CONTINUING BUSINESS WITH MALWARE INFECTED CUSTOMERS

BEST PRACTICES AND THE SECURITY ERGONOMICS OF WEB APPLICATION DESIGN FOR COMPROMISED CUSTOMER HOSTS

Today's media is full of statistics and stories detailing how the Internet has become an increasingly dangerous place for all concerned. Figures of tens of millions and hundreds of millions of bot-infected computers are regularly discussed, along with approximations that between one-quarter and one-third of all home computer systems are already infected with some form of malware. With a conservative estimate of 1.4 billion computers browsing the Internet on a daily basis (mid-2008 figures), that could equate to upwards of 420 million computers that can't be trusted – and the numbers could be higher as criminals increasingly target Web browser technologies with malicious Web content – infecting hundreds of millions more along the way.

Despite these kinds of warnings and their backing statistics, online businesses have yet to fully grasp the significance of the threat. Most of the advice about dealing with the problem has focused on attempting to correct the client-side infection and yet, despite the education campaigns and ubiquity of desktop anti-virus solutions, the number of infected computers has continued to rise. The problem facing online businesses going forward is, if upwards of one-third of their customers are likely to be using computers infected with malware to conduct business transactions with them, how should they continue to do business with an infected customer base?

This paper discusses many of the best practices businesses can adopt for their Web application design and back-office support processes in order to minimize this growing threat, along with helping to reduce several of the risks posed with continuing to do business customers likely to be operating infected computers.

THREAT LANDSCAPE OVERVIEW

While there are many different types and classes of malware currently in circulation, with each malicious feature designed for a specific insidious task, the net effect is that an attacker who can remotely control an infected host has the ability to observe and manipulate any piece of data going to, or coming from, the compromised computer.

The most advanced malware iterations can usually be found targeting online banking customers. Driven by the prospect of high monetary rewards, criminal malware authors have continued to invent new mechanisms to combat the latest authentication and authorization technologies deployed by the banks. Today's specialized banking malware features are routinely combined with advanced social engineering techniques, and are capable of thwarting *any* client-side security technologies – *including* most current multifactor and out-of-band authentication device implementations.

The malicious technology lying at the core of their success is the ability to proxy and manipulate Web content from *within* the Web browser. An evolutionary step from classic man-in-the-middle attack vectors, this ***man-in-the-browser*** vector means that the attacker can:

1. Observe all communications between the customer and the bank in the clear, regardless of any network-layer encryption.
2. Alter any data communicated between the customer and the bank in real-time including, but not limited to :
    a. the creation of new and additional page content,
    b. the removal and manipulation of displayed account balances,
    c. the insertion of additional fraudulent banking transactions.
3. Add new content (such as pages asking for the customers Debit card and PIN information) without alerting the customer through recognizable Web browser security features (e.g. broken padlock, mixed-domain content warnings, etc.).
4. Function wholly within the Web browser, never being visible to the banking organization, and operating transparently to the customer.

While this type of malware originated and continues to evolve in the online banking arena, its success has also seen it being adopted for other online financial frauds (such as stock trading portals). The expectation is that the man-in-the-browser feature-set will be absorbed in to other standard malware suites, and that its use in other fraud and criminal attacks will become more prevalent.

WHERE DOES THE SOLUTION LIE?

The most important factor as to why the man-in-the-browser vector is an increasingly successful vehicle for online fraud has to do with the relative complexity of current generation Web applications and their peripheral security technologies – as encountered by end-users.

Consider for the moment the number and types of Web pages a customer must navigate through (and type data in to) in order to authenticate themselves. Then consider the additional steps they must go through to actually conduct a fund transfer in a typical online banking application. While navigating through five to ten Web pages, and inputting three to eight discrete pieces of information may not be much to most IT professional, for the majority of non-IT customers this is a daunting task. When coupled with additional out-of-band and multifactor authentication devices – often requiring onscreen instructions or walk-through for their successful use – the process becomes a technology maze for the vast majority of customers, and is often destined for failure.

Application complexity can, and is, exploited by criminals utilizing the man-in-the-browser vector.

By injecting additional page content through the use of a man-in-the-browser agent, the attacker can easily "socially engineer" their victims in to surrendering the information they desire. In addition, they can coax their victims in to performing any additional out-of-band validation tasks, thereby gaining access to one-time passwords and temporal keys.

For many years Web application security professionals have been adamant that client-side validation of customer data is insecure, and must be sanitized at the server. In the face of the man-in-the-browser malware threat, not only should that data be categorized as insecure, but it may be advisable to class it as "untrusted" and not have been intentionally sent by the customer.

Because of this, developers of these Web applications must implement new server-side validation algorithms and greatly reduce the complexity of the application as presented to the customer – thereby reducing the surface area through which attackers can affect application flows and shim their malicious constructs.

BEST PRACTICES IN ANTI-MAN-IN-THE-BROWSER DEFENSES

A problem facing organizations seeking to continue to do business with customer's who are probably relying upon compromised computers to use their online services, is that the man-in-the-browser compromise vector makes practically all of the classic man-in-the-middle defenses built in to the Web application redundant. And, since the attack (i.e. fraud) is conducted while the customer is logged in from their own computer and can potentially modify any piece of data presented rendered within the browser, a new protection strategy needs to be adopted.

This section provide application design advice specifically designed to help limit the exposure to fraudulent customer transactions conducted by, or linked to, a man-in-the-browser agent running on a customer's computer.

For ease of study and implementation, the best practices advice has been divided in to the following sections:

- Application flow
- Online changes
- Back-office verification

The best-practice advice listed in this section has been provided in the context of answering the types of questions application developers and designers should be asking themselves. These are also the same types of questions that security professionals (such as penetration testers and auditors) should be asking when assessing the security of an already deployed Web application.

Note that the advice is not exhaustive and not all Web applications are created equal. Depending upon the nature of the Web application, the demands of the organization, and the needs of their customers, Web application developers should use this advice as a guide and cherry-pick the defensive stratagems most applicable.

Note also that this paper does not provide advice on other protection technologies that actively seek to secure the customers Web browser from the client-side (e.g. security toolbars, browser-specific anti-virus agents, etc.), or hardware technologies that attempt to secure data outside of a potentially compromised Web browser (e.g. USB token technologies, cell phone banking, etc.).

APPLICATION FLOW

The flow of an application, and the way in which customers are expected to navigate their way through it, has a significant impact on whether attackers can insert additional content and manipulate the customer in to surrendering confidential information.

Web application designers should seek to answer the following questions:

- **Is it likely that additional pages or fields would be spotted by a customer?**
  Given the prospect of malware "inserting" additional pages and content in to an application (from the customers Web browser perspective), can any obvious markers be used to indicate the correct flow of the application? For example, if a particular process has four steps (or pages) to navigate, the application should initially inform the customer that it is a four-step process, and clearly label each step as "Step 2 of

4" etc. as the customer completes them. Make sure that the completion or success confirmation page is also included in the numbering system.

To help increase the customer identification rate of additional form fields injected by the attacker within an existing page, the application could use a fragmented image background (within the form constructs) that only renders as an intact image if all the form fields are in the correct alignment and no additional fields have been added.

- **Is it clear to the customer what's expected of them?**

  What information is publicly available to customers so they can appreciate how application is meant to operate, and how to navigate any key processes? If customers are unfamiliar with the Web application, or if the application regularly changes, it is advisable to provide a step-through guide (complete with images) of how they should use it. Ideally these instructions would be available in a printed format and mailed to customers. Alternatively, the instructions could be provided in a PDF format form, and hosted on another domain.

  Customers should be officially informed of any changes to the visible application alterations in advance of actually logging in. Ideally, these advanced warnings should direct customers to review the updated guide.

- **How many pages must customers navigate or scroll through?**

  Wherever possible, developers should seek to minimize the number of pages that a customer must successfully navigate in order to use the application and complete their objective. However, a balance must be reached with the volume of information presented on a single page. Ideally all information should fit within a single screen and not require the customer to scroll down the page.

  If confronted with the quandary to make a particular task two pages, or one page with an element of scrolling, the later solution is to be preferred.

- **Are all the steps logical?**

  The process steps through which the customer must successfully navigate should be as clear and logical as possible. If a process (such as conducting a intra-bank transfer) must be divided in to a number of distinct steps by the application, each step should be in a logical order – ideally trying to follow a physical process with which the customer would be most likely already be familiar with (e.g. the process of filling out and signing of a check).

  The purpose of constructing the process this way is so that customers can help identify any rogue steps that may have been inserted by an attacker, as well as reducing the overall complex nature of having to learn an unfamiliar process.

- **Are important questions and steps presented as text or as graphics?**

  Today's man-in-the-browser technologies largely rely upon the scripted insertion of text content in to the pages being rendered within the Web browser. It becomes a more complex and onerous task for the attacker if he has to contend with the manipulation of informational graphics and, if the attacker tries to link to alternate remotely stored graphics, native browser security functions would also likely alert the customer (e.g. mixed security zone warnings).

  Wherever possible, key questions and stage information (e.g. "Step 2 of 4") should be rendered in a graphical format.

- **Can visual anti-tampering techniques be applied?**

  Graphical border trims that surround the key text of the page can provide a visual indicator of additional attacker-inserted form fields. Border trims that are constructed in such a way that HTML graphic manipulations (e.g. resizing, scaling and duplication) would be obvious are to be recommended – for example, a border graphic that consists of the institutions full name and logo, in miniaturized form, repeated continuously in graphical format, and encircling the critical for input areas (e.g. like the anti-counterfeiting techniques used in official documents).

Ideally the graphical border trim would not be a single graphic. Instead, it should be constructed of multiple segments that would not render as being visually contiguous if any piece was duplicated, or the build order changed.

- **Could the application interface be simplified further?**
  At the core of the application build process, developers and designers should constantly ask themselves whether the interface could be simplified further, and whether the number of steps the customer must follow in order to complete any task can be reduced further still.

ONLINE CHANGES

Assuming that the man-in-the-browser vector successfully manipulates the customer's experience of the Web application and is capable of usurping control, additional verification and visualization techniques should be employed by the application designer to alert the customer of any new transactions or changes associated with their account.

Web application designers should seek to answer the following questions:

- **Can the customer change *everything* online?**
  While tempting to offer the facility for customers to be able to view and subsequently change any of their stored personal information online, developers should strongly resist. Some categories or pieces of information should always require an alternative method of access for review and change, and never be accessible (or visible) though the Web application.
  Information that cannot be viewed or changed online can then be more reliably used for other out-of-band validation purposes – e.g. confirmation of changes through automated phone systems.
  Ideally customers should be educated as to which types of information would never be visible from within the application, and be alert to any requests within the application for that personal information.
- **What out-of-band verification of changes are there?**
  If confidential or personal information details are available for change online, there must be processes for the out-of-band verification of those changes. This is critical for changes to key contact information such as delivery address details, phone numbers and email addresses.
  Out-of-band verification could encompass the delivery of a "change notification" (a message confirming the request for the data change, along with date and time information – but ideally not revealing the content of the change) to a pre-agreed address – e.g. a letter to a postal address, an SMS text to a cellular phone, or an automated voice message to a home phone number.
- **Are change notifications sent to previous contact details?**
  A common online fraud tactic is to alter the contact details of where out-of-band alert messages are delivered – thereby preventing the victim receiving any subsequent fraudulent alerts.
  Ideally, changes to any alert-related contact details (e.g. email address, home and cellular phone numbers and postal address) should result in a change confirmation notification being sent to *both* the new and old contact addresses.
  The content of the notification should not contain details of what the data was changed to, but should contain advice on how to proceed if the address change was not intentionally instigated by the customer.
- **Are there delays before going "live" with contact changes?**
  The activation of any changes to contact details, in particular those associated with the alerting of account changes of the validation of transactions, should not be instantaneous. Since a common criminal tactic is

to change alerting details and promptly proceed with the fraud before the customer is alerted to the changes, the inclusion of delays before critical account changes are activated should be considered.

The delays before changes are made "live" should be proportional to the amount of time it would be reasonable for a customer to receive notification of the information through the alerting systems they have chosen – taking in to account factors such as system delays, weekends, holiday periods, and possibility of the alert being "lost". For example, SMS text alerting of changes may take several days to be received for customers that travel, while postal mail alerts may take upwards of a week for national residents and four to six weeks for account holders that live abroad.

For critical contact information changed online, it may also be advisable to repeat an alert message after a after few hours, or the following day.

- **How visible are customer initiated changes?**

  Changes to customer information, whether they are initiated through the Web application or other methods (e.g. telephone service desks or "in-person" branch visits) should be made visible to the customer – in an obvious, but unobtrusive manner.

  Depending upon the nature and sensitivity of the data, the changes should ideally identify the type of data changed (e.g. "Postal address changed", "Cell phone alert number changed", etc.) and they should be clearly visible on the main customer menu page once they have successfully authenticated themselves, and on any other relevant transactional pages. The notification should also include the date the change took place and, if the change occurred through mechanisms outside the Web application, it may be appropriate to indicate the method of change.

- **How far does the change history reach in to the past?**

  The display of past changes to critical or personal information should not be unduly limited to a single "last change" notification – nor should it be overwritten by other innocuous changes and updates (e.g. "Last logged in from:"). Ideally, these kinds of notification should exist, and remain visible, for a period of time proportional to the periods over which the account would normally be accessed in a worst case (e.g. if a customer normally conducts business through the Web application on a weekly basis, and the longest period between logging in over the last 12 months has been three weeks, the change notification should exist for at least four weeks – and preferably 6 weeks).

  Application developers should also be cognizant of the fact that attackers may exploit systems that limit the number of recent changes by relying upon a scrolling history window (e.g. only listing the last three changes), as they may simply "overwrite" these alerts with other less obvious changes (e.g. overwriting a "Postal address changed on 01/01/09" with three "Last logged in from: 123.321.123.321").

- **Are transaction histories available in HTML and Print/PDF for reconciliation purposes?**

  Since the man-in-the-browser vector can provide the attacker with the facility to alter any data displayed inside an infected Web browser, it is important that methods exist for customers to reconcile historical transactions and account changes.

  The use of alternative or multiple in-application reporting mechanisms is to be recommended. The ability to access monthly statements in formats such as Acrobat Reader or Microsoft Excel can facilitate the reconciliation process for customers – all the while making it more difficult for the attacker to modify the data.

  While not a particularly "Green" strategy, printed statements should be made available to customers – along with onscreen instructions on how to reconcile between the different mediums.

  Other out-of-band update systems could also be considered, and aid the reconciliation process – such as a weekly automated SMS text messages listing the number of transactions (and perhaps the transaction type and account balance).

Weekly or monthly statements should also include any change histories (e.g. contact phone number changes) – regardless of the source of the change (e.g. telephone, "in-person" branch visit).

BACK-OFFICE VERIFICATION

As previously discussed, there are considerable security and integrity benefits to be had by simplifying the customer's experience of the Web application and reducing the overall level of complexity of the steps through which they must navigate in order to complete a task. Wherever possible security-related tasks should be handled in the "back-office", and be invisible to the customer.

A key element to the reduction of client-side security obtrusiveness, and the strengthening of overall transactional security or customer integrity, lies with the efficient use of back-office correlation and anomaly detection processes.

Application developers should carefully consider the following points:

- **How are error messages, such as failed transactions, rendered and presented to the customer?**
  It is frequently pointed out that warning and error messages which require an application user to make a choice between stopping or continuing what they were doing, will almost certainly result in the user choosing to continue – regardless of what the message contains. As such, it is advisable that application developers not force customers to make decisions based off error and warning messages. For example, if the customer fails to type in the correct transaction authorization code, they should receive a warning that they failed to complete the transaction and are taken back to the main menu screen (perhaps automatically after a few seconds) – rather than presented with another page asking them retype the code and "try again".
  In the cases where customers *fail* authentication steps related to the answering of specific secret questions (e.g. "What make was your first car?") or characters of a memorable passphrase (e.g. "provide the first, second, fifth and eighth characters of your password"), it is important that exactly the same question(s) be asked the next time the customer tries to log in. This is important because some malware types that have recorded answers to precious questions will repeatedly attempt to re-initiate a login or transaction validation until a question appears exactly like the one it had previously observed the answers to.
  Ideally, any failure – whether that be during the customer authentication or transaction authorization phases – should result in an alert being sent to the customer through a pre-agreed out-of-band communication system (e.g. SMS text message).
- **Does the back-office system implement thresholds on transactions per minute?**
  Several current-generation man-in-the-browser malware implementations seeks to "piggy-back" additional fraudulent transactions on top of the legitimate transaction the customer is trying to do. A key component of this process is the tricking the customer to provide an additional transaction validation value – typically by presenting a fake "authorization failed" message, followed by a "please try again" message. Application developers should consider tracking the time taken between transaction requests (especially relevant if the application uses unique 'seeds' each time a customer starts a new transaction), and set minimum thresholds for either alerting or blocking an account. These thresholds should take in to account the speed in which a real customer would navigate the application, type in the required data, and server response times.
  In addition, a common tactic of some malware is to instigate several small value transfers (designed to stay below back-office transaction value alerting thresholds) while the customer is logged in to the

application. Therefore, application developers may wish to monitor how many transactions are conducted within a single customer session, or monitor how many in a predefined period of time.

- **Is there a delay between creation of a new "payee" account, and ability to transfer money to that account?**
  Financial organizations should consider when a customer can transfer funds to a newly created "payee" account, and whether some level of alerting (and logging) should be provided to the customer each time a new payee is added. Ideally, some kind of delay should occur between the creation of a new payee and the transfer of funds to it. This delay may be visible to the customer (e.g. "payment requests to newly created payees can only be setup after 48 hours") or, preferably, invisible to the customer while the back-office system assumes a higher degree of monitoring and alerting over new payee accounts.
  Back-office anomaly detection algorithms can be employed to monitor newly created payee accounts, and correlated over multiple customer accounts in order to identify an organized fraud attempt (e.g. multiple customer computers have been infected, and the attacker is trying to shift funds from multiple accounts to a single account they own (or a small group of money mule accounts)).
- **Do customer-initiated identity, alert and contact information changes make sense?**
  To reduce the prospect of their fraud being detected by the customer, attackers will often seek to harvest as much personal information about the customer as possible, and alter any contact or alerting information. Developers should ensure that the back-office fraud detection systems are capable of identifying address and contact information discrepancies. For example:
  - If the customer changes their home phone number, is the area code for the new number applicable for that physical address?
  - Is the new cell phone number (or area code) associated with known VoIP messaging portals, or used in other customer accounts?
  - Should it be deemed suspicious if the customer changes their cell phone number, and then changes their out-of-band alerting mechanism from postal mail to SMS text alerting in the same session? Is it more suspicious if they subsequently initiate a new fund transfer?

CONCLUSIONS

As businesses begin to truly grasp what the infection rates for home PC's really means for online business going forward, they will find themselves having to adopt different security models in order to help reduce or displace the escalating threat. Not only having to protect against classic malicious attacks and data corruption, online businesses must also now contend with the compromise of the end-point trust relationship.

The man-in-the-browser attack vector has major implications on Web application design, requiring a considerably different suite of protection strategies from what has been adopted in the past. The core requirement for limiting the threat lies with the simplification of the "customer experience" – i.e. the drastic reduction application complexity – thereby reducing the attacker's capability to "shim" the application, disguise fraudulent activities and socially engineer customers in to performing additional out-of-band validation activities.

Additional Reading:

http://www.techzoom.net/publications/insecurity-iceberg/index.en

http://www.technicalinfo.net/papers/HostNamingAndURLs.html

http://news.bbc.co.uk/1/hi/business/6298641.stm

http://www.owasp.org/index.php/Man-in-the-browser_attack

http://www.f-secure.com/weblog/archives/VB2007_TheTrojanMoneySpinner.pdf