2011

Trusted interoperability and the patient safety issues of parasitic health care software

Vincent B. McCauley
Medical Software Industry Association

Patricia A H Williams
Edith Cowan University

DOI: 10.4225/75/57b547bfcd8c7
Originally published in the Proceedings of the 9th Australian Information Security Management Conference, Edith Cowan University, Perth Western Australia, 5th -7th December, 2011

TRUSTED INTEROPERABILITY AND THE PATIENT SAFETY ISSUES OF PARASITIC HEALTH CARE SOFTWARE

Vincent B McCauley[1] and Patricia A H Williams[2]
[1]Medical Software Industry Association, [2]Edith Cowan University
[1]vincem@mccauleysoftware.com
[2]trish.williams@ecu.edu.au

Abstract

With the proliferation of software systems and products in the healthcare environment, it is increasingly common for such software products to be constructed in a modular design. However, for modular software to be securely interoperable with other software products requires agreed consistent and accountable interfaces. This agreement may take the form of bilateral vendor to vendor arrangements or via a trusted external third-party who coordinates agreed interaction methods, such as a jurisdiction. Standards are a particular form of mutually trusted third party. Unfortunately, this agreed method of interoperability is not always present in vendor software. Where one software product or module interacts with another, in the absence of any agreement, it is referred to as "bolt-on". It is perhaps more descriptive to refer to such software in terms of its potential to cause harm and refer to it using the biological analogy of "parasitic" software and associated "host" software. Analogous to biological systems, parasitic software can operate by data injection into or data extraction from, the associated host database. Both forms of parasitic software exploit access mechanisms or security flaws in the host software independent of the host vendor and in ways not intended or supported by the host vendor. This paper discusses the mechanics of this security vulnerability and more importantly, the potential adverse consequences to patient safety of such susceptibilities. As Australia moves to a national connected e-health system these issues are causes for grave concern. This paper provides a case study of this insecurity to highlight the problem, promote discussion and encourage potential change.

Keywords
Medical software, health information security, third party software, healthcare software, bolt-on software.

INTRODUCTION

The explosion of software products and information systems for healthcare has seen an increase in development of these software products in modular form. Many of these products are sitting 'on top of' or 'alongside' existing software providing additional services. The additional services range from aggregation of data for the purposes of healthcare management to programs that facilitate integration of systems with existing products and databases. Healthcare providers currently have to rely on independent third party software/services vendors for access to the essential services of the new national e-health system, such as the Health Identifiers Service (Medical Software Industry Association, 2010). The importance to the healthcare environment is in the benefits that such third party software can provide in both the integration of electronic services and in providing facilities such as clinical audit tools and healthcare practice analysis.

The third-party software is referred to as 'bolt-on' because it is providing specific functionality outside the normal applications used. As such, these products are of modular design because they are not systems that can run independently of a main (or host) system and its associated databases. However, for modular software to be securely interoperable with other software products requires agreed consistent and accountable interfaces. By definition these applications bolt-on and make use of the existing systems and databases. Parasitic software is a form of bolt-on software that the host system is not aware of or has no agreed consistent and accountable interfaces with.

This paper discusses the mechanics of the security vulnerabilities associated with bolt-on and parasitic software, and their potential impact. This is considered in the context of the Australian e-health environment with specific examples and the important issue of patient safety.

SECURITY ISSUES WITH SOFTWARE

Research has shown that most common vulnerabilities in software are caused by a small number of coding errors and practices (CERT, 2011). These common vulnerabilities and those specific to third party and parasitic software include buffer overflows, input manipulation and application authentication.

Common vulnerabilities

Buffer overflows (or boundary checking) occur where a program writes outside the buffer limit effectively violating memory protection. Due to the various mechanisms that can induce buffer overflows, they are a specific vulnerability in software that can be readily maliciously exploited. These mechanisms include simple arithmetic variable overflows, stack-based overflows, and heap-based overflow attacks. This issue is particularly prevalent when using languages that do not inherently have boundary checking as part of their construction such as C and C++ (Goodrich & Tamassia, 2011).

Input manipulation is vulnerability where filtering and sanitization of data input is not performed adequately. This is not restricted to direct data input; it also applies to data passed from client to server as in web based applications (Ravel & Fichadia, 2007). This vulnerability includes SQL data injection, cross-site scripting, light weight directory access protocol (LDAP) injection used for accessing and updating directories, and application specific input as is common with data passed between web browser and web servers. Format string attacks are another specific type of input manipulation error.

Application authentication, where a user is verified before being allow access, is the basis for right of entry to, and use of, an application and its associated data. Control of connections to a database can be complex and depend on server security controls, database access control, and access restrictions (Burtescu, 2009). In the case of third party software this refers to the authentication of another application to have access to the host application and its associated databases. To date, this third party authentication has been mainly unconsidered in the design of existing healthcare software.

Other vulnerabilities relate to the use of application add-ons (usually web based) and cookies, the manipulation of session ID's, lack of change management control, and the security present in the operating system and databases.

Parasitic software vulnerabilities

Parasitic software has specific application of the common vulnerabilities and as such presents specific threats to host software.

Buffer overflows

Buffer overflows are a real vulnerability in non-standard, parasitic software (Posey, 2005). This is an issue particularly if the host application is running without minimized privileges. Also, where the parasitic software has not followed established standards of development, or has not been developed consistent with the style and construction of the applications and database that it is interacting with, this is a significant threat.

Input manipulation using injection

Analogous to biological systems, parasitic software can operate by data injection into or data extraction from, the associated host database. Both forms of parasitic software exploit access mechanisms or security flaws in the host software independent of the host vendor and in ways not intended or supported by the host vendor. Cross-site scripting may become a larger issue for healthcare as more systems become web-based rather than server-based applications (Symantec, 2008).

Application authentication, unintended uses and change management

The application architecture relies on the application knowing the level of security of its databases and operating system interfaces. When a third party software product is introduced it must rely on an agreement between the applications and provide connection to the appropriate databases. Unfortunately, the databases are often not secured and this access does not require a secure level of authentication.

In addition, many existing software products were not designed with the new e-health systems in mind, and therefore lack sufficient controls in regard to bolt-on and parasitic software programs. In effect, unintended use of their associated databases is occurring without sufficient security design measures in place. This leaves the

integrity of the data at risk. Further, when program updates occur in the host program, the parasitic software, that has not (by definition) been developed in synchronisation, may be unaware of the changes to the host functionality and database resulting in potential malfunctions and vulnerabilities. Where a bolt-on program has agreement with the host system this may not be an issue as long as there is tested integration between the host application and bolt-on product.

PATIENT SAFETY IN HEALTHCARE SOFTWARE

Conformance, compliance and standards

The role of standards is to ensure safety and reliability, and that software performs consistently in the way it was intended to perform, to a tested and expert specification. Providing a uniform set of rules using a standard also provides a metric for conformance testing (CERT, 2010; Standards Australia, 2010). In considering the security in using bolt-on third party software, the solution lies in the agreements between the software product and vendors. This agreement may take the form of bilateral vendor to vendor arrangements or via a trusted external third-party who coordinates agreed interaction methods, such as a jurisdiction. Standards are a particular form of mutually trusted third party. Unfortunately, this agreed method of interoperability is not always present in vendor software. Where one software product or module interacts with another, in the absence of any agreement, it is referred to as a "bolt-on". It is perhaps more descriptive to refer to such software in terms of its potential to cause harm and refer to it using the biological analogy of "parasitic" software and associated "host" software. Figure 1 illustrates the different types of software agreements applicable to bolt-on software. As can be seen, the bilateral agreement allows for accreditation, and therefore assurance, between the host and bolt-on providers. Where a trusted third party such as a jurisdiction manages the interactions between software vendors this provides a mediation alternative. Conformance to standards is an example of this. Where no concurrence in use of standards, or demonstrated conformance, or agreement is established the relationship is referred to as parasitic.

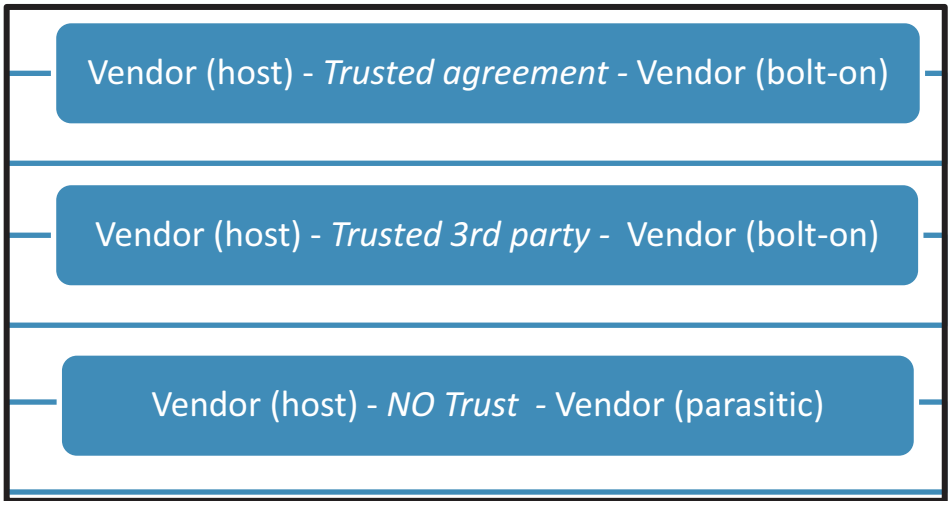

Figure 1- Trusted interoperability of modular software security agreements

Examples of safety issues managing patient identity information

These issues associated with parasitic software are not restricted or peculiar to the healthcare domain. However, the potential for more devastating outcomes from the exploitation of the vulnerabilities are specific to this domain. A well known example of such parasitic software using data extraction is the PEN Computing Sidebar application (PEN Computer Systems, n.d.). It accesses data directly from the database of the host software (e.g. Medical Director) using access mechanisms that were put in place by the host vendor to permit interoperability amongst its own product suite. However, this software does not manipulate patient identity information and the majority of extracted data is presented as aggregates.

The safety issues apparent with parasitic software when managing patient identification and identity, in particular the new Australian e-health individual health identifiers (IHI) are one example. The following are some scenarios of how parasitic software can lead to serious patient safety risks in the context of managing patient IHIs.

Data injection example:

A typical circumstance might be

1. The parasite software designer identifies an "unused" column in the host patient database table. For example one called patient_id2;
2. This is done without input from the Host vendor. The column is not populated in any of the clinical desktop systems examined;
3. The column has a data type compatible with the 16 digit format of the IHI;
4. Unknown to the parasite software designer, this column is intended for future use as the patient identifier for anonymous pathology testing as required for HIV tests. It has been designed to use the same ISO based identifier standard as the IHI. The patient identifiers will be unique across sites but specific to the host software product. It will be sent to the pathology lab along with the sex and DOB as the patient identifiers;
5. The parasitic software is deployed. It accesses the host patient table and uses the patient demographics to obtain a corresponding IHI from the Medicare HI service and these are stored in the host patient_id2 column; and
6. Sometime in the future, a host software update implements the use of patient_id2 for anonymous pathology testing. The parasitic software vendor is unaware of this update.

Scenario 1: Data injection

The parasitic software extracts what it thinks is an IHI from the host and sends it in a message to software that is not connected to the validating HI Service. However, it is in fact the host software's new internal patient identifier. This incorrect "IHI" which may match to some other patient, rapidly spreads to the attached systems.

Scenario 2: Data injection

The host software sends an electronic request for an HIV test to a laboratory. The parasitic software notes the patient associated data has been updated and checks the IHI against the HI service. It finds the IHI is incorrect (because it is in fact a host internal patient identifier) and updates it to the value obtained from the HI service. When the HIV result message is returned noting a new positive result, the patient identifier cannot be matched and the result is discarded. The patient's treatment is delayed and a number of sexual partners are consequently infected.

Data extraction example

In these scenarios the parasitic software extracts a copy of the patient demographics and the host software's unique patient identifier into a database managed by the parasitic software. It accesses the HI service and adds IHIs to a locally maintained database.

Scenario 3: Data extraction

The host software operator cleans up old patients who have not been seen for some years. These are assigned a new archive patient identifier key and the previous patient identifier keys are re-used as new patients are added to the database. This is necessary in some mature products as the internal patient identifier keys are only sixteen bits and support a maximum of 37267 active patients. The parasitic software is designed to intercept messages being sent out from the host and add IHIs. It does this by matching the host patient identifier contained in the message to its IHI table. New patient identifier keys are assigned starting at the highest patient identifier of the cleaned up patient data. This leads to some patient identifier keys for archived patients being reassigned to new patients. Hence patient identifier keys that have been re-used in the host software will subsequently have the incorrect IHI inserted. Message receiving software that has access to the HI service will detect this error and reject the message. Receiving software that is not able to access the HI service will propagate the incorrect IHI. This may lead to pathology results being filed against the wrong patient or pathology requests and specialist referrals not being received.

It is common practice in software to read the unique patient identifier from the underlying database and then use this as an update key. Where the patient identifier key is shared across different software components any update must involve a two-stage commit process to keep entries in both products synchronised. However, in the absence of this practice in loosely or poorly coupled systems, it is impossible to keep the two separate databases synchronised as there is a deficiency in the required tightly coupled commit process. Failure to implement a two-stage commit for updates across heterogeneous systems, risks updates being applied to only one system and the data becoming inconsistent.

Scenario 4: Data extraction

The host software operator enters two new patients David Smith and his twin John Smith. The parasitic software scans the host patient table, notes these two new entries and adds them to its local patient table after obtaining their IHI from the HI service using their name, DOB, sex and associated Medicare number. David is seen by a doctor that day but John has an appointment for next week. When John comes in, it is discovered that David's consultation notes have been entered by mistake into John's record. Because the notes can only be altered by David's doctor who is not working that day, the front desk staff fix this by changing John's name and demographics to David's and vice-versa. Now the notes are associated with the correct patient. However, the parasitic software has no mechanism for being informed this has occurred and has incorrect data associated with the Host's patient identifier. Subsequently, outgoing messages have the incorrect IHI inserted and this is disseminated. If the parasitic software manages incoming messages on behalf of the host, the data will be matched to the wrong patient if the IHI, surname and DOB are used to perform patient matching.

In this case the two record updates effectively create a transposition of records. Products that have significant penetration of the Australian primary healthcare marketplace currently allow this scenario in their software. Whilst it is acknowledged this is not current best practice software design, it reflects the level of flexibility often required by medical practices and only becomes of major concern when third-party software is also accessing the patient demographics in an uncontrolled manner.

Example summary

In general, maintaining coherence of shadow tables is impossible without a closely coupled communication (insert/update/delete) mechanism. The nature of the software host/parasite relationship precludes this possibility. Once an exact matching view of the host demographics table in the parasite's database is lost, it becomes possible for IHIs to be associated with stale or incorrect patient demographics. Depending on the functions using IHIs that are incorporated into the parasite software, this can have significant local patient safety implications. If the parasite software is used to insert IHI's into outgoing messages, the invalid information can be rapidly dispersed across the eHealth system. Similar arguments apply to other Australian e-health identifiers for individual healthcare providers (HPI-I) and healthcare provider organisations (HPI-O) although the implications for patient safety may be less significant. Nonetheless incorrect or invalid HPI-Is and HPI-Os could cause significant disruption. In the Australian context, the inability to generally be able to validate the HPI-Is or HPI-Os using the HI service in its current design, to ensure that they are matched to the correct provider and organisation, means that they are more vulnerable to error and dissemination of erroneously matched identities.

The Standards community has recognised the safety concerns that arise when multiple interacting e-health software components are introduced without adequate coordination and attention to demographic data coherence. It has taken many years to develop this standards based approach. The international standards organisation Health Level 7 (HL7) incorporates a Clinical Context Object Workgroup (CCOW) Committee which works in conjunction with the HL7 Security Committee specifically to define standards for 'linking' applications in a secure manner. This standard means that applications are aware of the context in which they operate and ensures that data, and in particular patient demographics, are synchronised and their integrity assured (HL7, 2010).

RECOMMENDATIONS FOR CONFORMANCE AND COMPLIANCE

There are minimum levels of safety that must be assured before deploying software in the health environment, and economics are independent of this minimum level. Indeed, there are always implementation and development costs, however it is only above this minimum threshold that such costs become relevant. At advanced levels it is acceptable that cost benefit analysis should be considered. However, where there are real

risks to patient safety that are unacceptable, the fundamental software design needs to be changed and the insecure software design practices need to be forbidden.

Recommendations for software conformance and compliance testing need to be contextualised for the specific functionality of the software integration in place, and thus its status and resulting adverse security impacts, would be minimised.

Using the examples above of the HI Service, recommendations are given below such that software submitted for HI conformance/compliance testing should declare whether it manages IHIs on behalf of other software products either by injection or extraction. If it does manage IHI's, it must conform to the following:

1. Describe the communication mechanism with the underlying (host) software, noting potential delays in recording demographic changes;
2. Subsequent testing should be performed both within and outside of the time window in which changes to the host tables are reflected in the IHI Manager's tables;
3. Describe the purpose for which Health identifiers are maintained and used – this may have implications for subsequent test use cases. This should be noted on the test report;
4. Software must be tested with any and all software on whose behalf it manages Health Identifiers, as a single test unit. All possible combinations of host and parasite need to be tested;
5. The software that connects to the HI service must demonstrate that it continues to maintain correct patient demographics and the correct associated IHI for every possible operation involving patient demographics supported by each of the intended underlying software hosts (patient demographic data coherence), and especially for patient record merging and de-merging operations; and
6. The testing report should nominate each specific combination of software and the versions of each that were tested.

The examples discussed above are not a comprehensive list however they give an insight into the type of exploitations that are possible. Not all software of this nature is 'unsafe', but given the potential for rapid proliferation of errors, opportunities for unprofessional practice and exploitation of database vulnerabilities, there is significant cause for concern. Incorrect patient identity information can be disseminated as demonstrated by the incident last year, in 2010, when there was an incorrect update by Medicare to the Medicare Online Patient Verification system such that it returned incorrect Medicare numbers. Because of the closely connected nature of the eHealth IT systems, these incorrect values were distributed rapidly to local attached systems and dispersed across the community. This type of incident will be a major issue as Australia implements its connected e-health system. It is the potential adverse consequences to patient safety of the security susceptibilities discussed that require addressing at the level of software design, implementation and use.

CONCLUSION

Assurance of patient safety should outweigh all other issues in regard to the use of software in healthcare. The examples given with the current specification of the HI service and the interaction with Australian legislation highlights the problems with parasitic bolt-on software. These types of problems will only become more prominent and more prolific as the e-health systems in Australia are established and as healthcare providers make more use of the e-health opportunity. There is no doubt that the use of bolt-on programs can be of benefit in many areas of healthcare provision and administration as long as appropriate governance and testing is in place As Australia moves to a national connected e-health system these issues are cause for grave concern. The recommendations for improvement given in this paper address some of these specific concerns. However, what is required for a safe future in the healthcare software industry is a broader and more coordinated approach to software development and interoperability where third party software is concerned. There is no doubt that the benefits to our healthcare system and improved patient outcomes can be realised as long as due consideration is given to this important and fundamental software security issue.

REFERENCES

Burtescu, E. (2009). Database security – attacks and control methods. *Journal of Applied Quantitative Methods, 4* (4), 449-454.

CERT. (2010). *Secure coding standards.* Retrieved from http://www.cert.org/secure-coding/scstandards.html.

CERT. (2011). *Secure coding.* Retrieved from http://www.cert.org/secure-coding/.

Goodrich, M.T. and Tamassia, R. (2011). *Introduction to computer security*. USA: Pearson.

HL7. (2010). Clinical Context Object Workgroup. Retrieved from http://www.hl7.org/Special/committees/visual/overview.cfm

Medical Software Industry Association. (2010). Submission on Proposed Regulations for the Healthcare Identifiers Service. Retrieved from http://www.health.gov.au/internet/main/publishing.nsf/Content/eHealthregs-047/$FILE/047_Medical%20Software%20Industry%20Association_12-04-10.pdf

PEN Computer Systems. (n.d.). *Primary care sidebar.* Retrieved from http://www.pencs.com.au/product-sidebar-overview.html

Posey, B. M. (2005). *Buffer overflow attacks: How do they work?* Retrieved from SearchSecurity.com

Ravel, V. and Fichadia, A. (2007). *Risks, controls and security: Concepts and application*s. NJ, USA:Wiley.

Standards Australia. (2010). *Benefits of standards*. Retrieved from http://www.standards.org.au/DevelopingStandards/BenefitsofStandards.aspx.

Symantec. (2008). *Symantec Internet Security Threat Report: Trends for July–December 07.* Retrieved from http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_exec_summary_internet_security_threat_report_xiii_04-2008.en-us.pdf