusted Computing

One problem of a Secure client is that it should be able to remotely attestate to the server, that it is a secure client. Otherwise the Server has to assume that the client is not a Secure client. If a secure client can be developed, and that secure client can attestate it´s security to the server, the server can distinguish trusted clients from untrusted clients, and deliver the normal application to the trusted clients, and switch to second channel authorisation (or other mechanisms) for untrusted client. This way users with a secure client can get good usability, while other users without trusted clients will have to use second channel or other mechanisms for authorisation.

The question is whether the Remote Attestation feature of TCG/TCPA platform will deliver that attestation function in practice.

Pros:

- Theoretically makes it possible for the server to distinguish between secure clients and insecure clients.

Cons:

- The proposal is a long way from deployment. Only a small part of the TCG infrastructure is designed yet, and even those parts have only been developed to an experimental level.
- TCG is years away from a usable platform.
- Remote Attestation would only address the server's need for a trusted computing base on the client. It does not answer the question as to whether the rest of the client is secure. For example, almost all the operating system would be outside the 'Trusted' base, including such elements as display graphics, keyboard and mouse.

3.1.7 External Authorisation devices

External authorisation devices are similar to Class4 SmartCard readers, with the difference that they have no connection to the PC. So the user has to enter the transaction details himself on the device, possibly entering the transaction details twice

- http://www.vasco.com/phishing/solution_sig.html
- http://www.vasco.com/phishing/solution_cards.html

Mobile phones could also be used as external standalone authorisation devices:

- http://motp.sourceforge.net/ (only authentication at the moment, but could be adapted to authorisation)

Pros:

- Likely secure enough

Cons:

- User has to enter the data twice

3.2 Second channel

If the computer cannot provide a secure channel between the application server and the user, we could use a second channel that adds the necessary security property of uncorrelated attacks:

- If the user's PC is compromised, the second channel remains un-compromised.

The following systems are candidates for secure channels:

- Telephone. This channel could be automated, but VOIP often goes through the PC!. (Note that the human voice has interesting security properties.)
- SMS messaging from the cell/mobile phone.
- HTTP connections from the cell/mobile phone.
- FAX
- Pager
- Email and IM/Chat. As these agents are on the PC, they can also be affected by the trojan, so using them just raises the cost of the attack.

We should differentiate between different applications here:

3.2.1 Transaction based Applications

Transaction-based systems have the problem of **authorisation** of the transaction, as opposed to authentication of the user. Is the proposed transaction really the transaction the user wanted done that way, or has it been modified in flight?

For transaction-based applications, the following concept has been developed by several European banks:

- User enters the transaction in the normal way, using her computer and browser on the website.
- When the server receives the transaction, it automatically sends back to the user the important transaction details and a transaction code through the alternate channel (SMS, telephone, email, ...)
- For example: „If you want to send 140 USD to account 51234829348 please enter the following transaction code: AB234DEF342“
- The user verifies the details. If the transaction details are correct, and what the user wanted, she enters the transaction code („AB234DEF342“) on the website.
- The server receives transaction code and takes that as authorisation for the transaction.

Pros:

- The two channels – web page and cell/mobile – are uncorrelated, so the attacker would have to achieve control over both channels. Two attacks on unrelated platforms is much more expensive.

Cons:

- The process is very dependent on the application.
- It is easy to make mistakes in the system design, potentially opening the door for complex exploits.
- SMS is limited to 160 characters, so only a limited number of transactions can be covered in each message (an issue for businesses).
- Telephone, SMS, FAX and Pager can have cost ramifications for the server-operators.

3.2.2 Non-Transaction based applications

Non-transaction systems are where the user needs to get data from the server, with all the traditional security properties such as confidentiality, integrity and authentication of source. For example, queries to databases.

Such non-transaction systems likely need to secure more data than transaction systems, which means the above alternate channel solution (based on low volume devices such as SMS) is not practical. Potentially, other channels such as Email and proprietory communication systems could be used.

Even if a second channel was available, it is not clear that the channel could be used in conjunction with the PC in order to secure the non-transaction systems.

3.2.3 Storage problem

A further problem that has to be considered is that some of the second channels mentioned above will store messages in the device, and afterwards display the stored message. Unless the user explicitly deletes the message, the message may be stored on the device for an indefinite time. Since the messages may contain confidential transaction details, that confidential data could be accidently

leaked.

Currently known affected second channel types:

- SMS
- Email
- Fax

These communication systems were not originally designed to transport and maintain data with confidentiality.

3.3 Secure Communication over insecure systems

If we cannot secure the communication systems, and we cannot get a secure second channel, a third possibility is to create secure communication systems on top of the insecure systems.

3.3.1 Captcha

It is frequently suggested that the transaction details and transaction code could be transmitted through pictures with the text on it. This is based on the idea that the text cannot be easily modified in an image. This is however a chimera.

Cons:

- Captchas have proven quite capable at separating blind people from non-blind people.
- They have not proven effective for differentiating between computers and humans.
- The entire image can be easily replaced by man-in-the-middle attacks.
- There are well-developed tools available for reading Captchas, created for websites that wish to franchise another website's capabilities.

3.3.2 Transaction model

The cost of the attack can be raised with a more secure transaction model:

Old Internet style model:

Client	***Direction***	***Server***
Identification	--->	Authentication
	<---	Transaction form
Transaction data entered	--->	Transaction validation
Verifying transaction report	<---	Transaction report

The traditional model generates the transaction data and sends it in one request to the server. This has the disadvantage in letting a man-in-the-middle attack insert itself in one place only. After transaction completion, a single receipt is sent back which can easily be changed as well.

Client	*Direction*	*Server*
Identification	--->	Authentication
	<---	Transaction form
Destination code is entered	--->	Destination code lookup
Verifying destination description	<---	Destination description
Transaction details	--->	Transaction validation
	<---	Transaction results

One idea is to change the transaction model from a one-shot model to a dialogue between the user and the server. In this dialogue, the user and the server „discuss“ the details of the transaction in several rounds, in such a way that it is not possible for a trojan to change details without breaking a single transaction.

This is achieved by the server giving individual answers to each bit of the information of the user.

The security comes from the earlier transaction details not being changeable at a later stage of the transaction anymore. When something is entered wrongly in the beginning, the whole transaction has to be cancelled and restarted from scratch.

Better security can be achieved, if the answers from the server not only depend on the general data the user enters, but also on secret knowledge of the server, which is derived from the user.

Pros:

- No need for extra hardware or software.
- A dialogue style transaction could additionally have a better usability for the user.

Cons:

- An attacker can launch a separate transaction in parallel in order to divine the information it wishes to display to the user.
- It raises the cost of an attack, but it does not solve the problem.

3.3.3 Delayed Webpage delivery

One interesting piece we found is that some of the attack mechanisms are only starting to work after the page is fully loaded, so it can change the (visible) contents only after the loading process is complete. On the other hand, the browsers are displaying the first part of the page already during loading it. Since the webserver can delay displaying a page, it should be possible to create a large enough timeframe, that the user in front of the browser has a chance to notice any relevant changes when the page is loaded completely.

The webpage would work like that:

```
Content-type: text/html

<html><body>Your browser has been taken over, if this text changes itself in the
next 5 seconds!</body>

<? sleep(5); ?>
```

```
</html>
```

One potential problem of this solution are possible delays of the delivery due to the webserver waiting for the page to have finished before sending it to the browser. So good control over the webserver is necessary for this trick to succeed.

Pros:

- Easy to implement

Cons:

- Might be difficult for the user to understand
- Delays the website for the user, potentially creating a bad user experience.
- Does only help against some browser engines
- Does not help against advanced attacks with API Hooking or Virtualisation

3.3.4 Using SSL

Opera´s security concept for Userscripts is that the user additionally has to allow Userscripts to be run on SSL websites. So having a website use SSL will help a bit against Man-in-the-browser attacks on Opera users, unfortunately the other browser vendors haven´t implemented the same security mechanism yet.

Pros:

- Should be done anyway

Cons:

- Does not help against advanced attacks.

3.4 Human Security – the „last meter“

One viewpoint on the problem is that current security systems and indeed much of security thinking today are not really end-to-end. One end of the transaction is normally rooted in the user's brain, but most transaction protocols only go as far as her computer. This section looks at how we can go the „last meter.“

3.4.1 Simple Encryption

Simple encryption tables printed on paper, where the user can easily encrypt the transaction details to the server.

Pros:

- Potentially creatable by the user themselves using other secured platforms
- Mobile solution – can be carried in pocket/handbag.

Cons:

- Needs Paper
- Users do not like doing sums.
- Integrity, Protection against Replay would need to be solved

3.4.2 Multi-path passwords - SecLookOn

A multi-path password works by defining graphical rules such as „If you see image X in the upper left area, then type in the number that is displayed in the image with a white colour in the upper right area.“ The user has to be trained through the multiple paths, and she has to apply the process on each login.

Pros:

- Very good security concept.
- It could be adapted into a strong authorisation solution.

Cons:

- It is quite a challenge for the user to learn the concept and memorize all the steps.
- Current implementations are authentication only, so they do not as yet solve the problem.

This concept has been implemented by MERLINnovations under the name SecLookOn: http://www.seclookon.com/seclookon/ A time-limited demonstration of it can be found at http://sec.generalibank.at/SecLookOn_Generali/ It is currently authentication only.

3.4.3 Shared Secrets - PassFaces

Another possibility is to transmit information such as graphics that the user is familiar with and has configured on the site. Since the secrets are transmitted during the dialogue, they are available to the attacker as well as the user.

Pros:

- May carry side benefits such as personalisation for online sites.

Cons:

- There are no secrets kept from the attacker.
- It covers simple authentications only. It is not clear that it is possible to adapt it for authorisation.

For example, see PassFaces: www.passfaces.com

3.5 Server side security

3.5.1 Non-Automatability

It is possible to hardening the web-application to make it hard for an attacker to change the page. For example, randomising all URLs and the HTML so as to thwart pattern matching. This would mean automatically rewriting the entire output of a web server, for each page view.

Pros:

- It raises the cost of an attack.

Cons:

- The attacker can use the user herself to teach it where to change the data.

For example, see Visonys Airlock: www.visonys.com

4 Give up

It is important to state the worst case scenario – abandon the net. If none of the presented concepts can be applied successfully to a system, and others are not found, the last alternative would be to give up using the internet for security-demanding applications.

Pros:

- It was good while it lasted.

Cons:

- Might not be a solution for all applications
- Dramatic cost implications, if the users have to switch to other channels
- Will likely cause a shock to confidence in some sectors such as retail banking.

Epilogue

On the first blush, the problem of the man-in-the-browser seemed impossible to solve. After serious investigation by banks and security workshops in Europe, a couple of plausible patches and solution ideas have emerged.

All of these solutions have challenges. Implementation, user acceptance and the basic security properties are not forgone conclusions, and only a few promise to be a sustainable solution. A lot of usability challenges will straing the roll-out of the solutions, creating great risks for operators.

Still, the alternative is yet more bleak. Early calculations suggest that if the Internet channel is to be abandoned, this will immediately effect issues such as branch staffing levels and even the number of branches. Such a path is considered unacceptable.

This paper is presented as only the beginning of the necessarily long and active research in this new field. A lot more concepts and ideas have to be discovered, peer-reviewed, implemented, tested, and rolled out. We will need many risk experiments to find better, usable and workable solutions.

(It might be helpful to collect more ideas in this paper, so I will try to update this paper as and when new ideas are available. Keep an eye on the date at the top.)

References

Visonys Airlock	http://www.visonys.com/
PassFaces	http://www.passfaces.com/
SecLookOn	http://www.seclookon.com/seclookon/
FinRead	http://www.finread.com/
Browser Helper Objects	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebgen/html/bho.asp
SecurityLayer	http://www.buergerkarte.at/
TCG	https://www.trustedcomputinggroup.org/
Captcha	http://www.captcha.net/

http://www.cr80news.com/library/2006/09/16/on-card-displays-become-reality-making-cards-more-secure/