9 July 2011

An Extended Public Key Infrastructure Framework For Host-Based Information Security Management

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

Terry Rowlands
University of Queensland, t.rowlands@business.uq.edu.au

ISBN: [978-1-86435-644-1]; Full paper

AN EXTENDED PUBLIC KEY INFRASTRUCTURE FRAMEWORK FOR HOST-BASED INFORMATION SECURITY MANAGEMENT

Peter Clutterbuck, UQ Business School, University of Queensland, Brisbane, Australia, p.clutterbuck@business.uq.edu.au

Terry Rowlands, UQ Business School, University of Queensland, Brisbane, Australia, t.rowlands@business.uq.edu.au

Abstract

Web software that is designed and deployed to collect end-user information and transmit it to a remote server destination is proliferating. This overall software paradigm spans many scenarios – from fully legitimate software updating and remote management, to profiling user surfing habits (i.e. adware), to illicitly collecting personal user-information (i.e. spyware). The research within this paper is based upon two research paradigms – a qualitative approach to explore the security challenge, followed by a design science approach to evolve an enhanced control solution. This solution comprises an information security management framework that extends existing code-signing conventions via an extended X.509 Version 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 certificate is then used by the end-user's operating system as the basis for a persistent security association which authenticates each outgoing Web transmission from each specific host-based software application. The security framework facilitates improved end-user management and regulatory governance of all Web communicated information transmitted from the user host computer.

Keywords: Information, security, management, privacy.

1 INTRODUCTION

Web protocols 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 updating and remote management), 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 CSI (2009) where respondents reported "the largest increases (in security technology) were in anti-spyware software and encryption of data". The SANS Institute (2007) 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.*" 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 improve the security management of user-information collecting software and prevent unauthorized disclosures (i.e. improve the control of confidentiality at the user's end). This research question has been pursued via a two-stage research methodology. The first stage comprises a qualitative investigation of the research question via detailed interviews with a group of information security professionals. This qualitative evaluation data were then analysed via a general inductive approach (Thomas 2006) – with the results comprising a conceptual mapping of the major underlying categories evident in the raw data. This conceptual mapping from the first stage then feeds into the second stage - a design science methodology to produce a logically designed security framework that delivers the objectives of the conceptual

mapping. This logical security framework, at its core, extends the existing code-signing public key infrastructure (PKI) to require each user-information collecting application to notify the user of what information is collected and transmitted to which specific remote server destinations. This extended code-signing certificate is then processed by a user operating system to create and enforce a persistent security association with the information collecting software, and thereby authenticate each outgoing Web transmission from each specific host-based software application. The security association ensures that user-information collecting software actually performs to those transmission specifications originally notified within the code-signing certificate (i.e. at time of installation). This overall framework facilitates the improved user-management of all Web-based information 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 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 two-part research methodology (firstly qualitative, secondly design science). Section three will present in the overall results from this research – again in two parts (qualitative and design science). Section four will conclude the paper.

2 RESEARCH METHODOLOGY

The research question underpinning this paper is: "*How to improve the security management of user-information collecting software and prevent unauthorized disclosures?*" In an IS security context, this research question centres upon the research strategy of confidentiality as defined in Stoneburner et al. (2002). The pursuit of this research question – within a confidentiality context - has been pursued via a two-stage research methodology as outlined in Figure 1. This section will now describe these two stages: firstly the qualitative '*front end*', followed by the design science '*build and evaluate*' back end.

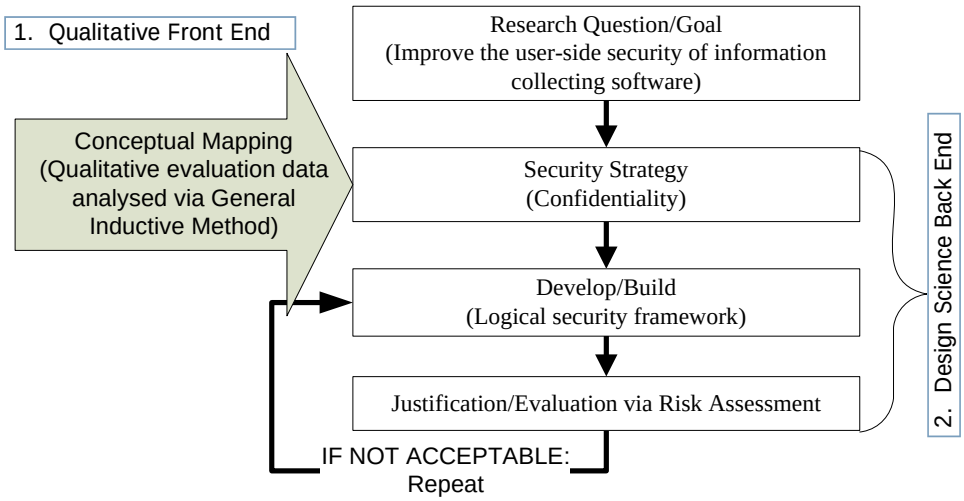

Figure 1. Research Process Model

2.1 Stage 1: Qualitative Font End

Stage 1 of this research project comprises a qualitative investigation of the research question via detailed interviews with a group of five information security professionals, all responsible for an IS security program within an Internet-enabled business environment. The qualitative evaluation data from these interviews were then analysed to produce a conceptual mapping of the major underlying categories evident in the raw data. This conceptual mapping from the first stage was then input into the second stage - a design science methodology (discussed in Section 2.2) to produce a logically designed security framework that delivers the objectives of the conceptual mapping.

Stage 1 data was obtained via a total of 22 hours of interviews in which all participants were requested to discuss two topics. The first topic centred upon each interviewee discussing the strengths and weaknesses of existing security controls (both at their individual business level and an overall sector level) in relation to the research goal. These interviews were loosely shaped by the generalised IS security control structure outlined in Oppliger (2007) – that is, IS security controls comprise *policy/legislation*, *education/awareness*, and *technology*. The second interview topic centred upon all interviewees offering an assessment of the risk level posed by client-side, user-information collecting software. To ensure assessment consistency, interviewees were requested to frame their assessment as per the well accepted taxonomy used in (Stoneburner et al. 2002), and described in Table 1.

Risk 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.

Table 1. Risk level taxonomy

The qualitative evaluation data from these interviews were analysed via a general inductive approach (Thomas 2006). The general inductive approach is a "*a straightforward set of procedures to follow without having to learn the underlying philosophy and technical language associated with many qualitative analysis approaches*" (Thomas p. 238 2006) – for example, grounded theory (Strauss & Corbin 1998), phenomenology (van Manen 1990), or discourse analysis (Potter & Wetherell 1994). The inductive approach is a systematic procedure for analysing qualitative data in which the analysis is likely to be guided by specific evaluation objectives (i.e. an open research question). Inductive analysis uses detailed readings of raw data to derive concepts or themes through interpretations made from the raw data by the researcher(s). This understanding is consistent with Strauss and Corbin's (1998, p. 12) description: "*The researcher begins with an area of study and allows the theory to emerge from the data*". This understanding is also consistent with the general patterns of qualitative data analysis described in Miles & Huberman (1994).

Coding consistency within the inductive analysis of this research was achieved via the coding checks proposed in Thomas (2006). Independent parallel coding was consistently utilised – a researcher completing the raw data analysis and proposing a set of categories to form preliminary findings – the second researcher independently analysing the same raw data and proposing a second set of categories – both category sets are then reviewed and reconciled by both researchers. Stakeholder checks – involving all five interviewees - were also utilised to enhance the credibility of findings (both at the interview transcript level and also at the coded category level).

2.2 Stage 2: Design Science Back End

Stage 2 of this project – as outlined in Figure 1 - receives the output results of stage 1 and utilises 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). Put simply, the maturity, profile and

research community acceptance of the statistical methods within behavioural IS research are considerably greater than those research methods used within design science.

To meet this challenge, this research has blended the proposed "*build-evaluate-theorize-justify*" (March & Smith 1995) and "*develop/build – justify/evaluate*" (Hevner et al. 2004) design science methods with the generalised risk assessment approach described in Oppliger (2007). According to Oppliger (2007) and Stoneburner et al. (2002) risk assessment fundamentally drives IS security via a structured process of identifying, controlling, and eliminating or minimizing uncertain events that may affect system resources/assets. Risk assessment 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 assessment provides an ideal template upon which to *justify* and *evaluate* the logical framework as it emerges from the "*develop/build"* front-end' within our overall design science research stage. This is shown in Figure 1 as the "*Justification/Evaluation*" activity that is iteratively performed as the logical security framework evolved within stage 2. This risk assessment of the "*Justification/Evaluation*" was performed on the logical security framework by all five interviewees. The assessment of residual risk then allows a security gap assessment – does the residual risk level match (or better) the risk level that exists within the existing security control environment?

3 RESEARCH RESULTS

Research results will be presented for the two stages outlined in Section 2. The conceptual mapping results from the general inductive analysis of interview data will be presented firstly. The logical security framework from the design science stage will then be described. The section concludes with a summarised evaluation of the logical security framework as provided by all original interview participants.

3.1 Conceptual Mapping Results

The inductive analysis initially identified 26 categories across the completed interview data set. Further analysis to reduce overlap and redundancy amongst these categories reduced the identified set to 10. Final analysis to derive the most important categories produced a set of 4. This final set of categories is shown in Table 1 (with categories listed in terms of their prominence within the interview data set). Table 1 also shows that the interviewees rated the existing spyware threat as MEDIUM+ to HIGH.

Interview Question	Categories Identified
Discuss the strengths and weaknesses of existing security controls (both at your individual business level and an overall sector level) in relation all mobile information collecting/reporting software (an industry 'check-list' would be policy/legislation, education/awareness, and technology.	Anonymity of code and transmitted information
	Greater trust in mobile code
	Flexible and future focused solutions
	Enhanced user-management of spyware
What is your rating of the existing spyware threat?	MEDIUM+ to HIGH

Table 2. Final categories from interview data

This section will now discuss these categories in the sequence reported in Table 2.

Anonymity of code and transmitted information: This was clearly the most identified category within the interview data set. All interviewees contributed strong statements on this category – examples are as follows:

- "All data streams leaving the box are impossible to identify."
- "Mobile code easily moves across the network perimeter and onto hosts without any identification."
- "That's the problem, information is transmitted from a host and we cannot tell what it is or anything about why it was collected."
- "Extrusions all look the same, some are appropriate, some are not, we cannot tell them apart."

Interviewees spoke of the great difficulty in identifying different code streams as they crossed from the network perimeter and entered host machines. Interviewees spoke of the even greater challenge of identifying information streams (i.e. extrusions) as they left the host machines bound for remote destinations. Interviews spoke of the limitations of filtering software controls such as firewalls, proxies, and desk-top or server based filtering applications. The interviewee comments concur with the effectiveness of filtering software as 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 (Gibson 2005). 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 (FTC 2005) and very much contested by many vendors (Edelman 2006).

The interviewee comments on extrusions are also well supported in the literature. At a regulatory level the Federal Trade Commission (2005) defined spyware as: "*…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.*" At a technical control level, the Web Tap utility (Cui et al. 2005) has been used to collect and analyse extrusions initiated by specific spyware software over a several week period. The Web Tap utility was designed to run on a proxy Web server and analysed the extrusions along several dimensions, including *remote server address, request frequency, request size,* and *request periodicity*. The results reported by Cui et al. (2005) indicate some discernable patterns in the spyware extrusions. However these patterns were not strong and furthermore the operational characteristics producing these patterns could easily be changed by spyware authors to avoid detection. One of the most compelling observations arising from the Web Tap analysis was that all spyware extrusions used the protocol suite HTTP/TCP. 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.

Greater trust in mobile code: This was a very strongly identified category within the interview data set. Examples of interview statements are as follows:

- "One certificate for one vendor – too much emphasis on the intellectual copyright of the vendor."
- "Processing of certificates is very generalised…too cliché…certificates mean little to the user."
- "The certificate trust model has lost momentum…needs to evolve."

The interviewees were all very keen to improve user-level trust in mobile code. Interviewees endorsed the global code-signing PKI - but spoke of its limitations and its vendor weighted bias. Interviewees were optimistic about the potential of the global code-signing PKI – but clearly stated that the framework had not evolved sufficiently with the very rapid technology advances in Web architecture and software engineering. Code signing is described in the literature (Rubin et al. 1998) as client management of a list of entities that the client trusts – and therefore code signing could be the basis of a stronger trust relationship between vendors and end-users (Mansfield-Devine 2009). Code signing is the basis for Microsoft's Authenticode framework. Code-signing 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.

Flexible and future focused solutions: All interviewees (as business/industry professionals) expressed strong views as to the need for technology innovation to thrive now and into the future. Examples of interview statements are as follows:

- "The Web is just starting to be realized…cannot let security become stifling."
- "We do not want legislation to limit innovation...we want legislation and regulators to pursue fraudulent activity."
- "Legislators need to allow the industry to move forward and deter fraudulent activity – not technology innovation."

All interviewees stated clearly that anti-spyware legislation was not a viable solution because of the risk posed to technology and business innovation and Web evolution in general. Two interviewees were very familiar with U.S. efforts to legislatively proscribe spyware and spoke strongly against this possible direction. The literature describes the level of circumspection that the U.S. Congress has displayed in this area this century. Legislative control proposals for the specific management of spyware have been quite frequent in the U.S. – but not in Europe or Australia. Table 3 below has been produced within this research and summaries the spyware-specific proposed legislation (i.e. bills with the U.S. Congress).

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).

Table 3. Proposed U.S. Spyware Bills

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. Most bills were never reported out of committee and never voted upon. The Congress records associated with these unsuccessful bills all contain a consistent criticism that the definitions (within the bills) 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).

Enhanced user-management of spyware: This was the final major category to emerge from the data. The category may appear somewhat surprising considering all five interviewees are very experienced network and security managers. The interviewees all suggested the provision to end-users of greater

event management and decision making support in relation to information security. Example comments are as follows:

- "End-users are far more technology literate and can make very good IS decisions if adequately resourced and supported."
- "In an environment of so much host processing of information, end-users must be allowed to exert greater information security control."
- "We cannot just prescriptively set user policy…we need to have clear policy and then provide users with the environment on their desk top to allow adoption and implementation of the policy."

3.2 Logical Security Framework

The four principal categories emerging from the qualitative research of Section 3.1 are:

- Anonymity of spyware – anonymity relating to both the mobile code and the operationally transmitted data streams i.e. extrusions. The accepted security control solution for mitigating anonymity risks is authentication which is defined as the validation of identity (Whitman et al. 2009). Authentication may be provided via a number of solution strategies – what an entity knows (e.g. a password), what an entity has/possesses (e.g. a digital certificate), or what an entity produces (e.g. signature pattern recognition). Whitman et al. (2009) suggests the digital certificate – issued by an appropriately recognized Certificate Authority (CA) – is increasingly seen as the most scalable and effective security control for providing authentication.
- Greater user trust in mobile code. Trust is the central deliverable to all users within a Public Key Infrastructure (PKI) – defined in 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*".
- Flexible and future-focused solutions to facilitate emerging business directions. Global Public Key Infrastructures (PKI) are universally recognized by both the business and most regulatory communities (Whitman 2009).
- Enhanced end-user management of spyware. All mainstream operating systems and mainstream Web browsers support the distribution, processing, and revocation of digital certificates within a Public Key Infrastructure (PKI).

These four principal categories suggest a control solution comprising a digital certificate that identifies *all* client-side software by providing an appropriate description of the egressing data streams transmitted by that software (if any). This control solution significantly builds upon the current code-signing concept currently used for the trusted distribution of executable content (e.g. Microsoft's Authenticode). This control solution is now described as per the following set of four logical components: *Certification*, *Certificate installation*, *Information transmission*, and *End-user-management*. Within this control solution a software application is referred to as a *signed_application*.

3.2.1 Certificate Issue

Commercial code-signing certification by a Certificate Authority requires: (1) proof of applicant identity; (2) a pledge by the applicant that the distributed code will not contain viruses; and (3) a level of applicant financial standing as indicated by a Dun & Bradstreet rating of corporate financial stability. This security framework evolved from this research requires the extension of these requirements to obtain the following additional information (Table 4.) from each code signing certificate applicant. The information in Table 4 is now included within an X.509 version 3 code-signing certificate by the Certificate Authority. The X.509 specification is contained in IETF (2008). This specification states that "*Certificates may be used in a wide range of applications and environments covering a broad spectrum of interoperability goals and a broader spectrum of operational and assurance requirements*" (IETF p. 16, 2008).

The specification (IETF, 2008) uses the Abstract Syntax Notation Version 1 (ASN. 1) to provide the following top level (i.e. abstracted) certificate description:

```
Certificate     ::=     SEQUENCE  {  -- An ASN type or class to describe multiplicity.
        tbsCertificate          TBSCertificate,  -- the plain text content of the certificate.
        signatureAlgorithm      AlgorithmIdentifier,   -- algorithm used in digitally signing cert.
```

```
    signatureValue          BIT STRING  }       -- the actual digital signature data.
```

Within the recursive nature of the ASN, the certificate component TBSCertificate is then described as follows (with an ASN comment abstracting away all certificate components not used within this research framework):

```
TBSCertificate  ::=     SEQUENCE  {
        -- nine certificate fields (of the specified ten) are not shown here.
        extensions      [3]     EXPLICIT Extensions OPTIONAL
                                -- If present, version of certificate must be v3.
```

Extension Name	Extension Definition
application_name (a single entity per certificate – cannot be NULL)	The name of the specific *signed_application*. This name must be unique (and remain unique) within the application name space of the software vendor.
information_type (a possibly repeated entity – that is, one to a maximum of four occurrences of this entity – cannot be NULL)	Defined by the set {USER, APPLICATION, OTHER, NONE}. The classification of information collected and transmitted by software application. USER information covers all data that relates to the end-user (e.g. browser history or personal details). APPLICATION information refers to all data that relates to the configuration or operation of the *signed_application*. OTHER refers to all cases where information is collected/transmitted and not classified as USER or APPLICATION. NONE refers to those cases where the *signed_application* will not collect/transmit any information.
destination_url (a possibly repeated entity – that is, one or more – cannot be NULL)	Comprises either: (1) The DNS name set to where the *Information_Type* is sent by *signed_application* OR (2) NONE (where the *signed_application* will not collect/transmit any *Information_Type*.

Table 4. Extended applicant information requirements for certificate issue

The IETF (2008) provides for the presence of an arbitrary number of extensions (a SEQUENCE of SIZE (1..MAX)) within any version 3 certificate. These extensions "*provide methods for associating additional attributes with users or public keys and for managing relationships between CAs*" (IETF p. 26, 2008). Each extension includes an OBJECT IDENTIFIER (or OID) for unique naming within the appropriate domain of interest. Each extension occurrence is fully defined by the following ASN.1 structure:

```
Extension       ::=     SEQUENCE    {
        extnID          OBJECT IDENTIFIER  -- a unique name within the community of interest.
        critical        BOOLEAN DEFAULT FALSE  -- an indicator for processing the certificate.
        extnValue       OCTET STRING – the encoding of the value of the extension.
```

The IETF (2008) currently defines 15 Extension OBJECT IDENIFIERs – with all OIDs created and remaining unique within the appropriate domain namespace (usually global). This research framework proposes a 16th Extension to include the information of Table 4 within the code-signing certificate. Consequently, the ASN.1 description of the extended code-signing certificate within this research framework would be:

```
extnID          ::=     {  id-extended-code-signing  }
critical    :   ::=     TRUE  -- cannot be false.
extnValue       ::=     SEQUENCE  {
                        application_name        [0]     Name
                        information_type        ::=     SEQUENCE    {
                                type_name  [1]    String  }   -- {USER, APPLICATION, OTHER, NONE}
                        destination_url         ::=     SEQUENCE    {
                                destination_name  [2]   String  }
                        }
```

3.2.2 Certificate Installation

The user-agent application must possess one X.509.version 3 digital certificate (code signing certificate) with obligatory, enhanced X.509.v3 extensions as described in Section 3.2.1. This extended code-signing certificate has been issued to – and owned by - the publisher of the software (i.e. the software vendor/distributor). 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*' field of this code signing certificate will still list the corporate owner of the certificate. The extended field '*application_name*' will name the specific *signed_application*. This is contrasted with a conventional code signing certificate which authenticates and names the software publisher only (i.e. only contains the '*subject*' field). Following successful installation, the host operating system has created a secure association with the installed *signed_application* – we shall call this (*secure_association*) $_{signed\_application}$. This secure association is temporally persistent for the installation lifetime and will be used to authenticate the software application on each occasion it initiates an outwardly directed Web communication session. Consequently the processing logic employed by the operating system at installation is described as follows (exceptions are not shown):

(1) receive installation packet -- a structure comprising {code set, signed hash, certificate}
(2) validate code signing certificate -- validation via signing CA root certificate
(3) create and store (*secure_association*) $_{signed\_application}$

3.2.3 Information Transmission

All network communication is managed exclusively by the modern operating system. Therefore all communicating applications must request and receive the appropriate support from the host operating system before network communication (local host or remote host) can proceed. The processing logic employed by the operating system is as follows (exceptions are not shown explicitly)

(1) Receive communication request
(2) Retrieve the appropriate (*secure_association*) $_{signed\_application}$
(3) Validate communication request against (*secure_association*) $_{signed\_application}$
-- that is, validate request against *type_name* and *destination_name*.
(4) Approve communication request
(5) Write communication request to appropriate system log

3.2.4 End-User Management

The following two activity types are defined within end-user management: {*Routine*, *Exception*}.

Routine end-user management would provide to the end-user on request a summarised historical listing of the transmissions made by all *signed-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, OTHER, NONE), and total bandwidth used*}. Exception end-user management centres upon notifications provided by the operating system to the user as exceptions arise within the two certificate processing scenarios outlined in Section. 3.2.2 (Certificate Installation) and Section 3.2.3 (Information Transmission). The following exceptions are defined by the framework for installation and information transmission:

- The received code (*file_name*) will not be installed: no attached code signing certificate.
- The received code (*file_name*) will not be installed: cannot validate attached code signing certificate.
- The *signed_application* has been denied communication approval – code signing certificate declares NONE for *information_type* and *destination_url*.
- The *signed_application* has been denied communication approval – code signing certificate *destination_url(s)* does not match the remote URL nominated in current communication request.

3.3 Security Framework Evaluation by Interviewees

Figure 1 described the overall research methodology underpinning this project – including how the final security framework would be assessed by the original interviewees via a risk assessment process (using Table 1. as the risk assessment criteria). This activity unfolded as a focus group process in which all interviewees were given a presentation of the logical security framework, including screen simulations of the end-user management notifications of Section 3.2.4. The major themes from this assessment process are as follows:

Simplicity: Interviewees rated the framework as very simple. The strong view was expressed that it is prudent to pursue a security strategy of extending an existing control framework with which all stakeholders have some significant level of experience and understanding. An extension solution strategy is seen as having a high adoption potential.

Authentication: Again the view was strong that the existing anonymity of all mobile code and their information transmissions was the major challenge to more effective information security management. Authentication within a globally trusted environment is seen as the clear solution.

Enhanced end-user management: This was the final very strong view of all interviewees. The security framework would significantly encourage open and transparent information collection by the very sizable vendor community associated with the software paradigm. The exception notifications would significantly assist in better identifying deceptive, misleading, and fraudulent behaviour within the software paradigm. End-user management was also viewed as very much required in terms of the architectural placement of the solution control. End-user deployment of the IS security controls (supported by the appropriate education/awareness program) is seen as vital in achieving a robust solution. Server-based and network-deployed solutions are seen as necessary but not sufficient.

Residual risk assessment: All interviewees – based on the labelling criteria of Table 1 – rated the residual risk assessment of the proposed security framework as MEDIUM: whilst the threat-source remains highly motivated and capable, the proposed security framework significantly impedes the vulnerability from being exercised. This rating is contrasted with the MEDIUM+ to HIGH rating originally suggested by interviewees (Table 2) at the beginning of the research process.

4 CONCLUSIONS

The goal of this research was to contribute to the effective security management of user-information collecting software (including spyware and adware). The research produces three important factors: (1) spyware extrusions are 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 can be extended to provide greater specificity and therefore greater security. The security framework described in this paper proposes a new and more comprehensive direction for software code signing. 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 all transmission requests 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 and manage information transmissions from the host machine.

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

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.

CSI. 2009. *2005 CSI Computer Crime and Security Survey*. (Online) Available at: http://www.fbi.gov/publications/ccs2009.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.

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. Available at: <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.

Miles, M. B., & Huberman, A. M. 1994. Qualitative data analysis (2nd ed.). London: Sage Ltd.

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

Potter, J., & Wetherell, M. 1994. Analysing discourse. In A. Bryman & R. Burgess (Eds.), Analyzing qualitative data (pp. 47-68). London: Routledge.

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

Strauss, A., & Corbin, J. 1998. Basics of qualitative research (2nd ed.). Newbury Park, CA: Sage.

Thomas, D. R. 2006. "A General Inductive Approach for Analysing Qualitative Evaluation Data". American Journal of Evaluation (27:237). Available at: http://aje.sagepub.com/content/27/2/237.

van Manen, M. 1990. Researching lived experience: Human science for an action sensitie pedagogy. London, Canada: Althouse.

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

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

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.