2010

Fireguard - A Secure Browser with Reduced Forensic Footprint

Don Griffiths
Curtin University of Technology

Peter James
Edith Cowan University

DOI: 10.4225/75/57b2950a40cdc
Originally published in the Proceedings of the 8th Australian Digital Forensics Conference, Edith Cowan University, Perth Western Australia, November 30th 2010

Fireguard – A Secure Browser with Reduced Forensic Footprint

Don Griffiths[1] and Peter James[2,3]
[1]School of Information Systems
Curtin Business School
Curtin University of Technology
Western Australia
don.griffiths@cbs.curtin.edu.au

[2]Secure Systems Ltd
Balcatta, Western Australia
pjames@securesystems.com.au

[3]School of Computer and Security Science
Edith Cowan University
Perth, Western Australia

Abstract
Fireguard is a secure portable browser designed to reduce both data leakage from browser data remnants and cyber attacks from malicious code exploiting vulnerabilites in browser plug-ins, extensions and software updates. A browser can leave data remnants on a host PC hard disk drive, often unbeknown to a user, in the form of cookies, histories, saved passwords, cached web pages and downloaded objects. Forensic analysis, using freely available computer forensic tools, may reveal sensitive and confidential information. A browser's capability to increase its features through plug-ins and extensions and perform patch management or upgrade to a new release via a software update provides an opportunity for an attacker to embed malicious software and subsequently launch a cyber attack.

Fireguard has been implemented using both Mozilla Firefox and the storage and protection capabilities of the Mini-SDV, a secure Portable Execution and Storage Environment (PESE). In this paper the design and development of Fireguard is discussed. The requirement for a secure PESE and the functionality of the Mini-SDV is presented. An overview is given of the motivation for the development of Fireguard. The reasons Firefox was selected and the Firefox structure and security vulnerabilities are summarised. The implementation approach adopted is discussed and the results of an analysis of the Firefox implementation are presented. The Mini-SDV configuration for Fireguard and an outline of the concept of operation is given. The changes made to Firefox to implement Fireguard as a browser that reduces the opportunity for data leakage and cyber attack, and minimises its forensic footprint are discussed. The paper concludes by considering the strengths and limitations of the Fireguard implementation.

Keywords
Secure Browser, Secure Portable Execution and Storage Environments, Mozilla Firefox, Mini-SDV, Computer Forensics, Anti-forensics.

INTRODUCTION

Organisations that enforce best practice ICT security, and with a workforce consisting of remote and mobile[1] workers, often restrict the use of portable storage media and the use of remote connections to corporate servers due to security concerns relating to data leakage and cyber attacks. The risk of data leakage (NTI, 2004) from loss of portable media and/or the recovery of sensitive data remnants from hidden storage areas on PCs, and cyber attack from malicious software (malware), introduced onto a corporate system from the Internet and/or portable media (Moscaritolo, 2008), are amongst some of the key security issues confronting organisations (Lemos, 2010; McAfee 2010; Ponemon, 2009). Secure portable storage media (TrueCrypt-Foundation, 2010; IronKey, 2010) has enabled organisations to define policies for the secure transport of sensitive corporate data by remote and mobile workers. However, these products do not provide strong protection for trusted applications which can be uploaded and executed on a host PC to enable secure remote and mobile computing.

[1] In this paper a remote or mobile worker is considered to be an individual who works in a location where the physical and logical security controls do not necessarily satisfy the organisational IT security policy defined for the corporate IT environment.

Secure Portable Execution and Storage Environment (Secure PESE)

A secure PESE provides both a solution for the transportation of sensitive data and the provision of a trusted application to enable secure remote and mobile computing. The secure PESE is recognised as a way to prevent data loss and cyber attack by organisations who process sensitive data, yet want to achieve cost savings and productivity gains from allowing remote and mobile working. A secure PESE trusted application can be defined as an environment that limits data leakage, cannot be compromised by malware and leaves no forensic footprint on the host PC upon which it executes, i.e. no evidence exists on the host PC hard disk drive (HDD) that the trusted application has executed on the PC. A secure PESE trusted application can be a bootable environment that is uploaded from the secure PESE when the PC is powered-on, or an application that can be uploaded to a host PC with its own booted and executing operating system (OS).

A functionally strong secure PESE (James, 2008) will include a USB or an eSATA interface, different modes of operation to suit different operational environments, strong pre-boot and post-boot authentication, hardware based encryption of the device's storage medium, a choice of trusted applications (e.g. bootable or executable on the host PC OS), protection mechanisms for the trusted applications to protect against malware, and a storage area for temporary data (to prevent data remnants residing in hidden storage areas).

Mini Silicon Data Vault (Mini-SDV)

The Secure Systems Mini-SDV (Secure Systems, 2010) is a USB attachable secure PESE with the following functionality:

- Two modes of operation; Fully Trusted and Assured.
    - Fully Trusted mode allows an authenticated user to 'boot' a trusted application from a Mini-SDV connected to a host PC. The trusted application only uses the CPU and RAM of the host PC and leaves no forensic footprint.
    - Assured mode allows an authenticated user to execute a trusted application (the Fireguard browser) from a Mini-SDV connected to a host PC that is executing the Windows OS. Fireguard leaves a minimal forensic footprint.
- Hardware based encryption of the storage medium.
- Partitioning of storage medium. Partitions can be defined as Read-Only, Read-Write and No-Access.
- User/role profiles with differentiated access rights.
- Storage area for temporary data created by the trusted application.

The Mini-SDV functionality binds together to enable the device to be configured to provide a strong and secure PESE for remote and mobile computing. The Mini-SDV functionality is configured to counter the remote and mobile computing risks of cyber attack and data leakage by:

- Securing a trusted application in a Read-Only partition that prevents malware becoming embedded into the application and hence the exploitation of the malware to launch a cyber attack.
- Full encryption of the integral storage medium to protect the confidentiality of data and prevent data leakage if the Mini-SDV is lost or stolen; and
- Storing temporary copies of data created by the trusted application in an allocated storage area on the Mini-SDV which prevents data leakage through the recovery of the data remnants.

The Mini-SDV hardware and firmware provide the platform to counter cyber attack and data leakage, but trusted applications are also required, that utilise the Mini-SDV's functionality, to provide a fully functional secure PESE. The Mini-SDV trusted applications available in Fully Trusted mode consist of a minimal functionality OS, secure browser and secure Windows terminal client. Fully Trusted mode is used when no trust can be asserted about the host PC. This mode of operation is not considered further in this paper. In Assured mode a secure browser, Fireguard, is available. The Assured mode should be used when either a level of trust can be asserted about a PC or if an untrusted PC is used to perform limited Internet transactions through a secure browser. In this paper the development of the Assured mode secure browser, Fireguard, is considered. In particular the paper discusses how an open source browser was modified to:

1. Allow it to be installed on to, and be protected by, a Mini-SDV.
2. Disable features that could allow malware to become embedded in the browser.

3. Prevent any temporary, configuration or reusable data generated by the browser being written to the host PC's HDD.

As a result of the above three modifications a secure browser with reduced forensic footprint was produced that limits the opportunities for both cyber attack and data leakage.

FIREGUARD PROJECT – MOTIVATION AND APPROACH

Motivation and Requirements

The motivation for the development of Fireguard was driven by the requirement for a remote or mobile worker to be able to use an available PC and connect to the Internet with the assurance the session would not be compromised. For instance, if a remote or mobile worker accessed a web site, through a browser available on a public access PC, there is a possibility that the browser may have been compromised through malware caused by infected plug-ins, extensions[2], Javascript[3] or software updates. This browser will also save a range of temporary, configuration and reusable data that can remain on the PC in hidden directories after the browser session has completed and may be recovered, using basic forensic tools, to reveal information about the user.

Two important design objectives were that the user could neither add functionality to the browser through plug-ins, extensions and software updates (to ensure the user does not inadvertently introduce malware into the browser), nor allow the user to accidently cause sensitive data to remain on the host PC. Fireguard was therefore developed to provide a secure browser that could neither be compromised by malware exploiting browser vulnerabilities, nor allow data remnants to be recovered that could reveal details about the user.

It was decided that Fireguard should be a browser that executes securely on the Microsoft Windows range of OS'. The following functional requirements were identified for the Fireguard project:

1. To prevent malware and hence cyber attack it must be executable from a Mini-SDV Read-Only partition.
2. To prevent malware and hence cyber attack it must prevent plug-ins, extensions and software updates.
3. To prevent forensic discovery it must not update the Windows registry[4].
4. To prevent data leakage and forensic discovery it must store configuration data and temporary files in a nominated storage area on the Mini-SDV.
5. To prevent data leakage and forensic discovery it must not write to the Host PC HDD, with any created temporary data being written to the Mini-SDV and deleted when the browser terminates.

Implementing these five functional requirements will protect the integrity of the browser's source code, prevent malware becoming embedded into the source code, prevent vulnerabilities in the browser being exploited, prevent data leakage and as far as is practical leave no forensic footprint.

Browser Selection

An investigation was performed into popular browsers to determine if any of these browsers could be configured to achieve the five requirements (defined above) without having to make changes to the source code and user interface, i.e. achieving the five requirements through a browser's user configuration capability. None of the available proprietary browsers provide a configuration capability to achieve the desired outcomes. Whilst configuration settings could enable a number of the requirements to be satisfied, it was determined that changes to a browser's source code and user interface code were necessary to implement all of the five requirements. However, for proprietary browsers access to the source code is not possible, for example Microsoft Internet Explorer (IE) is Microsoft proprietary software and access to its source code is not available. Also a key user requirement for the secure browser was that it should be configurable to have a look and feel similar to the IE browser, because IE is the most popular browser available with a 62% market share (NetMarketShare, 2010).

[2] Plug-ins and extensions are sometimes used to describe the same capability. Both browser plug-ins and extensions are additional software components created after the release of the browser that increases the browser's functionality. A plug-in can be defined (and differentiated from an extension) as implementing a narrow specific feature. Where as an extension implements a new capability or modifies an existing capability. Once integrated, extensions form part of the browser's functionality; an implemented extension can provide the basis for new plug-ins to be added.

[3] Javascript is a language used to add interactive features to webpages and can be executed within the browser.

[4] The Windows registry is a database of configuration settings and global constants/variables for both the OS itself and applications that execute on the OS.

The development was therefore restricted to open source browsers; Mozilla Firefox and Google Chrome were both considered. It was decided to use Mozilla Firefox because it could be configured to have a similar look and feel to IE, had a number of built-in security features and at the time of the development planning, in late 2009, Google Chrome had been formally released only ten months earlier. At the time development work commenced in January 2010, Mozilla had released its stable Firefox version 3.6.

An obvious version of Firefox to consider was the Firefox Portable Edition as it has been designed to be uploaded from portable storage media and execute on a host PC. However, Firefox Portable Edition modifies the Windows OS registry, which is a necessary aspect of its implementation. Firefox Portable Edition was therefore not selected because registry modifications leave evidence of browser use. It was decided that Fireguard would be based upon Mozilla Firefox version 3.6 (Mozilla 2010). The Mozilla Foundation licence requires that once the source code has changed the resultant product is no longer a Mozilla product and therefore this resultant product can not use the Firefox name and logo. The name Fireguard was selected as a 'security spin' on the name Firefox.

AN OVERVIEW OF THE FIREFOX IMPLEMENTATION

Firefox – Structure and Vulnerabilities

The first version of Firefox, version 1.0, was released in November 2004. By 2010, Firefox was rated as the second most popular browser in the world with a 24% market share (NetMarketShare, 2010). Firefox is written predominately in C++ with user interfaces written in a mixture of XUL[5] (MDN, 2010a), XBL[6] (MDN 2010b), Javascript (MDN 2010c) and CSS[7] (PHPforms, 2010).

Firefox utilises an installer to create the Firefox folder structure, establish the required registry entries and install the Firefox binary, configuration files and other associated files and data onto the PC HDD. Firefox is a functionally rich browser that allows a user to individually tailor the browser through the addition of extensions and plug-ins which can be added after the browser has been installed.

Firefox provides a good level of built-in security capabilities including:

- Phishing[8] detection – the browser maintains and constantly updates a list of known fraudulent web sites, warning the user when the site is about to be accessed.
- Malware detection – the browser warns of web sites known to contain malware. Also Firefox can integrate with anti-virus software installed on the PC to scan any downloads performed using the browser.
- Master Password – the browser allows passwords required by specific web sites to be retained so subsequent rapid logons are achieved. To protect these passwords from other users a master password can be set.
- Site identity: Allows the user to confirm a web site is genuine and confirm the level of security enforced.

Whilst these built-in security capabilities provide good security for general browsing, Firefox is still susceptible to cyber attack and data leakage through the following vulnerabilities:

- *Extensions*: Firefox extensions are programs typically written in Javascript. An example extension is the language translation capability ImTranslator (Smart Link Corporation, 2010). Most Firefox extensions do not originate from the Mozilla organisation and have been used (accidently or intentionally) on occasions to introduce security vulnerabilities into the browser (Barth, A., A. Felt, et al.,2010; Help Net Security, 2009).
- *Plug-ins*: Firefox plug-ins are dynamic link libraries (DLLs) written to conform to an application programming interface (API) provided by Mozilla. An example of a plug-in in common use is the Adobe Flash player (Adobe 2010). Browser plug-ins have been identified as a source of security vulnerabilities (Symantec, 2010).

[5] XUL (XML User Interface Language) is a markup language; it is an application of XML that defines various user interfaces elements.
[6] XBL (XML Binding Language) is a markup language; it is used to define the behaviour of XML elements.
[7] CSS (Cascading Style Sheets) is a language used to describe the formatting of a document written in a markup language.
[8] Phishing is the fraudulent act of masquerading as a genuine entity with the intent to acquire sensitive information from a user.

- *Software updates*: Firefox automatically fixes functionality problems and security vulnerabilities, and when a new version of the browser is released, the user is given the option to upgrade. Recently some research has highlighted how the software update process may be vulnerable to the introduction of malware (Cnet News, 2009).
- *Javascript*: Javascript has been the source of browser vulnerabilities; clickjacking[9] is a good example of a malware technique that utilises Javascript (Zalewski M, 2009). Firefox provides the option to disable Javascript so that a web page using Javascript will not interact with the browser. However, disabling Javascript in Firefox can significantly reduce the range of web sites that can be accessed.
- *Media Autoplay*: Firefox allows detected audio and video media files on a web site to automatically play. Media files can be a source of malware; also the media player launched to play the media file may be compromised. (Symantec 2010)
- *Java Applets*: A number of web pages use Java applets[10], particularly for interactive content. Firefox allows Java applets to execute if both the Java Runtime Environment has been installed on the host PC and (more recently) if a special browser plug-in has also been installed. Java applets can be a source of malware. (Reynaud-Plantey, 2005)
- *Downloaded Objects*: When an object is downloaded from a web site Firefox will place it in a default location if a specific storage folder is not nominated by the user. If the default storage location is used and the user does not remove the object at the end of the browser session then the object may be accessed at a later date (by an unauthorised individual) and reveal sensitive information about the user.
- *Residual data*: Firefox, like all browsers, retain cookies[11], site visit histories, saved passwords, form data[12] and cached web pages. This residual data, although hidden, is vulnerable to recovery by a knowledgeable user to reveal details about the user and his use of the browser, thus the browser can facilitate data leakage.

Fireguard was developed to prevent the exploitation of the above vulnerabilities.

Firefox Functionality Configuration Files

Firefox utilises multiple configuration files to configure functionality, most of which are located in sub-folders in the folder in which Firefox is installed. Firefox configuration files are generally Javascript files with the extension ".js" and contain calls to the function "pref" that is used to set preferences, e.g. *pref("browser.cache.offline.capacity", 512000);* sets the maximum capacity of the Firefox offline cache.

In addition to the configuration files in the Firefox installation folder, there is a file "*prefs.js*" located in the user profile. This is created from user preferences set either through the Tools >Options and other Firefox menus or by using the about:config[13] uniform resource locator (url) in the Firefox address bar. The settings in "*prefs.js*" may override settings from the configuration files in the Firefox installation folder unless a preference has been locked.

It is possible to direct Firefox to always load a custom preferences file by creating a *loadcustom.js* script that points to another custom script file. This custom script file is located in the Firefox installation folder and by convention is named *firefox.cfg*. Preferences may be set in the *firefox.cfg* file using the function "*lockPref*" (mozillaZine (2010), e.g. *lockPref("privacy.sanitize.sanitizeOnShutdown", true);* clears all data accumulated during a browser session including histories, form and search data, cookies and active login data. The "*lockpref*" function can be used on a range of preferences to prevent specific settings from being modified.

[9] Clickjacking is a malicious software technique used to trick a user into performing undesired actions by clicking on a concealed link. For example the technique is used to show a set of dummy buttons which overlay another page; the user thinks he is clicking the visible buttons but is actually performing actions on the hidden page.

[10] Java applets are small applications that perform a specific task; they provide web applications with interactive features that cannot be provided by HTML. Applets execute within the context of the browser or a plug-in.

[11] Cookies are information placed on the browser's PC by the web site and that the browser provides to the web site upon subsequent visits. A cookie can contain a variety of details.

[12] Form data is the data entered into a web page which can be saved in Firefox and re-used automatically at a later date to pre-complete a form.

[13] The about:config url allows the Firefox configuration settings to be viewed and modified within the browser itself.

Lockpref, like "*pref*", sets a preference value but prevents the user from altering the setting via menus or the about:config feature, and any corresponding settings in the *prefs.js* file are ignored. The user will see a greyed-out menu option or locked setting in the about:config display. The "*lockPref*" method of preference locking is best used in an environment where Read-Only access may be enforced to the program folder to prevent a user from editing the custom preference file, e.g. the Linux OS that supports Read-Only directories or a device like the Mini-SDV which enforces Read-Only partitions.

Firefox User Interface Configuration Files

Most of the Firefox user interface is implemented in the XUL. XUL uses XML, XBL, Javascript and CSS to specify the interface elements. The XUL files are concatenated into jar[14] archives and stored in the "*chrome*" folder of the Firefox program directory. The XUL files may be extracted and modified to change the interface without having to recompile the Firefox source code.

ANALYSIS OF FIREFOX IMPLEMENTATION

Analysis of Firefox Source Code

Firefox has been written to be a multi OS application and includes "conditional defines" for the code to be built under Microsoft Windows, Apple OS X and Linux/Unix running X-Windows. Therefore understanding how Firefox interacts, utilises and depends upon an OS is important. An analysis was performed on how Firefox interacted with the Windows OS file system and registry; in particular an understanding was required on the Firefox dependencies for constant special item ID list (CSIDL[15]). The Firefox input and output functions were analysed for Microsoft Windows use of CSIDL and environment variables to identify the locations where Firefox read and wrote Firefox user profiles[16] and temporary files. An analysis was performed of the use of the CSIDL locations in the source code to gain an understanding on how to change the functionality that read and wrote browser generated data. The objective of the analysis was to identify all Windows OS file system and registry dependencies in the source code so that the appropriate code could be changed to ensure no writes occurred to the file system or registry. The analysis identified the following:

- Multiple entries in the registry are created during installation including but not limited to:
    - Identifying installation directory, execution path and location for shared extensions.
    - Specifying additional file types that can be processed by the browser.
    - Identifying the uninstall activities.
- The Firefox installer creates an application folder in the location pointed to by CSIDL_PROGRAM_FILES.
- A shortcut to the firefox.exe is created in the location pointed to by CSIDL_DESKTOP.
- On the first execution of Firefox (i.e. the firefox.exe program) by a user, a Mozilla folder and a Firefox user profile are created in the location pointed to by CSIDL_APPDATA .
- Plug-ins provide their own installer and will create their own install folders and registry entries.
- At the start of browser execution a search is performed for available plug-ins.
- Firefox software updates are written to the program directory (CSIDL_PROGRAM_FILES) to replace the updated executables.
- Firefox stores cache data in a location pointed to by CSIDL_LOCAL_APPDATA.
- Firefox stores temporary data in the location pointed to by the system environment variable TEMP, which is generally mapped to a location in CSIDL_LOCAL_APPDATA.

[14] A jar archive combines a number of files into a single compressed archive file.

[15] A CSIDL is a Windows feature that provides a unique way to identify special folders and variables, but which may not have the same name or location on any given system.

[16] A user profile is a set of files containing user browser information including bookmarks, site visit histories, user preferences and passwords.

- When a Firefox extension is installed, it is copied to the Firefox profile or if the extension is to be shared with other user instantiations of the browser then it is installed in a custom folder (this type of installation creates a registry entry so that Firefox knows where to locate and load the extension).
- Firefox allows the user to select an image from the Internet to save and set as the desktop background image, which involves a write to the registry and the copying of the image to the Mozilla/Firefox directory in the user's home directory.

The analysis provided a good understanding of the Firefox implementation and how changes could be made to develop a secure browser.

Implementation Approach

Implementation of the five requirements identified above required both changes to the Firefox configuration settings/files and modification to the Firefox source code. Many of the required changes could be made without changing the Firefox source code and therefore where possible, changes were performed by using configuration settings. However, it was necessary to make changes to both the browser's C++ source code and the XUL user interface. All work was performed on a development platform with the following implementation approach used:

Configuration Settings: The *Fireguard.cfg* file was created which contained a series of "*lockPref*" functions to disable functionality. A *loadcustom.js* script was also created that pointed to the *Fireguard.cfg* file to enable Fireguard to disable the required functionality upon start-up. The *Fireguard.cfg* file was co-located with the Fireguard executable in the Fireguard installation folder on a Mini-SDV Read-Only partition to prevent user changes.

C++ Source Code Changes: A C++ development and test environment was established to enable source code changes to be made, compiled and tested.

XUL User Interface Changes: The user interface files were extracted from jar archives, changes were made and placed back into the jar archives; compilation of the changed XUL files was not necessary.

In a number of cases the source code was changed to completely remove a feature and also the respective feature user configuration settings were disabled for both completeness and to ensure the user could not try to enable a feature that had been removed. The use of the "*lockpref*" function to disable functionality defined in the *Fireguard.cfg* file and the integrity protection mechanism of the Mini-SDV Read-Only partition enabled many of the data leakage and forensic recovery vulnerabilities to be addressed; supported by only minimal source code changes.

Once all the development and testing was complete, the Fireguard binary, configuration files and other associated files and data held on the development platform were imaged onto the Mini-SDV.

IMPLEMENTING FIREGUARD

Key Design Decision

Although a source of vulnerability it was decided not to disable Javascript in Fireguard for the following reasons:

- Javascript is used extensively by web sites. Disabling the capability in Fireguard would restrict the sites that could be accessed, resulting in both a browser usability issue and a limitation during testing.
- The primary objective of Fireguard is as a secure portable browser to enable remote and mobile workers to access (predominately) trusted corporate systems from untrusted PCs. The use of Javascript by trusted corporate web sites would not be expected to compromise Fireguard.
- Preventing a user adding functionality and protecting Fireguard's integrity through the use of a Mini-SDV Read-Only partition capability will prevent hostile Javascript becoming embedded in the browser.

It is acknowledged that retaining Javascript presents a risk to exploit browser vulnerabilities and hence launch a cyber attack, but for the first release of Fireguard it was considered to be a risk that could be managed.

Mini-SDV Configuration

The Fireguard executable is installed in a Read-Only partition on the Mini-SDV. Fireguard was imaged from the development platform; the Firefox installer was not used. Once all of the source code and configuration changes (discussed below) were applied and tested on the development platform, the executable and configuration files were imaged on to a Mini-SDV Read-Only partition (labelled the ***Fireguard Browser Partition***). The ***Fireguard Browser Partition*** can be formatted as either a FAT32[17] or NTFS[18] file system.

A temporary Read-Write data partition, the ***Fireguard Data Partition***, is created on the Mini-SDV to store the user profile and configuration parameters created by Fireguard. A separate Read-Write partition, the ***User Data Partition***, is created for user data to be transported and/or processed. The ***User Data Partition*** provides the large secure PESE mass storage capability.

Figure 1 presents a conceptual model of the Mini-SDV configuration for Assured mode and can be summarised as:

- ***Fireguard Browser Partition:*** A Read-Only partition that protects and contains the Fireguard executable and configuration files.
- ***Fireguard Data Partition:*** A Read-Write partition that allows certain browser configuration parameters to be retained and any temporary data necessary for the correct browser operation. The Firefox user profile folder structure is contained in this partition.
- ***User Data Partition:*** A Read-Write partition for user data.

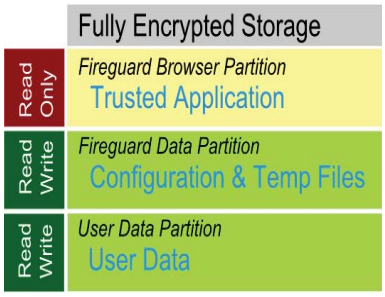

Figure 1: Conceptual Model of Mini-SDV Configuration

[17] FAT32 (File Allocation Table) is a relatively simple file system format used for storage devices. Since the introduction of NTFS, FAT is now predominately used on portable devices rather than PC HDDs.
[18] NTFS (New Technology File System) is a Microsoft proprietary file system format.

The ***Fireguard Data Partition*** contains a file structure that mirrors the Firefox file structure created for each user, i.e. a folder structure mimicking a Windows user home directory was created. This folder structure would contain the user profile and temporary data created during browser execution. Figure 2 presents an image of the ***Fireguard Data Partition*** hierarchical file structure. By mimicking the user directory structure the level of source code changes could be contained.

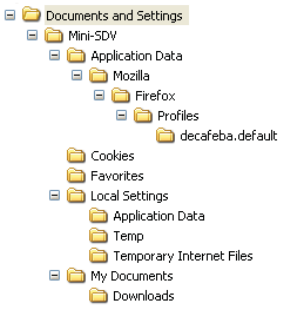

Figure 2: Structure of Fireguard Data Partition

Concept of Operation

In Assured mode, when the Mini-SDV is plugged into a USB port on a host PC executing the Windows OS an AutoRun[19] is detected and an authentication application is executed. Upon successful authentication Fireguard can be selected; upon selection it is uploaded into the host PC to enable secure remote access to the Internet. Access to data stored on the Mini-SDV is also available.

More specifically, once successfully authenticated the Mini-SDV authentication application accesses a custom MBR[20] on the Mini-SDV to retrieve the partition table.

The Windows OS assigns a drive letter for each Mini-SDV partition (defined in the partition table) containing FAT32 or NTFS file systems. Firefox determines its execution path from the registry value CSIDL_PROGRAM_FILE and the location of the Firefox user profile from the registry value CSIDL_APPDATA. However, as Fireguard does not use the registry values, because the Fireguard binary/configuration/interface files and user profile are stored in the ***Fireguard Browser Partition*** and the ***Fireguard Data Partition*** respectively, Fireguard needs to determine the drive letters for these partitions. An important design aspect of Fireguard is the ability to determine the drive letters allocated to the ***Fireguard Browser Partition*** and the ***Fireguard Data Partition***.

Upon logging-off the Mini-SDV, no further access to the device is possible without re-authentication.

Important Source Code Changes

Two key source code changes were made; removing dependencies on the PC file system and registry, and drive letter determination for the ***Fireguard Browser Partition*** and ***Fireguard Data Partition*** to ensure drive letter conflict does not occur.

Removing file system and registry dependencies: All CSIDL pointers to PC file system locations were changed to point to locations on the Mini-SDV ***Fireguard Data Partition***. A folder structure mimicking a Windows user home directory was created (see Figure 2). When the modified code was executed from the ***Fireguard Browser Partition***, it populated the empty Mozilla Firefox user profile folder on the Mini-SDV ***Fireguard Data Partition*** with its required configuration content.

Drive letter determination: As Fireguard does not use CSIDL values the following approach was implemented (by making source code changes) to determine the drive letters allocated to each partition:

- ***Fireguard Browser Partition***: When the Windows OS creates a process to execute an application it creates a parameter block which includes the application's full file path name, including the allocated drive letter. Therefore when Fireguard is executed it can determine the drive letter for the ***Fireguard Browser Partition*** by querying its process parameter block.
- ***Fireguard Data Partition***: The Mini-SDV is configured so that its first partition is always the ***Fireguard Browser Partition***, the second partition is the ***Fireguard Data Partition*** and the third partition the ***User Data Partition***. Logically, it could be assumed that the ***Fireguard Data Partition*** is allocated the next drive letter after the drive letter allocated to the ***Fireguard Browser Partition***. However, this may not be the case as Windows does not necessarily allocate drive letters sequentially. For example if a specific driver letter (e.g. H) is already allocated to a network share and if the drive

[19] AutoRun is a Windows OS feature that defines a set of actions to be performed when a drive/volume/partition is mounted; these actions are specified in the file autorun.inf held in the root partition of the drive.

[20] MBR (Master Boot Record) is the first sector of a partitioned data storage device that contains the partition table and may bootstrap an operating system if resident on the device.

letters C to F are currently allocated, then when the Mini-SDV is plugged into the PC the ***Fireguard Browser Partition*** will be allocated drive letter G. However, as drive letter H is already allocated another drive letter (e.g. I) will have been allocated to the ***Fireguard Data Partition*** and Fireguard needs to identify this drive letter to retrieve the user profile. To determine the drive letter allocated to the ***Fireguard Data Partition*** requires Fireguard to query the Windows OS structure containing partition table details (extracted from the Mini-SDV MBR) and the respective drive letter allocated to each partition.

- ***User Data Partition***: Like the ***Fireguard Data Partition*** the drive letter for the ***User Data Partition*** is determined by querying the Windows OS structure containing details on each physical partition and the respective allocated drive letter.

A number of more minor source code changes were also made which are identified below together with all the configuration changes made to protect against data leakage, cyber attack and to minimise the browser's forensic footprint.

Preventing Data Leakage

Changing the Firefox source code to ensure Fireguard writes browser created temporary data to the Mini-SDV ***Fireguard Data Partition*** is a key aspect in preventing data leakage. In addition, to specifically prevent data leakage, through the recovery of browser created data, the following features were changed, disabled or removed using "*lockpref*" functions placed in the *Fireguard.cfg* file to prevent saving:

- *Browser data in the form of history, cookies, saved passwords and form data*: This browser data is saved in the user profile on the Mini-SDV and therefore could be considered to be protected. However, this browser data becomes available to web sites when Fireguard is used to browse the sites. Therefore Fireguard has been configured to always delete history, cookies, saved passwords and form data when the browser exits for the following reasons:
    - Cookies can be used by web sites (hostile or otherwise) to track a user's Internet usage and behaviour, enabling an activity log to be formed.
    - Although passwords and form data are now securely retained on the Mini-SDV, such data can be automatically used by web sites to logon or pre-complete web forms. It was therefore considered good practice to ensure passwords and form data were not retained at the end of a browser session.
    - As the Fireguard history is stored on the Mini-SDV it presents neither a privacy threat nor vulnerability and so could be retained. However, as the Firefox private browsing mode does not save history records it was considered good practice for Fireguard to also not retain history records.
- *Web page cache*: The cache holds copies of recently retrieved content in the user profile for re-use if the same page is accessed again. The cache is retained between browser invocations and is therefore vulnerable to access by a hostile web site when the site is accessed by Fireguard (Soni, R. and B. Adhikari, 2009). In Fireguard cache retention is disabled.
- *Third-party cookies*[21]: Like browser cookies, third party cookies allow a user's Internet usage and behaviour to be tracked. By placing a third party cookie in the user profile an increased 'picture' can be formed of the user's site visits and interests (Jackson, C., A. Bortz, et al., 2006). Allowing a web site to write third party cookies to the user profile is prevented within Fireguard.
- *Importing another browser's settings*[22]: Firefox allows a user's settings and browser data from an alternative browser to be transferred, including the alternative browser's history, saved passwords, cookies and form data. As this data is not retained in Fireguard for the reasons given above Fireguard prevents the option of importing (into the user profile) an alternative browser's settings.
- *Default download area*: When a user wishes to download and save an object from the Internet and the user does not specify a location, the object is stored in a default folder, usually the *Desktop* or *My*

[21] Third party cookies are cookies set by one web site that can be read by another web site, i.e. a cookie is placed in the user profile of a web site the user may never have browsed. They are used by advertisers to track website visits.

[22] Firefox allows the browser data and settings established by a user in another browser (e.g. Microsoft IE) to be transferred into Firefox. Transferable data includes bookmarks, histories and saved passwords.

Documents folders. The feature was changed so that no default storage location is possible. The user is forced to save the downloaded object in a specific folder.

The above configuration changes (specified in the *Fireguard.cfg* file) coupled with both the protection of *Fireguard.cfg* in the ***Fireguard Data Partition*** and source code changes to remove file system and registry dependencies ensure Fireguard does not write data to the host PC HDD and reduces the possibility of data leakage.

Preventing Cyber Attack

The following features were disabled to prevent the exploitation of browser vulnerability and the introduction of malware from which a cyber attack could be launched:

- *Plug-ins*: To disable plug-ins both source code changes and "*lockpref*" settings in *Fireguard.cfg* were necessary. The source code function that attempts to determine the location of plug-in files was modified to immediately return an error code to indicate plug-ins are not available. A capability that allows the OS to be scanned for available downloaded plug-ins was disabled using a "*lockpref*" setting.
- *Extensions*: Like plug-ins both source code changes and "*lockpref*" entries in *Fireguard.cfg* were made to prevent the installation and use of extensions. There are 2 types of extensions; user only extensions which are held in the user profile and shared extensions held in a common folder with a registry entry identifying the path name of the common folder. *Lockpref* entries in the *Fireguard.cfg* file were used to prevent the loading of extensions. For completeness the source code was also changed to prevent the registry entry being created that contains the pathname of the folder containing shared extensions.
- *Software Updates*: There are four types of software updates; software patches for the browser, installation of new browser releases, updates for plug-ins and updates to the search engine list maintained by the browser. As the Fireguard binary, configuration and interface code are contained in the Read-Only ***Fireguard Browser Partition*** software updates are not possible. Also plug-ins are completely disabled so plug-in updates would not be possible. For completeness "*lockpref*" entries in *Fireguard.cfg* were set to ensure automatic updates for browser patches, new releases and the search engine list were not attempted.
- *Java*: To stop Java applets executing it is necessary to use a "*lockpref*" setting to prevent the browser detecting if the Java Runtime Environment was installed on the host PC. Also as plug-ins have been disabled it was not possible for the required Java plug-in to be installed to enable applets to execute.
- *Media Autoplay*: To prevent the automatic playing of audio and video files on a web site a "*lockpref*" entry in *Fireguard.cfg* was set.

With the exception of Javascript, disabling the above features prevents currently known sources of browser malware attacks. The placement of the *Fireguard.cfg* file in the Mini-SDV ***Fireguard Browser Partition*** (a Read-Only Partition) ensures the integrity of the file is preserved.

Minimising the Forensic Footprint

Many of the techniques used to prevent data leakage and cyber attack also reduce the forensic evidence of browser use. To reduce the forensic footprint for a browser involves ensuring that after the browser has executed there is no residual data on the PC HDD nor have any changes been made to the PC system configuration. A reduced forensic footprint was achieved in Fireguard by:

- *Ensuring Fireguard did not change the Windows registry settings*:
    - As discussed above, the Firefox installer makes a number of changes to the registry during installation. The installer was not used in the Fireguard development and all dependencies on registry values were changed in the source code. To avoid the use of the installer Fireguard was compiled and built on a development platform and then imaged onto a Mini-SDV Read-Only partition.
    - Plug-ins, extensions and software updates all make changes to the registry, particularly during installation. These three capabilities were disabled in Fireguard as described above.
- *Ensuring Fireguard left no residual data*: The removal/deletion of cookies, histories, saved passwords, form data and cached web pages from the ***Fireguard Data Partition*** after browser use, as described above, minimised forensic evidence. Any additional retained browser generated data was retained in the ***Fireguard Data Partition*** thereby leaving no forensic footprint.
- Not performing a default browser check: Upon executing Firefox the user is given the option to make Firefox the user's default browser. If selected as the default browser a registry entry is made. A "*lockpref*" entry in *Fireguard.cfg* was set to disable the default browser option in Fireguard.
- Firefox allows the Windows OS desktop background image to be selected and set by the user, a feature that changes both the registry and performs a write of the image to the PC file system. In Fireguard setting the desktop background image is prevented by a source code change to a compile time option which has the effect of disabling the capability.

The above changes virtually eliminated the forensic evidence of browser use. However, the Windows OS creates potentially forensically recoverable evidence of Fireguard use through details contained in the OS swap space/page management system (pagefile.sys). Also if Fireguard is used to download an object the OS creates a registry most recently used (MRU) list entry to indicate an object has been saved.

User Interface Changes

The removal or disabling of Firefox features to create (the secure browser) Fireguard necessitated the modification of the Firefox user interface to remove menu options for features that were no longer available. Without the interface changes removed/disabled features would appear as "greyed-out" options on the browser drop-down menus. The following changes were made to the user interface by making changes to the XUL files held in jar archives in the "*chrome*" folder installed in the ***Fireguard Browser Partition***:

- Removal of the "Add-ons[23]" menu as both plug-ins and extensions are disabled.
- Removal of the Update Tab as software updates are disabled.
- Removal of the Privacy Tab as all history and cookies are cleared when the browser exits.
- Removal of the Password pane in the Security options as all passwords are cleared when the browser exits.

Retention of Built-in Security Capabilities

The Firefox phishing and malware detection capabilities are retained in Fireguard. The phishing and malware database is stored per-user in the respective user profile. For Fireguard, this means the database will be stored in the user profile in the ***Fireguard Data Partition***. The Fireguard phishing and malware protection is either on or off. As phishing and malware identifiers are like virus signatures the list of sites infected by phishing attacks and malware is highly dynamic. The browser checks for infected web sites each time the browser is started; the browser may also check periodically during a session for additional updates.

[23] Mozilla uses the term Add-ons to generically describe both plug-ins and extensions.

The site identity security capability is also retained. However, the master password capability is removed as Fireguard does not retain any password details and therefore a master password to protect other passwords is not necessary. The browser also had all default bookmarks removed and the home page set to a blank page to provide a clean setup.

It was decided to retain the bookmark feature so that favourite web sites could be accessed without entering the URL each time access to the site is required. Bookmarks provide no security or privacy vulnerability.

FIREGUARD TESTING

A comprehensive test plan was created that covered:

- Functionality testing of the core features and user interface of Fireguard. The testing included error trapping to confirm the browser could recover from errors.
- Vulnerability testing to ensure the browser could not be re-configured to include functionality (e.g. plug-ins, extensions, etc) that could make it vulnerable to malware. The testing also included exception, logical and operational testing.
- Stress testing Fireguard under heavy workloads and over prolonged periods.
- Sociability testing of Fireguard in an environment with other executing applications.
- Compatibility testing of the browser across the range of Windows OS' including versions with different service packs.

The above comprehensive testing confirmed that Fireguard operated correctly as a portable secure browser with vulnerable functionality removed.

The final area of testing to be performed was to confirm that Fireguard had a minimal forensic footprint. Digital forensic techniques can be categorised as dead forensic[24] and live forensic[25] techniques. Dead forensic analysis of the PCC HDD to determine if Fireguard had written data to the HDD would not provide a meaningful result as both the OS and other executing applications will also write to the HDD. Therefore the following live forensic tests were performed to verify if evidence remained on a PC HDD after a browser session:

- Monitoring file creation: The "Process Explorer" program (Windows Sysinternals, 2010) was used to monitor the executing Fireguard. This utility lists all resources, including file handles[26] that a running process is using. The list of file handles was monitored during a set of browser functionality tests and no unexpected files were read or written on the host PC HDD. An inspection was also made of locations that the standard Firefox browser would use and no files were found.
- Registry: An export of the registry to text was performed before and after a series of browser functionality tests. The resulting exported text files were compared using the program "Beyond Compare" (Scooter Software, 2010). This program highlights differences between two versions of the same file. Careful examination of the two versions revealed only changes to most recently used values (MRU). The MRU values were changed by the OS when Fireguard was used to download an object from the Internet.

CONCLUSION

Limitations

Whilst Fireguard provides a secure portable browser, resistant against cyber attack and data loss, with a minimal forensic footprint it does have the following limitations:

- *Javascript*: By allowing Javascript to remain enabled there exists a mechanism by which malware can be introduced into the browser.

[24] Dead forensic analysis is performed on data at rest, i.e. data acquired from a PC HDD after the PC has been powered down.

[25] Live forensic analysis is performed on a powered PC usually under the control of the PC's operating system, i.e. data is acquired from the PC whilst the PC is still executing - the data may be acquired from the PC's memory, HDD, communication interfaces or internal intelligent devices (e.g. graphics processor).

[26] A file handle is a descriptor indicating the status of an open file.

- *Clickjacking Attacks*: Clickjacking attacks occur because standard HTML features coupled with Javascript allow malware to exploit user actions often unbeknown to the user. Nothing has been designed into Fireguard to detect or prevent clickjacking.
- *Monitor Attacks*[27]: The removal of features like plug-ins, extensions, software updates and Java reduce the possibility of malware attacks launched from within the browser, however it does not prevent monitoring attacks from either software legitimately installed on the PC or from rootkits that may have evaded any anti-virus and anti-spyware installed on the host PC. Preventing monitor attacks is considered outside the scope of the Fireguard project.
- *Immediately Executable Malware*: Fireguard does not prevent malware that is covertly presented to the user and is executed immediately (without requiring the user to reboot the PC) by deceiving the user to invoke its execution, e.g. the user opens an email attachment (received through webmail accessed by Fireguard) that is actually malware disguised as an attachment of interest to the user; the malware is able to execute upon the user opening the attachment and compromise Fireguard. Disabled features like plug-ins, extensions and Java may minimise or eliminate the effect of the malware. Despite such malware possibly being able to compromise Fireguard the malware cannot become embedded in the browser's binary because of the Mini-SDV Read-Only partition protection, therefore at worst the compromise occurs for one browser session.
- *Page file/swap space data*: Windows implements virtual memory management based upon a page swapping system. All executing processes have data held in virtual memory. When a process terminates, data may remain in swapped out pages. It is possible that sensitive data viewed or processed by the browser may be retained in virtual memory and recovered by an unauthorised user (Ruff, N., 2008).
- *Hardware Dependent*: Fireguard is dependent on the functionality provided by the Mini-SDV, in particular the partition Read-Only access control feature to protect the integrity of the Fireguard binary. Fireguard is constructed using both amended Firefox software and the Mini-SDV capabilities to provide a secure browser; therefore porting the functionality to another secure PESE may not be desirable or even possible. To achieve the equivalent protection against cyber attack, data leakage and reduced forensic footprint through the use of an alternative secure PESE hardware device would require the device to support multiple partitions with differentiated access rights that work seamlessly with the range of Windows OS'.
- *Distributing a new Fireguard Patch or Release*: Although the integrity of the Fireguard binary and configuration files are protected by a Mini-SDV Read-Only partition this integrity protection mechanism prevents the automated distribution of patches and releases. Currently, the only method of updating the Fireguard binary is for the binary to be re-imaged onto the Mini-SDV ***Fireguard Browser Partition*** by an administrator with the appropriate permissions. However, although patch management can not be automated it can be argued that the application of security patches for Fireguard is not as imperative as it is for standard Firefox as both the Mini-SDV Read-Only partition provides protection against malware including zero day attacks[28] and the disabling of features like extensions, Java, etc reduces the available vulnerabilities which could be exploited for an attack.
- *Source Code Changes:* Satisfying the requirements for a secure browser could not be achieved through changes to configuration files alone, changes had to be made to source code. If Fireguard is to be upgraded to a newer release of Firefox it will be necessary to perform a review of the source code to determine if the code structure and layout has changed and identify if the code changes performed to create Fireguard can still be applied without new analysis.
- *Lacks Full Firefox Functionality:* To produce Fireguard, functionally rich features like plug-ins and extensions have been removed. Lack of this functionality obviously limits how Fireguard can be used. However, the objective of the Fireguard project was to produce a secure portable browser that can be used by remote and mobile workers to access corporate systems, corporate webmail and specific corporate web based applications, not to provide a highly sophisticated browsing tool.

[27] A monitor attack occurs when malware is able to spawn a number of processes that monitor one another to enable resurrection of a hostile malware process if one or more of the original hostile processes are discovered, thus enabling the hostile activity to continue.

[28] A zero-day attack is malware that tries to exploit software vulnerabilities before neither the software vendor is aware and able to provide a patch nor an antivirus product vendor has identified the malware signature to enable its detection.

Strengths

In summary a secure highly portable browser has been developed based on limited changes to an open source browser. Fireguard provides strong protection against cyber attack, data loss and leaves virtually no evidence on a host PC that it has executed. Fireguard is considered to be a secure tool for remote and mobile workers who need secure Internet access from a PC for which no level of trust can be assumed.

The strengths of Fireguard are demonstrated in Table 1 by showing conformance to the five requirements established for the project.

Requirement	Conformance
To prevent malware and hence cyber attack it must be executable from a Mini-SDV Read-Only partition.	The Fireguard binary was imaged and installed in a Mini-SDV Read-Only partition from which it successfully executed.
To prevent malware and hence cyber attack it must prevent plug-ins, extensions and software updates.	Plug-ins, extensions and software updates were disabled. Also the media autoplay and Java features that can be susceptible to malware were disabled.
To prevent forensic discovery it must not update the Windows registry.	Source code and configuration changes were made so that no Windows registry changes occurred.
To prevent data leakage and forensic discovery it must store configuration data and temporary files in a nominated storage area on the Mini-SDV.	All retained browser data is stored in the ***Fireguard Data Partition*** on the Mini-SDV.
To prevent data leakage and forensic discovery it must not write to the Host PC HDD, with any created temporary data deleted when the browser terminates.	Fireguard configuration settings ensure browser data is not retained. Testing of Fireguard confirmed the browser did not write data to the host PC HDD.

Table 1: Fireguard Conformance to Requirements

The Fireguard project has produced a secure PESE trusted application which satisfies the requirements specified for it.

Further work

Future work could include:

- *Javascript*: Fireguard has been successfully developed and tested. An obvious change could be to disable Javascript with the predominate use of the browser to access corporate web applications that do not utilise Javascript. Alternatively Javascript can remain enabled and the Firefox extension 'NoScript' could be integrated into Fireguard and used to allow Javascript to execute only from certain allowed sites.
- *Developing a capability to detect clickjacking sites*: Research has been conducted into mechanisms to detect and prevent clickjacking (Balduzzi, 2010; Rydstedt, G., E. Bursztein, 2010). A capability within Fireguard that can detect and prevent clickjacking would enhance security. Also if the 'NoScript' extension is integrated into Fireguard clickjacking attacks can be reduced.
- *Transitioning to Firefox version 4*: The changes made to Firefox to create Fireguard have been fully documented for Firefox version 3.6, therefore (in theory) transitioning the source code changes and configuration settings to version 4 should not be difficult. However, open source projects often restructure source code and functionality so a future activity would be to identify how rapidly a Fireguard release based on Firefox version 4 could be achieved.
- *Developing an automated patch management and release tool*: A Mini-SDV capability to enable remote and mobile workers to securely and automatically perform patch management and/or upgrade to a new release of Fireguard will be important if the Mini-SDV with Fireguard is used in large scale and dispersed deployments.

Although the identified future work would enhance the usability and security of Fireguard the project has achieved its objectives and produced a secure PESE trusted application in the form of a secure portable browser suitable for deployment to remote and mobile workers to enable secure Internet access from an available and potentially untrusted PC.

REFERENCES

Adobe (2010). "Flash Player." Retrieved September, 2010, from http://www.adobe.com/products/flashplayer/.

Balduzzi, M., M. Egele, et al. (2010). A solution for the automated detection of clickjacking attacks. ASIACCS '10 Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ACM.

Barth, A., A. Felt, et al. (2010). Protecting browsers from extension vulnerabilities, Citeseer.

Cnet News (2009). "Using software updates to spread malware." Retrieved October, 2010, from http://news.cnet.com/8301-27080_3-10301485-245.html.

Help Net Security (2009). "Zero-day vulnerabilities in Firefox extensions discovered." Retrieved October, 2010, from http://www.net-security.org/secworld.php?id=8527.

IronKey (2010). "IronKey Technology." Retrieved September, 2010, from https://www.ironkey.com/hardware-encryption.

Jackson, C., A. Bortz, et al. (2006). Protecting browser state from web privacy attacks. WWW '06 Proceedings of the 15th international conference on World Wide Web ACM.

James, P. (2008). Secure Portable Execution Environments: A Review of Available Technologies. 6th Australian Information Security Conference, Perth.

Lemos, R. (2010). "Could USB Flash Drives Be Your Enterprise's Weakest Link?". Retrieved September, 2010, from http://www.darkreading.com/vulnerability-management/167901026/security/storage-security/227200081/index.html.

McAfee (2010). McAfee Threats Report: Second Quarter 2010.

MDN (2010). "Introduction to XBL." Retrieved September, 2010, from https://developer.mozilla.org/en/XUL_Tutorial/Introduction_to_XBL.

MDN (2010). "JavaScript Guide." Retrieved September, 2010, from https://developer.mozilla.org/en/JavaScript/Guide.

MDN (2010). "XUL Tutorial." Retrieved September, 2010, from https://developer.mozilla.org/En/XUL_Tutorial.

Moscaritolo, A. (2008) US military bans USB thumb drives. SC Magazine

Mozilla (2010). "Mozilla Firefox."

mozillaZine (2010). "Locking preferences." Retrieved October, 2010, from http://kb.mozillazine.org/Locking_preferences.

NetMarketShare (2010). "Top Browser Share Trend." Retrieved September, 2010, from http://www.netmarketshare.com/browser-market-share.aspx?qprid=1.

NTI (2004). "Classified Data Identification & Data Elimination Guidelines." Retrieved August 2010, 2010, from http://www.forensics-intl.com/riskscan.html.

PHPforms (2010). "Introduction to CSS." Retrieved September, 2010, from http://phpforms.net/tutorial/tutorial.html#cat_180.

Ponemon, L. (2009). Trends in Insider Compliance with Data Security Policies - Employees Evade and Ignore Security Policies, Ponemon Institute.

Reynaud-Plantey, D. (2005). "New threats of Java viruses." Journal in Computer Virology 1(1-2): 11.

Ruff, N. (2008). "Windows memory forensics." Journal in Computer Virology 4(2): 83-100.

Rydstedt, G., E. Bursztein, et al. (2010). Busting frame busting: A study of clickjacking vulnerabilities on popular sites. W2SP 2010: Web 2.0 Security and Privacy 2010, Oakland, California.

Scooter Software (2010). “Beyond Compare.” Retrieved October 2010, from http://www.scootersoftware.com/moreinfo.php.

Secure Systems (2010). "Mini Silicon Data Vault." Retrieved October, 2010, from http://www.securesystems.com.au/index.php?option=com_content&view=article&id=6&Itemid=11.

Smart Link Corporation (2010). "ImTranlator." Retrieved September, 2010, from https://addons.mozilla.org/en-US/firefox/addon/2257/.

Soni, R. and B. Adhikari (2009). Browser Security, Carleton University: 46.

Symantec (2010). Symantec Global Internet Security Threat Report Trends for 2009. Volume XV.

TrueCrypt-Foundation (2010). "TrueCrypt Users Guide." Retrieved September 2010, from http://www.truecrypt.org/docs/.

Windows SysInternals (2010). “Process Explorer.” Retrieved September 2010, from http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx

Zalewski M (2009). Browser Security Handbook, part 2, Google Inc.