Assurance: the power behind PCASSO security

[1]Dixie B. Baker, Ph.D., [2]Daniel R. Masys, M.D., [1]Russell L. Jones, and [1]Robert M. Barnhart
[1]Science Applications International Corporation (SAIC) and
[2]University of California, San Diego (UCSD)
La Jolla, CA

The need for security protection in Internet-based healthcare applications is generally acknowledged. Most healthcare applications that use the Internet have at least implemented some kind of encryption. Most applications also enforce user authentication and access control policies, and many audit user actions. However, most fall short on providing strong assurances that the security mechanisms are behaving as expected and that they cannot be subverted. While no system can claim to be totally "bulletproof," PCASSO provides assurance of correct operation through formal, disciplined design and development methodologies, as well as through functional and penetration testing. Through its security mechanisms, backed by strong system assurances, PCASSO is demonstrating "safe" use of public data networks for health care.

INTRODUCTION

For the past three years, the Patient Centered Access to Secure Systems Online (PCASSO)[1,2,3] project has been developing and evaluating a model system that enables patients and providers to access medical records through any Internet Service Provider (ISP). In some ways, the PCASSO model is similar to many other technical implementations of secure Internet access. PCASSO uses the Secure Sockets Layer (SSL)[4] to achieve client-server authentication and to establish a secure channel. PCASSO uses a combination of a system-generated password and a challenge-response token for user authentication. The application is contained in a Java applet that is stored in a server and executes within a confined execution space ("sandbox"). PCASSO enforces a role-based access-control policy, and it audits user actions. From a functional perspective, the primary difference between PCASSO and many other "secure" healthcare applications is that it uses label-based access control to provide strong isolation between levels of sensitivity (low, standard, public deniable, guardian deniable, and patient deniable) within the server. What sets PCASSO apart from other implementations is its level of *assurance*.

METHODS

The international *Common Criteria for Security Evaluation*[5] defines assurance as "ground for confidence that an entity meets its security objectives" and identifies seven assurance classes: 1) configuration management; 2) delivery and operation; 3) development; 4) guidance documents; 5) life-cycle support; 6) tests; and 7) vulnerability assessment. This paper describes PCASSO's assurances in these classes.

Configuration management

The PCASSO development environment includes sophisticated tools, based on the UNIX Revision Control System (RCS), that organize and provide accountability for all changes to the source code used to build the distribution. All changes to the source code are tracked, and each release includes a "signature" file indicating the ownership, permissions and checksum for all files in the release. This facilitates detection of any files that may be intentionally or inadvertently modified.

Delivery and operation

PCASSO must be installed by the privileged *installer* role defined in Data General's DG/UX B2 Security Option operating system (OS). DG/UX's advanced label-based access control mechanism protects PCASSO executables from virus infection, and patient data from access by unauthorized software. The server uses host and packet filters that prohibit administrative access from any machine other than trusted local machines. Advanced logging capabilities monitor critical aspects of PCASSO execution, and administrative tools allow the system administrator to query and analyze the audit trail to review system behavior, to identify potential system misuse and intrusion, and to view statistical reports. Finally, because the Internet is not sufficiently trustworthy to be used for the distribution of user accounts, passwords, certificates or challenge-response authenticators, more secure channels, such as personal communication or public mail, are used.

Development

Developmental assurances, which include both processes and architectural principles, provide confidence that the behavior of the system entity is well understood and that it can be maintained securely throughout its life cycle.

Policy modeling. A simple, yet powerful security policy model forms the foundation for the PCASSO design. To aid in the development and validation of this model, formal

1091—8280/99/$5.00

methods were used to express the model in Prolog, using a state-machine approach. Key aspects of the security policy model that enhance PCASSO's assurance are: *role-based access control* (individuals may access patient information in ways commensurate with their relationship to the patient); *least privilege* (individuals have only the authorizations they require); and *explicit authorization* (individuals have no default authorization).

Design simplicity. Security policy enforcement is allocated to a single component (the PCASSO server) that mediates all accesses, cannot be bypassed, and is small enough to be subjected to rigorous analysis, testing, and validation. PCASSO's object-oriented design minimizes complexity in both the server and Graphical User Interface (GUI) to enhance system integrity and assurance by controlling access to information through abstractions with well defined semantics and interfaces.

Least-privilege enforcement. A system entity should be able to perform only those actions, and to access only those data, required in performing its assigned functions assigned. Many systems authorize their security mechanisms far more privilege than they require (e.g., most UNIX servers run as *root*). PCASSO's system servers run with the minimal set of capabilities they require. The DG/UX operating system and the Trusted Oracle database management system confine the system access of users and processes by assuring that only those data and applications to which they are authorized are visible to them. The label-based access control mechanism separates and isolates individual sensitivity levels, resulting in the equivalent of seven execution domains separated by "virtual firewalls." Thus, if malicious code were to penetrate the server, its actions would be confined to a single level. The actions of each PCASSO user are contained within a single user process. PCASSO allows only two network-visible services: the Hypertext Transfer Protocol (HTTP) service and the PCASSO service itself. In the client, the Java Virtual Machine (JVM) mechanism is used to confine the Graphical User Interface (GUI) code (Java applet) to a well-defined execution domain ("sandbox"), with access limited to those resources to which it is authorized (e.g., diskette containing encryption keys).

Minimal reliance on untrusted components. PCASSO minimizes its dependency on untrusted components that have low assurance with respect to security, such as the Web server, client OS, and Web-browser. For example, the Web browser just issues the URL, and the Web server's sole function is to download the PCASSO applet.

Software certification. Software certification involves attaching the digital signature of a trusted certifier to a data file or application. PCASSO's own certificate authority is used to digitally sign the public-private key pairs issued to users, thus certifying their authenticity. The GUI (Java applet) is contained in a signed Java ARchive (JAR). The signing provides assurance that the applet originated from a trusted entity and enables the applet to assume the capabilities required for reading the diskette containing the X.509 certificate and the private key. The diskette is used to minimize costs while maximizing flexibility. A smartcard could be used to store keys and certificates, but would cost more and would require that users have smartcard readers.

Protection commensurate with risk. Because PCASSO is designed to provide access to highly sensitive information within an extremely hostile environment (the Internet), it incorporates protective measures aimed to counter known and anticipated threats in that environment. The PCASSO server runs on the DG/UX B2 Security Option OS, which was developed to meet the Class B2 requirements of the *Department of Defense Trusted Computer System Evaluation Criteria*.[6] B2 requires a modular architecture and penetration testing by a team with access to the full source code. In contrast, the more common C2 (e.g., NT, standard UNIX) does not impose any architecture requirements or penetration testing. The DG/UX OS enables PCASSO to incorporate all the features of a strong Internet firewall. PCASSO also incorporates specific measures to counter OS vulnerabilities in the client environment; these are described in an earlier paper[2].

Safe failure. Multiple techniques are used to ensure that no failure in a single security mechanism will result in a violation of the system security policy. Login requires user identification and authentication through 1) possession of a PCASSO SSL certificate disk, 2) knowledge of a valid PCASSO account/password pair, and 3) possession/knowledge of challenge-response authenticators. Certain security checks are redundantly incorporated into the GUI, network servers, and the Clinical Data Repository (CDR) code to ensure that no single failure will result in violation of the security policy. Moreover, despite the fact that all client-server communications are encrypted, PCASSO does not divulge patient identity in its client-server protocol. Patient identities are translated to-from nonce "context-IDs" when exchanged between the server and the client. Should the encryption fail, no datagram would contain both data and the patient's identity. Should an attacker incorrectly attempt to guess a context-ID, the session would terminate.

Tests

Functional testing, or "positive" testing, provides assurance that an entity conforms to its specification. That is, it performs the actions specified in its functional contract, and it provides the user the feedback described in its user documentation. Functional testing attempts to exercise the system under normal, expected circumstances and is often performed by walking through usage scenarios. Functional testing was performed at the component level and at the system level. For the servers, special test harnesses (typically written in C++) were used to allow rigorous unit testing of critical or security-relevant components. End-to-end functional testing was performed in the laboratory for each build. A combination of Java code and shell scripts was used to simulate users interacting with the system in specific use cases or scenarios. This software was used for automated regression testing of code modifications and performance testing of the system as a whole. Regression testing assures that the modified system performs as intended and that changes do not introduce new security vulnerabilities.

Vulnerability assessment

Vulnerability assessment consists of the identification and attempted exploitation of vulnerabilities in the system. Vulnerabilities may result from flaws in the design or implementation of a particular component, or they may arise from interactions among components. Potential vulnerabilities are assessed through penetration testing, which determines whether they could, in practice, be exploited to compromise the security of the system. Penetration testing exercises the system in ways that may not be expected or described in the documentation. Because penetration assesses whether a system can be forced to do something it is *not supposed to do*, it is considered a "negative" testing method. Our risk assessment[7] for PCASSO indicated that the greatest risk by far was a "hacker" penetrating the server and successfully assuming a privileged role (e.g., system administrator, provider), thereby gaining access to the medical information for hundreds of thousands of patients. Thus we focused our penetration testing on the PCASSO server and its connection to the Internet.

All of the penetration tests were performed by SAIC personnel with specialized expertise in system vulnerability assessment. The attackers used a combination of commercial assessment tools, underground "hacker" tools, and tools developed by SAIC. To ensure the independence of the assessments, none of the attackers were directly associated with the PCASSO project; the only knowledge they were given was the Universal Resource Locator (URL) of the server.

We report here the results of the most recent penetration testing, which was performed on the "live" PCASSO system installed at UCSD. At the time of the test (March 1999), the system supported over 165 users and contained the records of over 174,000 patients. The test was performed from SAIC's vulnerability assessment lab in McLean, VA. Because the penetration test focused on a specific target (PCASSO) with a visible Internet Protocol (IP) address (132.239.78.176), many of the tools needed for a "zero-knowledge" attack were unnecessary. So the attacker started with a determination of what services were active and waiting for session initiation.

The first tool used for service identification was UltraScan Version 1.5, a very efficient shareware port scanner developed by Michael Marchuk. The attacker scanned all ports from 1 through 65000 and discovered that the only active services were on ports 777 and 8000. Port 777 is a privileged port, meaning it usually runs under the ownership of a privileged account (such as root on a UNIX system). However, port 777 is not a standard privileged port such as port 23 for Telnet or port 25 for Simple Message Transport Protocol (SMTP). With only ports 777 (privileged) and 8000 available, the attacker quickly realized PCASSO provided very little to work with.

The next step was to determine whether another host, such as a firewall, was blocking access to the PCASSO server. From his NT 4.0 workstation, the attacker ran a traceroute program that revealed the IP address of the last node the IP datagram passed through before reaching the server. This IP address was determined to be a Cisco router, indicating that the router (which may or may not have employed packet filtering) was the only barrier between the attacker and the PCASSO server.

Packet filtering is good for blocking packets based on allow/deny rules such as "allow only SMTP traffic from the Internet" or "deny all inbound Telnet sessions." However, packet filtering cannot make decisions based on the payload (i.e., content) of the packet. Therefore, the attacker was free to attempt to make connections, using ports 777 and 8000, using Transport Connection Protocol (TCP) and User Datagram Protocol (UDP) services that were not meant to connect on those ports. The attacker hoped he would be able to get these services to do something that they were not intended to do; but he was not successful.

The attacker next used two commercial security vulnerability scanners to attempt to exploit vulnerabilities via ports 777 and 8000: Internet Security Systems' (ISS) Internet Scanner Version 5.4 and Network Associates' CyberCop Version 2.4. These tools include tests for most of the known vulnerabilities that "hackers" commonly

attempt to exploit. To help the attacker to decide which modules of the tools to use, he needed to determine the OS on which PCASSO was running. Some modules exploit vulnerabilities commonly seen in certain classes of OSs (e.g., UNIX), while others exploit vulnerabilities in specific OSs. (e.g., AIX specific vulnerabilities). The attacker executed ISS and CyberCop against PCASSO to determine the OS and its version, and both tools failed. The only information that CyberCop was able to provide was the Ethernet address of the network card. At this point, the attacker realized that PCASSO was going to be far more difficult to penetrate than most of the systems his clients ask him to test.

Recognizing that blind penetration was not going to work, he decided to see what he could find out about the PCASSO architecture. On the PCASSO Web site, he found an abundance of information about the architecture and design. After learning that the server was running on a B2 version of Data General's DG/UX OS, he recognized that penetrating it was likely to be a futile exercise.

From information available on the Web site, he concluded that port 8000 was what PCASSO was using for HTTP instead of the standard HTTP port (port 80). So his next step was to execute ISS and CyberCop in their Web-scanning mode. This mode (on both tools) attempts to exploit known vulnerabilities in the Common Gateway Interface (CGI) and other Web scripting languages. Poorly written CGI, Internet Server Application Programming Interface (ISAPI), and Perl scripts can be exploited to gain privileged access to an OS or back-end database (such as PCASSO's CDR). So the attacker executed the ISS CGI tests identified in Table 1 against port 8000. ISS found no vulnerabilities. The attacker then proceeded to execute the CyberCop CGI tests identified in Table 1 against port 8000. CyberCop found no vulnerabilities.

The attacker next tried a brute-force password attack. From his Web browser (Microsoft Internet Explorer 4.0), he attempted to login as a registered user. First, PCASSO loaded onto the attacker's computer a Java applet that launched the client, which contained a "virtual" keyboard (i.e., keyboard displayed on the screen), thus forcing the attacker to enter a user identifier (ID) and password using this virtual keyboard. Noticing that a radio button option labeled "PCASSO Card" under the Additional Authentication section was checked, the attacker attempted to uncheck it, but was not successful. The attacker realized that to be successful, he would need to guess a legitimate user ID, that user's password, and the value expected in the PCASSO Card frame.
After the attacker entered what he thought might be a valid user ID and a guessed password (ignoring the PCASSO Card frame), he clicked the "Login" button and immediately received the message "Please Insert Your PCASSO diskette and press OK when ready." This meant that beyond having a valid user ID, password, and PCASSO Card, the user must possess a diskette containing some secret value stored in software. Thus the attacker realized that a brute-force password attack using ISS, CyberCop, or any other automated password-guessing tool would be pointless.

Realizing that he was not likely to gain access to the PCASSO server to ultimately compromise its information, the attacker decided to try to attempt to disrupt its services via a denial-of-service attack, the most difficult type of attack to avert. He first executed ISS in its Denial-of-Service (DoS) mode by running the DoS attacks identified in Table 1 against ports 777 and 8000. These DoS attacks attempt to consume so much of the OS's resources that it effectively becomes paralyzed, disrupting any subsequent connection attempts as well as processes currently in execution. Successful DoS attacks can cause an OS to crash, reboot, and even corrupt data and critical system binaries. However, the ISS DoS attacks were unsuccessful. The attacker then executed the CyberCop attacks in DoS mode. Again the attacker was not able to interrupt PCASSO's ability to function properly and continuously.

The attacker was still determined to ascertain the service represented by the mysterious port 777 so that he could attempt to exploit it. Knowing that he could connect to some services using a telnet client such as SMTP, the attacker tried to telnet into port 777. Again, success eluded him. During each of his three telnet attempts, he got no response in terms of text characters, and his telnet client froze. Without knowing what service port 777 represented, the attacker was unable to identify and exploit known vulnerabilities in that service. At this point, he concluded his penetration attempts.

CONCLUSION

The objective of the PCASSO project was to build a model system strong enough to protect very sensitive medical information accessible over a highly threatening public network, the Internet. We used developmental practices known to produce robust software. We built PCASSO on a very strong OS. We have shown through several penetration attempts that the PCASSO server is stronger than the vast majority of Web servers and firewalls on the Internet today. But have we actually proven that PCASSO is impenetrable?

As stated so succinctly by our attacker, "Any information security professional worth his/her salt will tell you that there is no such thing as 100% security. To approach 100%

Table 1. Penetration Tests Run Against PCASSO Server

ISS CGI Tests:		ISS Denial-of-Service Tests:
PHF (CGI program) Check	Webcrawler Index Check	Ping Bomb
Guess CGI Bin Check	Accessing Above the WWW Server Root	Out of Band Check
List CGI Bin Check	Directory Check	SYN (TCP connection request) Storm
CGI Echo Check	URL Password Integrity Check	Data Flood
Root Dot-Dot Check	NCSA (National Center for Supercomputer	Open Close
HTTP Basic Authority Check	Applications Webserver Buffer Overflow Check	Log Flood
HTTPD Type Check	Nph-Test-CGI (CGI script) Check	ICMP (Internet Control Message
Vulnerable HTTPD Check	AnyForm CGI Check	Protocol) redirects to PCASSO
Index Check	FormMail Check	
Unresolved Links Check	ScriptAlias Check	
Default Names Check	Guestbook CGI Check	**CyberCop Denial-of-Service Tests:**
	Test-cgi "*" Check	ICMP Unreachable Check
CyberCop CGI Tests:	PHP.cgi File Printing Bug	Routed Append Check
Test CGI Check	PHP.cgi Buffer Overflow Check	In.comsat Check
WWW perl Check	Glimpse HTTP Check	PASV (File Transfer Protocol "Passive"
WWW PHF Check	Website Uploader CGI Check	mode) Check
Shell Interpreter Check	PHP Mlog Example Script Test	Portmaster Reboot Check
PHF Bash Vulnerability	PHP Mylog Example Script Test	Ping DoS Check
WWW Finger Check	WWWcount Stack Overrun Check	

assurance that a system is impenetrable would be cost prohibitive and resource intensive." This penetration exercise showed that PCASSO is not vulnerable to the numerous attacks that are well documented in the public domain. PCASSO has done this by significantly limiting the number of services that are available and by implementing a strong authentication sub-system.

A server that makes only two TCP/UDP services available is what security professionals call a "hardened" system. The attacker commented that from the "hacker's" perspective, PCASSO looks more like a firewall than a state-of-the-art system designed to allow providers and patients secure access to patient information over the Internet. Other commercial vulnerability scanners and custom-built tools might successfully thwart PCASSO's security. However, to successfully gain unauthorized access to PCASSO, the attacker would need to be extremely skilled, well funded, and have a lot of time. At the same time, PCASSO has an intrusion detection system that is constantly watching for unusual activity and attack signatures, so the attacker would need to break in very quickly to avoid detection.

The PCASSO team set out to prove that we could build a system that was "safe and effective." So far PCASSO has shown itself to be safe. The determination of effectiveness, which includes the usability and acceptability of PCASSO's client interface and its performance in delivering patient-specific data to authorized users, is currently underway.

Acknowledgements

This work is supported by a Health Information Infrastructure research contract N01 LM63537-00 from the U.S. National Library of Medicine.

References

1. Masys, DR and Baker DB. "Patient-Centered Access to Secure Systems Online (PCASSO): A Secure Approach to Clinical Data Access Via the World Wide Web," In Masys, DR, ed. *Proceedings of 1997 AMIA Annual Fall Symposium*, American Medical Informatics Association, Nashville, TN, p. 340-3, Oct 1997.
2. Masys, DR, and Baker, DB. "Protecting Clinical Data on Web Client Computers: The PCASSO Approach," In Chute, C, ed. *Proceedings of the AMIA '98 Annual Symposium*, Orlando, FL, Nov 7-11, 1998.
3. Baker DB, Barnhart R, and Buss T. "PCASSO: Applying and Extending State-of-the-Art Security in the Healthcare Domain," *Proceedings of the Annual Computer Security Applications Conference*, San Diego, CA, Dec 1997.
4. Netscape Communications Corporation. *The SSL Protocol*. Dec 1994.
5. *Common Criteria for Information Technology Security Evaluation*, Version 2.0, Common Criteria Implementation Board. Distributed through the National Institute of Standards and Technology.
6. *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, Dec 1985.
7. Baker DB. "PCASSO Risk Assessment." internal white paper, May 1997.