2010

A Security Framework for Managing the Host-based Collection of End- User Information

Peter Clutterbuck
University of Queensland, p.clutterbuck@business.uq.edu.au

Terry Rowlands
University of Queensland

Murray Stubbs
University of Queensland

A Security Framework for Managing the Host-based Collection of End-User Information

Peter Clutterbuck
Terry Rowlands
Murray Stubbs
School of Business
University of Queensland
Brisbane, Australia
Email: p.clutterbuck@business.uq.edu.au

Abstract

There is an increasing prevalence of Web software that collects end-user information and transmits it to a remote server destination. This information collecting software paradigm spans many scenarios – from fully legitimate software updates, to identifying user surfing habits (i.e. adware), to collecting personal user-information (i.e. spyware). The design science research within this paper describes an information security management framework that extends existing code-signing conventions via an extended X.509.3 digital certificate specifying: (1) whether the signed software transmits any information from the end-user machine to any remote destination, and if so (2) a concise summary of the type of this information and the remote destination address(es). This extended code-signing is then supported by the end-user's operating system authentication of each outgoing Web transmission from each specific host-based software application. The framework facilitates improved end-user management and regulatory governance of all Web communication streams emanating from the user host computer.

Keywords

Information, security, management, privacy, risk.

INTRODUCTION

Internet technologies have revolutionized user-information collection across computer networks. The paradigm of user-information collecting software spans many functional scenarios ranging from the clearly legitimate (e.g. application software updates), to the ethically concerning (e.g. identifying user Web surfing habits), through to the illicit collection of personal user-information. The terms "*spyware*" and "*adware*" both describe software that collects user information – the key difference centres upon the type of user information collected (Gordon 2005). Spyware may log user key strokes (Hu et al 2005), or capture user email, instant messaging, passwords, and credit card information (Gordon 2005) – it is designed to illicitly collect and distribute user information (Cohen 2003). Adware is considered a benign subset of spyware – delivering targeted pop-up advertisements to the user's computer based on the analysis of that user's Web surfing habits. Adware may pose the security risk of privacy violation. Spyware poses the security risk of identity and information theft (Gibson 2005).

Spyware has proliferated with the increasing popularity of Web technologies (Fang et al 2005). Cosgrove (2003) described how over 7000 different forms of spyware had been estimated to be running on U.S. based corporate and personal computers. The U.S. marketing sector reported in Economist (2004) that software from the top three spyware firms in the U.S. was installed on approximately 100 million PCs. A survey of home PC users conducted by America Online found spyware present on 80% of home PCs, whilst approximately 90% of those users with spyware were unaware of the spyware's presence or purpose (Roberts 2004). An audit of over 4.6 million U.S. home PCs (Earthlink 2005) found 116.5 million instances of spyware infestation – with an average of 25 infestations per PC. Spyware proliferation across the corporate sector was indicated in (FBI 2005) where 79.5% of surveyed businesses reported a spyware security incident. The SANS Institute in FBI (2005) reported a 183% increase in websites that harboured spyware. This situation was very much negatively impacting users' confidence in online security, and therefore users' confidence in online business.

Spyware control has been problematic primarily because, unlike the universally distained criminal act of virus infection (Hu et al 2005), spyware distribution in general is a commercial venture and is difficult to distinguish from the many other forms of legitimate information-collecting software. Indeed the U.S. Federal Trade Commission stated in Federal Trade Commission (2005). "*It is difficult to define spyware with precision. The working definition proposed ... was software that aids in gathering information about a person or organization*

without their knowledge and which may send such information to another entity without the consumer's consent, or asserts control over a computer without the consumer's knowledge." The literature review conducted with this research confirms that U.S. legislators have repeatedly failed to pass bills in the Congress because of the risk of restricting existing or impeding future Internet business strategies. Spyware distributing businesses have routinely threatened legal action against the authors of any tools that label spyware for what it truly is (Edelman 2006). Sipior (2005) reported that the market for anti-spyware software was still small (between USD10-15 million in sales) compared to the anti-virus industry (USD2.2 billion). Even the general public seems to be confused with respect to spyware differentiation and attitude. Clyman (2004) reported that "*When most people hear the word 'spyware', they think in terms of malevolent software ...*" However Stafford (2004) reported that many users accepted spyware as a fair price to pay for getting free software and were very much reluctant to implement blocking/scanning procedures that risked upsetting this '*win-win*' arrangement.

The research question underpinning this paper is how to security manage user-information collecting software to prevent the unauthorized disclosure (i.e. breach confidentiality) of user information. This paper describes design science research that has produced a security management framework for all user-information collecting/reporting Web based software. The security framework, at its core, extends the existing code-signing public key infrastructure (PKI) to require each user-information collecting application to nominate at installation what user information is collected and transmitted to remote server destinations. Additionally the framework then requires each user-information collecting application to authenticate (with the host operating system) each outgoing Web session initiated by that software application. This framework facilitates the user-management of all Web communication streams emanating from the host, and this in turn supports the identification and control of software that engages in the deceptive, misleading, and fraudulent practices already proscribed in existing technology-focused legislation. The framework is economically feasible because it does not add financial costs to the production of software – no additional certificates or software signing is required. The framework is as secure from attack as any other component of a trusted computing base (i.e. operating system). The remaining sections of this paper will unfold as follows. Section two will describe the research methodology. Section three will describe the threats model and risk analysis (derived for the existing security controls relating to user-information collecting software) that fundamentally influenced the design of our new security framework. Section four will present the logical description of our security framework. Section five will conclude the paper.

RESEARCH METHODOLOGY

This paper describes design science IS research, the dominating IS research paradigm in the German-speaking countries and also heavily used in the Nordic countries, the Netherlands, Italy, and France (Winter, 2008). While behavioural IS research aims at the exploration and validation of generic *cause-effect* relations, IS design science research aims at 'utility', i.e. at the construction and evaluation of generic *means-ends* relations. That is, design science IS research aims at the construction of 'better' IS-related problem solutions (Winter, 2008).

Design science IS research and behavioural IS research both aim for what has been called the 'Pasteur quadrant'– combining a high standard of relevance with the highest standards of rigour (Winter, 2007). The relevance of the research in this paper has already been described in the opening section (i.e. the increasing popularity and high risk profile of user-information gathering software). The rigour of the research in this paper is more challenging to demonstrate simply because the rigour of generalized design science IS research "is less well defined and less commonly accepted than its behavioural counterpart." (Winter, 2008, p. 470). This lack of definition is perhaps best illustrated by the lack of a commonly accepted reference process model for design science IS research – with proposals including "build-evaluate-theorize-justify' (March and Smith, 1995) and 'develop/build – justify/evaluate' (Hevner et al., 2004). Consequently, this paper must demonstrate research rigour by an appropriate discussion of the research process model at its core. This research process model is based on the fundamental IS security process – risk management (Oppliger, 2007).

According to Oppliger (2007) and Stoneburner et al. (2002) risk management fundamentally drives IS security via a structured process of identifying, controlling, and eliminating or minimizing uncertain events that may affect system resources/assets. Risk management requires a clear understanding of the targeted resources/assets, the threats to those resources/assets, and what attack strategies are likely to be used in relation to those resources/assets (Oppliger, 2007). Consequently risk management provides an ideal template upon which to base the research process model used within this research. Figure 1 logically presents our five sequential stage research process model, adapted from Oppliger (2007) and Stoneburner et al. (2002) and now discussed.

- Security Policy (stage 1): This is a strategic, highly abstracted 2-tuple of: (1) the system resource(s) of value, and (2) the level of security that must be maintained with respect to this resource. For this research this 2-tuple is: {user-information; prevention of the unauthorized disclosure, i.e. confidentiality}. Security policy (stage 1) represents the research question underpinning this paper.

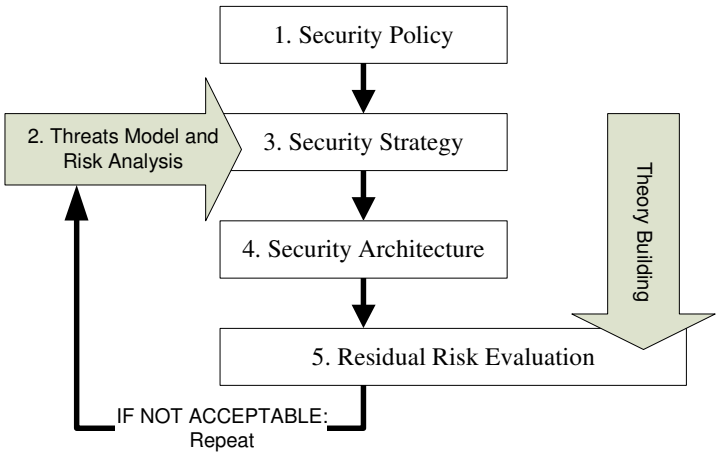

Figure 1: Research Process Model

- Threat Model and Risk Analysis (stage 2): Threat model describes the likely operational strategies of all possible threat sources. Risk analysis quantifies the success likelihood of an attack strategy against the identified system resource(s) within the context of deployed security strategies and security architectures. Success likelihood within this research is the well accepted taxonomy used in (Stoneburner et al. 2002), and described in Table 1.

Table 1. Success Likelihood of Attack Strategy

Likelihood Level	Definition
High	The threat-source is highly motivated and sufficiently capable, and controls to prevent the vulnerability from being exercised are ineffective.
Medium	The threat-source is motivated and capable, but controls are in place that may impede successful exercise of the vulnerability.
Low	The threat-source lacks motivation or capability, or controls are in place to prevent, or at least significantly impede, the vulnerability from being exercised.

- Security Strategy (stage 3): The strategic description of the security controls that will be deployed in offsetting (mitigating) the threats identified at stage 2. Security strategies within this research are the well accepted criteria used in (Stoneburner et al. 2002), and described in Table 2.

Table 2. Security Strategies

Security Strategy	Definition
Confidentiality	The protection of information from unauthorized disclosure.
Integrity	The protection of information from improper modification. The operational correctness of software processes.
Authentication	The verification of identity.

- Security Architecture (stage 4): The implementation of the operational security controls to achieve the strategies described in stage 3. Operational implementation possibilities are far too numerous to completely list – however each architecture implements directly one (or more) of the security strategies of stage 3 – for example, *encryption controls* implements *confidentiality*, *digital signatures* implement *authentication*, and *hash digests* implement *integrity*.
- Residual Risk Evaluation (stage 5): Residual risk is the risk remaining after the derivation of a security strategy and the application of security architecture. The calculation of residual risk then poses a security

gap assessment – does the residual risk level match (or better) the risk level that is desired by the security policy? If not, steps 2 to 5 inclusive are repeated until the desired risk level is reached.

The research process model of Figure 1 has been used in this research to calculate the residual risk level posed by user-information collecting software within the context of existing security strategies and architectures. The results of this exercise will now be presented in the next section *Threats Model and Risk Analysis*. The theory building component of Figure 1 (stages 2 to 5 inclusive) has been used in this research to derive a new security framework to further reduce the residual risk level posed by user-information collecting software. This new security framework will be logically described in the section *Framework*.

THREATS MODEL AND RISK ANALYSIS

The threats model constructed within this research has focused on analysing and understanding those data transmissions emanating from host machines running user-information collection software. Consequently our threats model will be discussed under the minor heading of *Data Transmission Analysis*. The risk analysis within this research will be discussed under the minor headings: *Success Likelihood of Illicit Information Collection*.

Data Transmission Analysis

Data transmissions (frequently referred to as *extrusions* in the research literature) from the target software group are clearly considered, within the research security community, to be the major security threat posed by spyware (remembering the two major concerns contained within the Federal Trade Commission (2005) definition: "*The working definition proposed ... was software that aids in gathering information about a person or organization without their knowledge and which may send such information to another entity without the consumer's consent, or asserts control over a computer without the consumer's knowledge.*") Our threats model required the construction of purpose built software residing on a host machine and collecting/analysing a large set of (1) spyware request transmissions/extrusions and (2) the request transmissions of legitimate data collection programs.

The analysis and understanding of the collected data extrusions were then performed with the goal of identifying any patterns that may differentiate spyware extrusions from the extrusions of legitimate data collection software. The analysis criteria (Table 3) comprised four {*remote server address consistency, request frequency, request size, request periodicity*} developed for the Web Tap utility (Cui et al. 2005) together with two developed within this research {*user-agent name and sending protocol, request encryption*}. The anatomy of the software constructed for this purpose comprised the three logical auditing components specified in Bishop (2003): *a logging engine*; *an analysis engine*; and *a notifier engine*. The logging engine essentially comprises a *WinPcap* software module that facilitates the packet imaging/copying of outgoing/egress communications. This approach is very similar to that used in established software utilities such as *tcpdump* and *ethereal/wireshark*. The notifier engine is essentially a reporting module. The analysis engine assesses all transmissions on the basis of: user-agent-name (i.e. application name), remote server address details, request periodicity, request frequency, and request encryption. This software was deployed for a continuous 30 day period on a test host machine – operated on a dedicated network behind a packet filtering firewall – and loaded with a selection of 40 spyware/adware agents and 10 professional applications utilizing data-reporting agents. The analysis of the resulting transmissions produced the results described in Table 3.

Table 3: Outgoing Transmission Analysis (Spyware and Legitimate Programs)

Analysis Criteria	Analysis Summary
User-Agent Name and Sending Protocol	All transmission used HTTP/TCP Therefore User-Agent field appears in HTTP requests and is used to name the client initiating the HTTP session. Web browsers have distinctive User-Agent fields identifying the browser type/version (e.g., Mozilla/4.0) and client operating system (e.g., Windows NT 5.1; SV1). 71% of the studied spyware transmissions displayed a User-Agent field quite distinct from the User-Agent field known to be running on the host machine. Most spyware displayed the User-Agent name of mainstream browser agents. Only 6% of studied valid transmissions contained a User-Agent name not related to the application.
Remote server address consistency	All legitimate transmissions and 38% of spyware transmissions contacted a consistent, single remote server. 62% of spyware transmissions contacted a set of remote servers (up to a maximum of 8).

Request periodicity	The HTTP content and capture/copied time are logged for each studied spyware communication. Approximately 58% of the studied spyware transmissions and 77% of legitimate transmissions revealed a request periodicity. That is, the transmitting software was initiating HTTP connections at defined time intervals. The actual time intervals defining this periodicity varied greatly.
Request frequency	This criterion is closely linked with request periodicity. All studied spyware initiated multiple HTTP sessions with remote servers within a 12 hour period. The average frequency of contact was 7 sessions within the 12 hour period.
Request size	Most studied spyware packets were not large. Of the total sample only 58 requests exceeded 2KB, whilst only 7 requests exceeded 5KB.
Request encryption	A measure of the entropy/frequency distribution of the data contained within requests. Complicated by routine character encoding schemes. Encryption considered probable in approximately 7% of spyware transmissions and zero percent of legitimate transmissions.

Analysis of the data summarized in Table 3 reveals several probable indicators for identifying spyware request transmissions. This is consistent with the results presented and used (Cui et al 2005) to filter probable spyware extrusions (i.e. the *Web Tap* utility). However the analysis within this research suggested these probabilities were not strong and furthermore the operational characteristics producing these probabilities could easily be changed by spyware authors to avoid detection. The most compelling observation arising from the analysis was that all transmissions (spyware and legitimate) used the protocol suite HTTP/TCP. In relation to spyware, this endorses the view of Borders et al. (2004): "*Often, the only two ways out of a network are through a mail server and through a proxy (web) server. Since e-mail is often more closely logged and filtered, the hacker may find outbound HTTP transactions to the best avenue for communication with a compromised workstation*." In relation to legitimate software transmissions, it endorses the view that the vast majority of networks run the Web (i.e. allowing Web traffic to pass through firewalls etc), and therefore HTTP/TCP is the protocol suite of choice for Internet communicating software. This poses *the* major problem in detecting spyware transmissions: the transmissions blend in anonymously with all legitimate HTTP traffic initiated by user Web activity and other legitimate Web communicating applications communicating with remote servers. Often it is only possible to avoid this issue by not running HTTP and this is not workable in most situations. HTTP should not be shut down – a better strategy is to remove the anonymity. This is a key finding to take forward in developing a new framework.

Success Likelihood of Illicit Information Collections

The success likelihood of illicit information collections directly depends on the effectiveness of existing security architectures in managing user-information collecting software. The literature survey conducted within this research suggests security architectures comprise three logical groupings: (1) *filtering software,* (2) *legislative controls,* and (3) *code-signing*.

Filtering software controls comprise utilities such as firewalls, proxies, and desk-top or server based filtering applications. In the main, filtering software controls have largely followed the existing mature anti-virus model (Gibson 2005) which comprises the following categories: Filtering software controls exhibit one or more of the following function capabilities:

- *Media filtering*: The automatic inspection of all incoming and outgoing communications for associated media type (e.g. text, audio, video) and the identification of malicious content via code signature scanning and anti-tampering integrity check.
- *Function scanning*: The application of heuristics that attempt to establish the functional capabilities of the content.
- *Attack method detection*: The application of heuristics that attempt to identify code that displays known attack methods.

The effectiveness of filtering software is reflected in Weis (2005): "*So far, none of the anti-spyware packages have proven to be 100% effective*". This is largely because filtering software (whether virus or spyware oriented) can only detect that software for which signature patterns have already been identified and extracted. This is always a 'catch-up' challenge in the anti-virus industry – an industry where virus classification is very clear cut. It is even more problematic in the anti-spyware industry where the classification of spyware is quite fuzzy. In these circumstances, the success likelihood of illicit information collection within a context of filtering software architecture must be rated (by Table 1) as MEDIUM.

Legislative control proposals for the specific management of spyware have been quite frequent in the U.S. – but not in Europe or Australia. Table 4 below has been produced within this research and summaries the spyware-specific proposed legislation (i.e. bills with the U.S. Congress).

Table 4. Proposed U.S. Spyware Bills

Bill Name	History/Result
Spyware Control and Privacy Protection Act of 2000	Introduced S.3180 6th Oct 2000. Never voted on by Senate.
Computer Software Privacy and Control Act of 2004	Introduced H. R.4255 30th April 2004. Never voted on by Senate
Internet Spyware (I-SPY) Prevention Act of 2004/2005/2007	Introduced H.R. 4661 23rd June 2004. Re-introduced H.R. 744 10th Feb. 2005. Re-introduced H.R. 1525 14th Mar. 2007. Each bill passed Reps. No bill ever voted on by Senate.
Software Principles Yielding Better Levels of Consumer Knowledge Act (or SPY BLOCK Act) of 2004 / 2005	Introduced S.2145 27th Feb. 2004. Re-introduced S.687 20th Mar. 2005. Neither bill voted on by Senate.
Enhanced Consumer Protection Against Spyware Act of 2005	Introduced S.1004 11th May 2005. Never voted on by Senate.
Securely Protect Yourself Against Cyber Trespass Act (SPY Act) of 2003/2005/2007	Introduced H.R.2929 25th May 2003. Re-introduced H.R.29 2nd Jan. 2005. Re-introduced H.R.964 8th Feb 2007. Each bill passed Reps. No bill ever voted on by Senate.
Undertaking Spam, Spyware, and Fraud Enforcement with Enforcers beyond Borders Act of 2006	Introduced S.1608 29th Jul. 2005. Passed by Senate and Reps. Became law on 22nd Dec. 2006.
Counter Spy Act (2007)	Introduced S.1625. Never voted on by Senate.
Informed P2P User Act (2008/2009)	Introduced H.R.7176 27th Sept. 2008 Re-introduced H.R.5th Mar. 2009. First bill never voted on by Reps. Second bill passed Reps. Yet to be voted on by Senate (111th Congress concludes end of 2010).

Analysis of the legislative controls specifically proposed to manage spyware (from 2000 to date) reveals that only one bill (of the nine proposed) has become law, and that bill focused specifically (and only) on extending the Federal Trade Commission (FTC) Act to (1) include a focus on foreign commerce "*unfair or deceptive acts or practices*" and (2) authorized the FTC to liaise with foreign law enforcement agencies. Only three bills were considered by the full Congress. One bill is still live and has until December 2010) to pass – although this bill (Informed P2P User Act) is very narrow in its focus. Indeed most bills were never reported out of committee and never voted upon. The Congress records associated with these unsuccessful bills all contain two consistent criticisms:

- The specific definitions within the bills (e.g. of *computer software* and *information collection programs*) would extend far beyond spyware and cover generic Internet and Web technologies. Consequently the bills risked a blanket prohibition of existing and emerging technology (with some exceptions attempted within certain bills). This approach would restrict existing – and impede future – business practices.
- The FTC (in hearings associated with several bills) testified that it would welcome the additional power to seek civil penalties from those who break the law (i.e. the FTC Act). The FTC testified that its authority to prosecute '*deceptive, misleading, and fraudulent*' activity was adequate from an enforcement viewpoint. It conceded, however, that it had prosecuted only 7 cases over a 10 year period.

In these circumstances, the success likelihood of illicit information collection within a context of specific anti-spyware legislation must be rated (by Table 1) as HIGH.

Code signing is conceptually described (Rubin et al. 1998) as client management of a list of entities that the client trusts. When mobile code is received by the client (e.g. via the Web), the client verifies that the received

code is signed by an entity on the list. If so, the code is run. Code signing (Mansfield-Devine 2009) is operationally described as a simple process in which the software vendor creates a hash (via an algorithm such as SHA or MD5) of the code file for distribution. The hash is then signed using the private key of a public-private key pair (i.e. via public key cryptography). The vendor's public key is itself signed by a Certificate Authority (CA), an entity we all trust within an overall Public Key Infrastructure (PKI). The vendor's public key is placed by the CA within a strongly authenticated digital certificate. The vendor then distributes the code, together with the signed hash of the code file and the CA-authenticated digital certificate in an area of the code file known as the signature block. The end-user will then verify the vendor's digital certificate, which in turn allows the end-user to verify the signed hash of the code file – and thereby authenticating that the code file has not been altered since it was prepared and signed by the vendor. Code signing is described as the basis for a stronger trust relationship between vendors and end-users (Mansfield-Devine 2009). Code signing is the basis for Microsoft's Authenticode framework. Code signing adoption is gaining in momentum and it is widely seen within the industry and by users as enhancing trust and progressing the goal of digital rights management. Criticisms have surfaced, however, in two areas (Mansfield-Devine 2009): (1) lack of consistency in standards across Certificate Authorities, and (2) the lack of specificity resulting from the issue of a single code signing certificate to a specific vendor (i.e. "*Giving someone a code signing certificate allows them to digitally sign anything that they have"*). The criteria for commercial certification is also very generalized and quite weak – comprising (1) proof of identity; (2) a pledge by the applicant that the distributed code will not contain viruses; and (3) a level of financial standing indicated by a Dun & Bradstreet rating of corporate financial stability.

In these circumstances, the success likelihood of illicit information collection within a context of code signing must be rated (by Table 1) as HIGH.

FRAMEWORK

The three principal factors emerging from the previous section were: (1) Spyware data streams (i.e. extrusions) are anonymously mixed with all other legitimate Web traffic and are very difficult to consistently pick from legitimate Web software extrusions; (2) Spyware activity must be recognized as deceptive, misleading, or fraudulent because of what the software does – not the technology it uses; and (3) The existing code signing model does not provide any real control or specificity as to the information type and information legitimacy transmitted by signed software. The overall success likelihood of illicit information collection within this overall security architecture must be rated (by Table 1) as MEDIUM. An improved security management framework is now abstractly described in Figure 2 and logically discussed in the following paragraphs. It is suggested that this framework reduces the success likelihood of illicit information collection down to LOW.

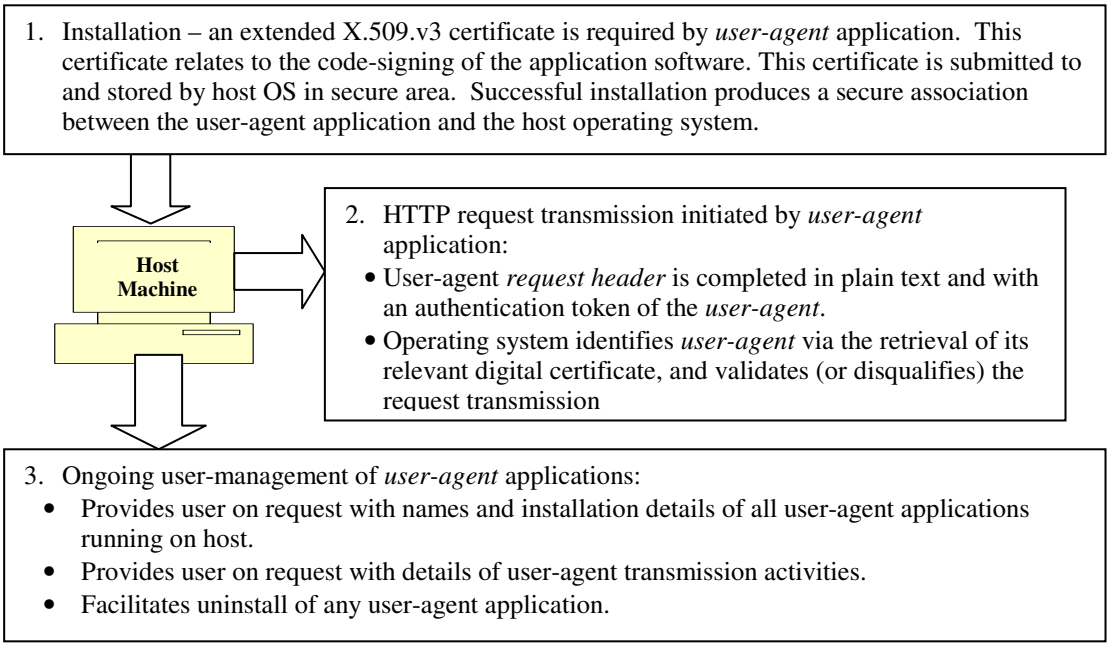

Figure 2. Security Framework

In overview, the new framework comprises a Public Key Infrastructure for all Web communicating software. A Public Key Infrastructure (PKI) is described (Whitman et al 2009) as: "*an integrated system of software, encryption methodologies, protocols, legal agreements, and third-party services that enables users to communicate securely*." The digital certificate is a central component within a PKI. The conclusion reached in

this research is that the relative anonymity applicable to all outgoing Web traffic from a host machine must be removed and the user provided with the clear capacity to more effectively identify and manage that outgoing Web traffic. An effective way to remove anonymity within a system is to name and authenticate each entity within that system. Digital certificates and an overall PKI facilitate naming and authentication. In this sense it is suggested that this security framework builds upon the concept of code signing widely used for the trusted distribution of executable content (e.g. Microsoft's Authenticode). The security framework is now described with respect to Figure 2.

The security framework allows three primary activities: *Installation* (a one-time activity), *Request Transmission* (a repeated activity), and *User-Management* (a repeated activity). Within this framework, a Web communicating application is called a *user-agent* application.

Installation: The user-agent application must possess one X.509.v3 digital certificate (code signing certificate) with obligatory X.509.v3 extensions as described in Table 5.

Table 5. New Framework Certificate Extensions

Extension Name	Extension Definition
Application_Name	The unique name of the specific software application – within the name space of the software vendor.
Information_Type	Defined as USER, or APPLICATION, or OTHER. The classification of information collected and transmitted by software application.
Destination_URL	The IP address to where the *Information_Type* is sent by *Application_Name*.

This extended code-signing certificate has been issued to – and owned by - the publisher of the software (i.e. the software vendor/distributor). The extensions in Table 5 are obligatory and convey additional subject identification information and policy information allowed for in Housley et al. (1999).

- The certificate is presented to the host operating system at installation (i.e. the installation package contains the user-agent code set, the signed hash of the code set, and the extended code signing certificate). This is very much as per existing code signing schemes (e.g. Microsoft's Authenticode). The conventional '*Subject_Name*' field of this code signing certificate may still list the corporate owner of the certificate. The extended field '*Application_Name*' will name the specific user-agent application. This is contrasted with a conventional code signing certificate which authenticates and names the software publisher only. Following successful installation, the host operating system has created a secure association with the installed software application – we shall call this (*secure_association*) $_{user\_agent}$. This secure association is temporally persistent and will be used to authenticate the software application on each occasion it initiates an outwardly directed Web communication session.

Request-transmission: This activity comprises three sequential actions: (1) the request by the user-agent application of a symmetric session key from the host operating system, (2) preparation of the Web request by the user-agent application and (3) the authentication and transmission/non-transmission of this Web request by the host operating system.

- The user-agent application requests a symmetric session key from the host operating system. This request is decided by the host operating system on the basis of the (*secure_association*) $_{user\_agent}$ created at installation between the user-agent and the host operating system. If granted, this symmetric session key will be known as (*sym_session_key*) $_{user\_agent}$. This key will exist and be used by the user-agent during the life of the forthcoming HTTP communication session. This key will be added to, and stored (for the duration of this Web session) within the applicable (*secure_association*) $_{user\_agent.}$ The session key will be destroy and not reused again when this HTTP session is terminated.

- The user-agent application prepares a Web request for transmission by the TCP within the host operating system. The HTTP *request-header* fields are constructed as outlined RFC 2616 (HTTP/1.1) – with one variation (the variation retains full compliance with RFC 2616): the *user-agent request-header* field. Within this security framework, the *user-agent request-header* field is used to name and authenticate each user-agent request transmission. Within RFC 2616 the user-agent request header field is described as: "*contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid user agent limitations. User agents SHOULD include this field with requests.*" The augmented Backus-Naur Form (BNF) used in RFC 2616, describes the *user-agent request-header* field as: *User-Agent = "User-Agent" ":"*(product | comment). An example would*

be: User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows XP; SV1) That is, the field can contain multiple product tokens and comments identifying the agent and any sub products. In this framework, the *user-agent request-header* field will contain the agent details as currently required – concatenated with the following:

[*Application_Name+timestamp*]*(sym_session_key)* $_{user\_agent}$.

That is, the *Application_Name* from the extended X.509v3 code-signing certificate - concatenated with the *user-agent* private key encrypted (*Application_Name* and timestamp). The use of *sym_session_key)* $_{user\_agent}$ will demonstrates the authenticity of this Web transmitting application to the host operating system.

- The host operating system receives the Web request for transmission. The operating system checks the *user-agent request-header* field and establishes the *Application_Name* of this *user-agent* application. The operating system, using this unencrypted *Application_Name* data (an implementation of the 'shared-secret' security principle), retrieves the appropriate (*sym_session_key*) $_{user\_agent}$ *user-agent* digital certificate, and the public key (i.e. $U\text{-}A_{Pub}$) within this certificate. The operating system then decrypts (*Application_Name+timestamp).* The decrypted *Application_Name* is compared with the name as stated in clear text by the user-agent. The decrypted timestamp is validated against operating system clock time (to offset 'replay' attacks). If all checks are passed, the authentication token (i.e. [*Application_Name timestamp*]*(sym_session_key)* $_{user\_agent}$ is removed by the operating system from the user-agent request-header field to ensure this data does not subsequently confuse remote servers. The request is transmitted and details logged by the operating system. If all checks are not passed, the host user is notified (via a security alert) and asked for adjudication. The details are then logged by the operating system.

User-management: This activity is initiated at the request of the host machine user. User-management provides the following reports:

- Summarised historical listing of the transmissions made by all user-agent applications. This listing provides the following information set: {*process name, date installed, total number of transmissions, remote addresses contacted, type of information transmitted (i.e. USER, APPLICATION, or OTHER), and total bandwidth used*}.

CONCLUSIONS

The goal of this research was to contribute to the effective security management of user-information collecting software (including spyware and adware). Our research produced three important factors: (1) spyware extrusions are relatively anonymous within the overall set of Web transmissions leaving any host machine, (2) anti-spyware controls must help identify deceptive, misleading, and fraudulent behaviour, and (3) the existing code signing model could be extended to provide greater specificity and therefore security utility.

The security framework described in this paper proposes a new and more comprehensive direction for software code signing. The framework is economically feasible because it does not add financial costs to the production of software. The framework, which is as secure from attack as any other component of a trusted computing base (i.e. operating system), directly aims to identify and authenticate Web request transmissions before they exit a host machine. This removes the current anonymity under which spyware proliferates, and also moves some distance to discouraging the authoring and distribution of suspect software. The security framework also provides a user with a much improved capacity to review the historical records of information transmissions from the host machine. These reasons strongly indicate that the success likelihood of illicit information collection under this new framework must be rated (by Table 1) as LOW.

REFERENCES

Bishop, M. (2003) *Computer Security Art and Science*. Addison-Wesley. 689-708

Borders, K. and Prakash, A. *Web Tap: Detecting Covert Web Traffic*. Proceedings of the 11th ACM Conference on Computer and Communications Security. ACM Press, pp 110-120

Whitman, Michael. E. Mattord, Herbert. J. (2009) Principles of Information Security (Third Edition). Thomson Course Technology. 2009. pp375.

Clyman, J. (2004) Antispyware: Adware and spyware are a growing nuisance and threat. PC Magazine, 23 (13), p82.

Cohen, J. E. (2003) *DRM and Privacy*. Communications of the ACM. Volume 46, Number 4, pp 46-49.

Cosgrove, F. (2003) Is someone using spyware to monitor how your employees are using their computers? Computer Weekly, 11 November Edition. 38.

Cui, Weidong. Katz, Randy. H. Tan, Wai-tian. (2005) Design and Implementation of an Extrusion-based Break-In Detector for Personal Computers. Proceedings of the 21st Annual Computer Security Applications Conference. 2005.

Earthlink (2005) *Earthlink Spy Audit*. http://www.earthlink.net/spyaudit/press. Visited October 26, 2009.

Economist, 2004. *A Hidden Menace*. June 5, 2004 edition, pp 61-66.

Edelman, Benjamin. 2006 *"Spyware": Research, Testing, Legislation, and Suits*. (Online) http://www.benedelman.org/spyware (last updated July 18, 2006) Visited 19th Feb. 2010.

Farag, Neveen. and Fitzgerald, Kristina. 2005 *The Deceptive Behaviors that Offend Us Most about Spyware*. Communications of the ACM. Volume 48, Number 8, pp 55-60.

FBI. 2005. *2005 FBI Computer Crime Survey*. (Online) Available at: http://www.fbi.gov/publications/ccs2005.pdf

Federal Trade Commission. 2005. *Monitoring Software on Your PC: Spyware, Adware and Other Software*. http://www.ftc.gov/os/2005/03/050307spywarerpt.pdf, p. 20.

Gibson, Steve. 2005 *Spyware Was Inevitable*. Communications of the ACM, (48:8), pp 337-39.

Gordon, Sarah. 2005 *Exploring Spyware and Adware Risk Assessment*. (Online) Proceedings of the 2005 EICAR Conference. http:www.eicar.org. Visited June 15, 2009.

Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Design science in information systems research," *MIS Quarterly* (28:1), pp 75-105.

Housley, R., Ford, W., Polk, W., and Sold, D. 1999. RFC 2459 Internet X.509v3 Public Key Infrastructure Certificate and CRL Profile. Available at: http://www.ietf.org

Hu, Qing. and Tamara, Dinev. 2005 "Is Spyware and Internet Nuisance or Public Menace?" *Communications of the ACM*. (48:8), pp. 61-66

IETF. (2008) Internet X.509 Public Key Infrastructure Certificate, RFC 5280. <http://tools.ietf.org/html/rfc5280>

Mansfield-Devine, Steve. (2009) "A matter of trust." *Network Security*, June 2009, pp. 7 – 9.

March, S. T. and Smith, G. F. 1995. "Design and natural science research on information technology," *Decision Support Systems* (15:4), pp 251-266.

Oppliger, R. 2007. "IT Security: In Search of the Holy Grail." *Communications of the ACM* (50:2), February, pp 96-98.

Roberts, P. 2004 *Your PC May Be Less Secure Than You Think*. IDG News Service, October 25, http://www.pcworld.com/news/article/0,aid,118311,00.asp. Visited October 15, 2009.

Rubin, Aviel. D. Geer, Daniel. E. Jr. 1998 Mobile Code Security. IEEE Internet. November-December 1998. pp. 30 – 34.

SANS Institute 2007 *One in three websites are infected*. (Online News Report) Available at: http://www.sans.org. Visited February 20, 2010.

Sipior, J. C., Ward, B. T., and Roselli, G. R. 2005 *A United States Perspective on the Ethical and Legal Issues of Spyware*. Proceedings of the 7th International Conference on Electronic Commerce (ICEC'05). ACM Press. 738-743

Stafford, T. F. and Urbaczewski, A. 2004 *Spyware: The Ghost in the Machine*. Communications AIS, pp. 291-306.

Stoneburner, G., Goguen, A., and Feringa, A. 2002. *Risk Management Guide for Information Technology Systems*, Special Publication 800-30, National Institute of Science and Technology. Available at: http://www.nist.org

Weis, A. 2005 *Spyware Be Gone*. netWorker, Volume 9, Issue 1 (March). ACM Press.

Winter, R. 2008. "Design science research in Europe," *European Journal of Information Systems* (17), pp. 470-475.

Winter, R. 2007. "Relevance and rigour – what are acceptable standards and how are they influenced? (with Contributions of Baskerville R, Frank U, Heinzl A, Hevner A, Venable J) *Wirtschaftsinformatik* (49:5), pp 280-287.

COPYRIGHT