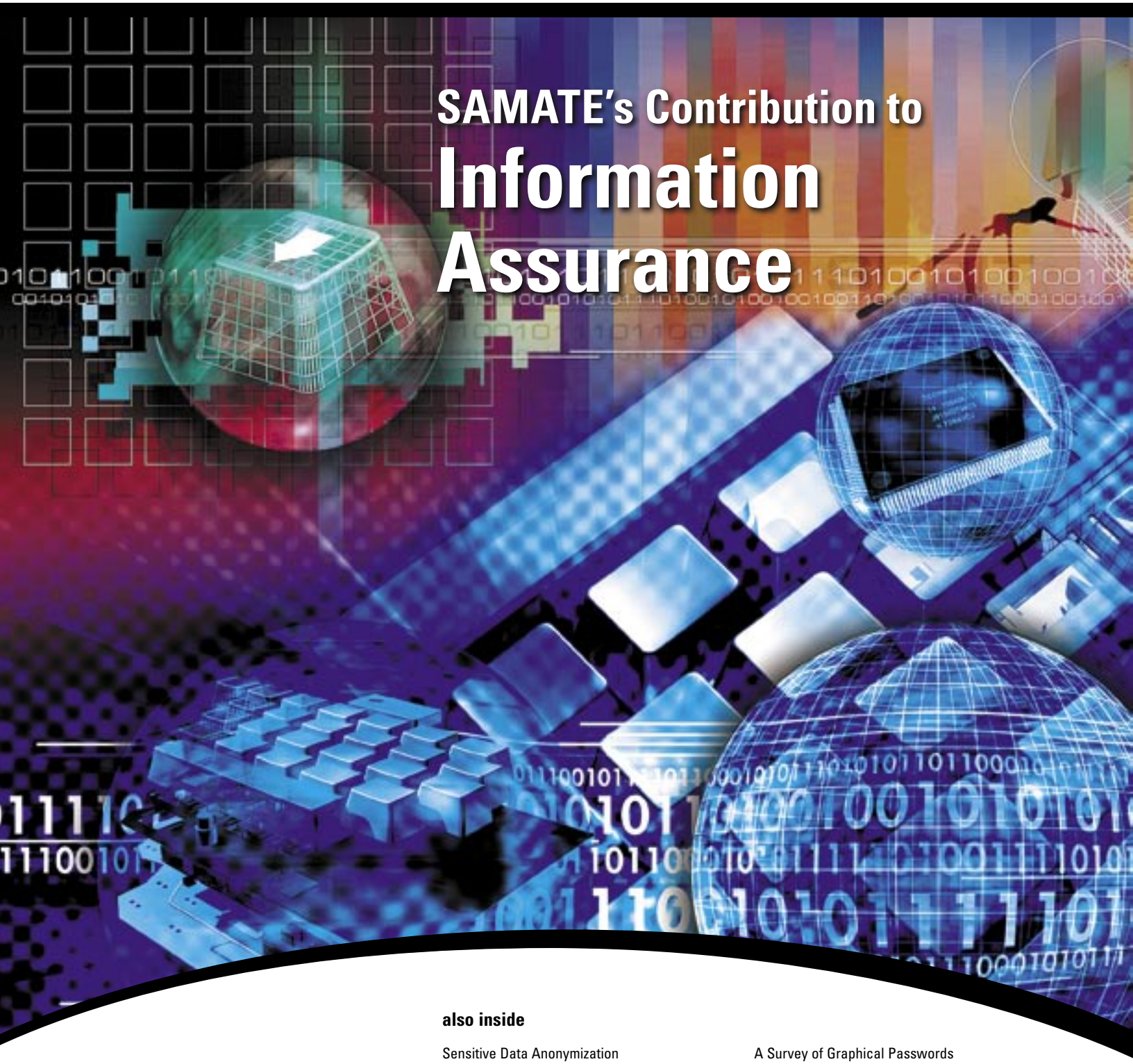
9/2

Volume 9 Number 2 • **Fall 2006**

IAnewsletter

The Newsletter for Information Assurance Technology Professionals



SAMATE's Contribution to Information Assurance

also inside

IATAC

contents

SAMATE's Contribution to Information Assurance

There is far too much software in today's information world to check manually. Even if people had the time to inspect thousands or millions of lines of code, nobody could remember all the constraints, requirements, and imperatives to make sure the software is secure. Automated tools are a must.

in every issue

About IATAC and the *IAnewsletter*
The *IAnewsletter* is published quarterly by the Information Assurance Technology Analysis Center (IATAC). IATAC is a Department of Defense (DoD) sponsored Information Analysis Center, administratively managed by the Defense Technical Information Center (DTIC), and Director, Defense Research and Engineering (DDR&E).

Contents of the *IAnewsletter* are not necessarily the official views of or endorsed by the US Government, DoD, or DDR&E. The mention of commercial products and/or does not imply endorsement by DoD or DDR&E.

Inquiries about IATAC capabilities, products, and services may be addressed to—

IATAC Director: Gene Tyler
Deputy Director: Greg McClellan
Inquiry Services: Peggy O'Connor

***IAnewsletter* Staff**
Promotional Director: Christina P. McNemar
Creative Director: Ahnie Jenkins
Art Directors: Bryn Farrar, Don Rowe
Copy Editor: Shelah Johnson
Designers: Dustin Hurt, Brad Whitford, Ricardo Real, Brittany Biggs, Kathy Everett
Editorial Board: Greg McClellan, Ronald Ritchey, Tara Shea, Gene Tyler

***IAnewsletter* Article Submissions**
To submit your articles, notices, programs, or ideas for future issues, please visit http://iac.dtic.mil/iatac/IA_newsletter.html and download an "Article Instructions" packet.

***IAnewsletter* Address Changes/ Additions/Deletions**
To change, add, or delete your mailing or e-mail address (soft-copy receipt), please contact us at—

IATAC
Attn: Peggy O'Connor
13200 Woodland Park Road
Suite 6031
Herndon, VA 20171

Phone: 703/984-0775
Fax: 703/984-0773

E-mail: iatac@dtic.mil
URL: http://iac.dtic.mil/iatac

Deadlines for future Issues
Winter 2006 Aug. 18, 2006

Cover design: Dustin Hurt
Newsletter design: Bryn Farrar, Donald Rowe

IATAC Chat

Gene Tyler, IATAC Director

This year, I have been given the opportunity to attend and brief for so many occasions and conferences. I would certainly say it is a testament to IATAC—who we are, and what we can do, for the Information Assurance (IA) community.

As I sit at my desk contemplating what to discuss in this Directors Chat, I begin reflecting on how much has occurred in such a short time. Only just a few months back, we made our move out to our new One Dulles facility. Just one month later, we had our first ever Open House, which was a fantastic success. Individuals from the government, academia, and small businesses were invited for an IATAC Expo of sorts. We had various stations positioned around our office space with experts discussing their respective topics. The stations included Total Electronic Migration System (TEMS), IATAC Information Systems, IATAC Publications, Enterprise Mission Assurance Support Service (eMass), IATAC Core capabilities, National Institute of Standards and Technology (NIST), NetOps, Multimedia and Creative Solutions (MACS), *etc.* This was an excellent opportunity for individuals to learn more about IATAC and our capabilities while mixing and mingling with the core staff and as well as others from the IA community.

Next, began a whirlwind of conferences. In April, the Defense Technical Information Center (DTIC) hosted its first Commanders Conference, concurrently with the 32nd Annual DTIC Conference. Here, with just two other Information Analysis Centers (IACs), I was afforded the distinct opportunity to brief commanders, scientists, engineers, and professionals in the information science, technology, Research and Development (R&D) and acquisition communities representing the Department of Defense (DoD), other federal agencies, and contractors. Additionally, we were able to host a booth, where most attendees stopped in to obtain more information from the core staff.

Next, I was off to Omaha for the first Defense Technical Information Center's IACs Industry Day, sponsored by the 55th Contracting Squadron. Industry Day promoted the IACs and included presentations on the DTIC program, display booths for the IACs, one-on-one sessions, and presentations on the mentor-protégé program and partnering. We greatly appreciated the 55th and DTIC hosting this event.

Just one day after retuning from Omaha, I was back on a plane, this time to Osan, Korea. While there, I presented at the U.S. Forces Korea (USFK) Information Assurance Conference. IATAC was invited to speak specifically on IATAC products and how they can assist the USFK IA community. This year, I have been given the opportunity to attend and brief for so many occasions and conferences. I would certainly say it is a testament to IATAC—who we are, and what we can do, for the IA community.

In this edition of the *IAnewsletter* you will find various articles that run the spectrum of IA information. The *SAMATE's Contribution to Information Assurance* article explains what the NIST Software Assurance Metrics and Tool Evaluation (SAMATE) project is and how it seeks to help answer questions on evaluating tools. One of our other articles is titled, *Removing Security through Obscurity from Software Watermarking*. This article focuses on the increasing trend of software piracy and how security professionals are researching various software watermarking techniques as one means to be used as tools to combat this issue. *Vulnerability Analysis of J2ME CLDC Security*, while a more technical piece than the other articles, is very applicable for individuals running mobile applications. Finally, *A Survey of Graphical Passwords* is a fascinating article on the concept of graphical passwords techniques. I encourage everyone to read this article, as it is not only intriguing but one of the authors, Professor Ying Zhu of the Department of Computer Science Georgia State University, is our featured subject matter expert (SME) and Georgia State is our featured university in this edition. ■

SAMATE's Contribution to Information Assurance

by Paul E. Black

There is far too much software in today's information world to check manually. Even if people had the time to inspect thousands or millions of lines of code, nobody could remember all the constraints, requirements, and imperatives to make sure the software is secure. Automated tools are a must.

These tools can help design and build the right software in the first place, for instance, checking protocols, consistency with rules, and properties. Preventing flaws at the beginning of the software life cycle is the best way to get high quality and highly reliable software.

But what if the system being designed includes commercial, off-the-shelf (COTS) packages? How can a contractor thoroughly audit or check large packages from subcontractors? What kinds of flaws does the current development process leave? Does a new software process yield better quality software? To address these questions, the finished software must be checked. Again, the quantity of software requires automated software checking or at worst manual checking of exceptional instances found by automated means.

To be sure, testing is a vital part of assurance, too. If one does not have access to the source code, which is often the case with COTS packages or Web services, testing may be the only feasible way to gain assurance. Even when the source code or the binary are available, testing can be closer to actual use. Testing can catch configuration or system problems that are taken for granted when code is examined. On the other hand, reviews can find problems that are unlikely to be found by testing. For instance, a malicious backdoor that grants special access for a particular user name, say "matahari," cannot feasibly be found by functional, or black box, testing.

Another advantage of automated tools is that they can be updated and rerun relatively quickly when a new type of flaw is discovered or the security policy is changed. It is impractical to recheck everything manually for apparently minor changes in the system.

The SAMATE Project

Which tools find what flaws? Backing up, what is the list of all flaws to be found? Can tools check compliance with internally developed style or guidelines? If a tool passes a system with no outstanding alarms, how secure is system, really? Is the new version of a tool "better" than the preceding version?

The National Institute of Standards and Technology (NIST) Software Assurance Metrics and Tool Evaluation (SAMATE) project seeks to help answer these and other questions. The SAMATE Web site [1] explains that the project, begun in late 2004, is largely funded by the Department of Homeland Security (DHS) to help identify, enhance and develop software security assurance (SSA) tools. NIST is leading in (A) testing software evaluation tools, (B) measuring the effectiveness of tools, and (C) identifying gaps in tools and methods.

Although much work has been done in these areas, there is little coordinated, comprehensive, thorough, and objective work uniting all these. Instead we see isolated papers comparing different tools, surveys of methods and techniques, endorsements and experience reports, and best practices Web sites.

The SAMATE project is producing and catalyzing:

- a common enumeration of software weaknesses and flaws
- a taxonomy of SSA tools
- a survey of SSA tools and companies
- specifications of SSA tool classes
- detailed test plans and test sets for SSA tool classes
- metrics and measures for software and for SSA tools
- white papers pinpointing gaps in tool functions and proposing research requirements for new tools and new tool classes
- proposals for experiments and studies

Workshops

This project's scope is very broad, and our particular group in NIST does not have as much background in security and software assurance as others. To build collaborations and reach commu-

nity consensus, SAMATE has held several public workshops.

The first workshop, in August 2005, examined the state of the art in security assurance tools, particularly those that detect security flaws and vulnerabilities. The workshop was also the beginning of a standard reference dataset of programs with known flaws. Forty-five people attended, including representatives from the federal government, universities, more than a dozen tool vendors and service providers, and many research companies. The proceedings, including presentations and meetings notes, are published as NIST Special Publication 500-264. [2]

In October, we sponsored and hosted an Open Web Application Security Project (OWASP) conference.

In Long Beach in November 2005, we produced a workshop co-located with the Institute of Electrical and Electronics Engineers (IEEE) Automated Software Engineering (ASE) conference. This workshop convened researchers, developers, and government and industrial users of software security assurance (SSA) tools to discuss and refine the taxonomy of flaws and functions, come to a consensus on which SSA functions should first have specifications and standard tests developed, gather source code analyzer tool developers for "target practice," see how reference datasets fare against various tools, and identify gaps or requirements for research funding.

Working with others, we brought a very early version of the software reference dataset (SRD). Participants ran their tools against a subset of the SRD to demonstrate the state of the art in finding flaws and to provide suggestions for extensions to and improvements of the SRD. [3]

We held a Static Analysis Summit on 29 June 2006 in Gaithersburg, Maryland. [4]

A Taxonomy of Flaws

To accurately determine how well a tool checks for flaws, one must begin with a taxonomy of flaws. A taxonomy is not merely a list, but an organization into classes with shared characteristics. For instance, buffer overflow is a well-known (and unfortunately still widely occurring) type of flaw. But the classification "buffer overflow" can be further refined into heap or stack overflows, underflows or overflows, *etc.* In fact, the *CLASP Reference Guide* [5] lists eight different types of overflows. Even finer distinctions may be important to language designers or tool researchers, but may be unimportant to the programmer.

Authors have created and published many taxonomies of flaws. [6] [7] [8] For instance, MITRE grouped repeated problems listed in the Common Vulnerability and Exposures (CVE) [9] into a list of vulnerability examples. These works approach the problem from different views and define flaws differently, but have limited effort to reconcile the definitions, classifications, and details. SAMATE workshops catalyzed work to come up with one common enumeration of weaknesses. [10] Over time the taxonomy is sure to expand and change, but work can be shared instead of starting over for each good idea.

A Taxonomy of SSA Tools

Having a taxonomy of weaknesses, can we start testing tools? In a project of such ambitious scope, effort must be prioritized: we must choose which kinds of tools to look at first and which must be left for the future. Then, how do we choose which classes to work on? We must be able to list all classes of SSA tools so we can rationally (or at least, coherently) decide which ones *not* to work on. It follows we must also have a taxonomy of software assurance tools. The proposed taxonomy is organized around four facets:

- life cycle phase
- automation level
- approach
- viewpoint

The life cycle phase corresponds to the type of artifacts used, *e.g.* specifications, source code, executable, *etc.* It is documented as a simple waterfall model, even though more elaborate models are often better for the software process.

The automation level is a simple classification of how much human expertise,

effort, or knowledge is required. Level 0 is manual procedures, like code review. Sometimes there is no replacement for human involvement. The next levels have varying degrees of automation:

1. analysis aid
2. semi-automated
3. automated

Level 1 is analysis aids that help human analysts, but have no particular software assurance function themselves. Some examples are call-graph extractors, configuration control systems, or random test generators. Semi-automated tools or techniques at level 2 are targeted toward assurance, but need varying degrees of human judgment for extreme cases or to make a final decision. Most code and Web scanners fall in this category. They may point out things that are certainly flaws, but in other cases can issue only warnings about potential flaws. A human must then check and make the final determination. Finally, a firewall is an example of a completely automated tool at level 3. It takes action on whether to pass, trash, or cache packets without human intervention. Manual setup or auditing of automated tools does not make them semi-automated.

A tool may take four different approaches to software assurance: preclude the flaw from possibly occurring, detect a flaw or its exploit and report it, mitigate flaws to reduce or eliminate damage, and react to a flaw or its exploit. Choosing another language instead of C precludes most buffer overflows. Source code and Web scanners take the approach of detecting flaws. A multi-level security system can mitigate many security flaws. Finally, an intrusion-detection system reacts to exploited flaws by denying access.

The final facet, viewpoint, is either internal or external. An external viewpoint corresponds to functional or "black box" testing or Web penetration testing. Code reviews and intrusion detectors are prime examples of tools that work from an internal viewpoint.

Testing an SSA Tool Class

With a coherent taxonomy of software security assurance tools, we can rationally decide which classes of tools are most important, which to do first, and which to leave for later.

When we have chosen a particular class of tools to work on, we begin by writing a specification. The specification typically consists of an informal list of features, for quick orientation, then more formally worded requirements for features, both mandatory and optional. Specifications often include a glossary and a section with technical background, which gives a tutorial introduction.

For each tool class, we also recruit a focus group to review and advise on specifications. Group members are developers, academic researchers, and users. We depend on their expertise to make sure the specifications are widely acceptable.

While we are developing a specification, we also work on a test plan and test sets. What do current commercial and research tools of this class do? How will we test this kind of tool? This practical work helps us understand the specification. Once the focus groups review the specification and we incorporate public comment, we develop a test plan. A test plan details how a tool or technique is tested, how to interpret test results, and how to summarize or report tests. Most test plans require a test suite, which is a set of test cases. For example, code analyzers require a test suite of dozens or hundreds of large and small examples of source code with known flaws. The test suite also includes examples that are free of flaws to test for false alarms. Web penetration testers need executable applications with known flaws, like WebGoat. [11]

Currently we are developing a specification and test plan for source code analyzers. The first draft should be available at the Static Analysis Summit. [4] We are also developing a specification for Web application scanners.

A Standard Reference Dataset

While developing suites of tests, we collect much larger numbers of candidate test cases. This collection, the SAMATE Reference Dataset (SRD) [12], is freely accessible online. So far, we have collected more than 1,400 test cases, which academic researchers, tool developers, and tool evaluators can freely access to develop new methods and compare results. New test cases are constantly being added. The SRD allows anyone to search the test cases on a number of criterion, select any combination, and download them. Upon approval, researchers will be given accounts to contribute to the SRD. The SRD is a repository and clearing house for samples of designs, code, binaries, and other artifacts to accelerate research and development.

A single test case can have explanatory information associated with it, for instance, the author or contributor, the date submitted, language, which flaw(s) it exhibits, and a description. In addition, test cases may have directions on how to compile and link source code, input that triggers the flaw, or expected output. Users also will be able to add comments on a test case.

For historical stability, the content of test cases will never be updated. If the code in a test case needs to be fixed or improved, a new test case will be added, and the status of the existing test case will be changed to "deprecated." Deprecated status advises against using the case for any new work. A reference to the new test case will be added to the deprecated case. This way, a test report referring to a certain test suite can be rerun exactly, even years later. Although the metadata may be changed or comments added, the original test case won't be changed.

Future Challenges

Ultimately, these tests for classes of tools and techniques exist to help answer real questions. Is a program secure (enough)? How secure does technique *X* make a program? How much more secure does technique *X* make a

program after doing *Y* and *Z*? How much assurance does tool *T* give? Dollar for dollar, can I get more reliability from methodology *P* or methodology *S*?

We will work with others on developing and validating metrics and measures, not only for software and designs, but also for the tools themselves. Possible measurable qualities for tools and techniques are effectiveness (do they find important flaws), completeness (how many kinds of flaws can they find? Do they catch all of those kinds?), soundness (ratio of false alarms to real weaknesses found), report precision (location, severity, and type of flaw), and scalability and maximum size of artifact that can be handled. We would also like to characterize the ability of the user to trade completeness for soundness, add their own rules and style policies, and set a severity cut-off points.

Throughout our investigation, we will find gaps and opportunities in tools and techniques. We will write papers detailing these gaps and research opportunities. We will also propose requirements for research funding to develop new tools, do studies or experiments, or explore methods for assuring information. With more than a century of experience in measurement science and standards, NIST is uniquely qualified to conduct or collaborate in studies and experiments to improve the foundation of computer science and software assurance. ■

References

1. "SAMATE," *http://samate.nist.gov/* accessed 29 March 2006.
2. *Proceedings of Defining the State of the Art in Software Security Tools Workshop*, Paul E. Black (chair) and Elizabeth Fong (ed), NIST Special Publication 500-264, November 2005. Available at *http://hissa.nist.gov/~black/Papers/nistSP500-264_aug05.html*
3. *Proceedings of Workshop on Software Security Assurance Tools, Techniques, and Metrics*, Paul E. Black (chair), Michael Kass (co-chair), and Elizabeth Fong (ed), NIST Special Publication 500-265, February 2006. Available at *http://hissa.nist.gov/~black/Papers/nistSP500-265_nov05.html*
4. "Static Analysis Summit," *http://samate.nist.gov/SAS* accessed 29 March 2006.
5. John Viega, *CLASP Reference Guide: Volume 1.1 Training Manual, Secure Software*, McLean, VA, 2005.
6. Huaiqing Wang and Chen Wang, *Taxonomy of Security Considerations and Software Quality, Communications of the ACM*, 46(6):75-78, June 2003.
7. Katrina Tsipenyuk, Brian Chess, and Gary McGraw, *"Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors," Proceedings of Workshop on Software Security Assurance Tools, Techniques, and Metrics, op. cit.*, pp 36-43.
8. Michael Howard, David LeBlanc, and John Viega, *19 Deadly Sins of Software Security.* McGraw-Hill Osborne Media, July 2005.
9. "Common Weakness Enumeration," *http://cve.mitre.org/cwe/* accessed 31 March 2006.
10. "WebGoat Project," *http://www.owasp.org/software/webgoat.html* accessed 29 March 2006.
11. "SAMATE Reference Dataset," *http://samate.nist.gov/SRD/* accessed 29 March 2006.

About the Author

Dr. Paul E. Black | is a computer scientist at National Institute of Standards and Technology (NIST). Before joining NIST, he had nearly 20 years of industrial experience in software for integrated circuit design and verification, quality assurance, and business data processing. He earned a PhD from Brigham Young University in 1998. He has published in software testing, formal methods, software verification, and quantum computing, and has taught at Johns Hopkins University. He is a member of Association of Computing Machinery (ACM), the Institute of Electrical & Electronics Engineers (IEEE), and IEEE Computer Society.

Sensitive Data Anonymization

by Anas Abou El Kalam, and Yves Deswarte

Today, more and more applications use sensitive and personal information, and preserving citizens' privacy is becoming extremely important. Addressing this issue, this article suggests a rigorous approach to define data anonymization requirements, as well as ways to characterize, select, and build anonymizing solutions. This approach is illustrated by presenting a new generic procedure to anonymize personal data, while enabling an authorized data collector to control the linkability between multi-sourced data belonging to the same person.

Keywords: Privacy and security, collaborative environments, healthcare systems, anonymization, smartcards, inferences in databases (DB).

Introduction

To counter terrorist and criminal threats, recent regulations broadly expand security and surveillance, and these regulations may impact an organization's ability to protect an individual's privacy. We believe that it is necessary to enforce both security (to protect the system owner's assets) and privacy (to protect other people's personal data) and this article will demonstrate a solution that solves both issues.

Let us take the example of healthcare systems. Although networking facilitates data communication, it creates serious security risks. On the one hand, healthcare providers must reliably identify the patients and manage all the information they need to provide care to patients. On the other hand, exchanging and sharing healthcare data between various actors endangers the patient's privacy (*e.g.*, by enabling inference attacks on personal information). Indeed, the age, the sex and the month of discharge from hospital, are enough to identify the patient in a limited population; likewise, knowing two childbirth dates is enough to identify one woman as the mother in a sizeable population.

In this work, we first discuss examples of how the United States and some European countries anonymize medical data. Second, we present a systematic methodology that links privacy needs and adequate solutions. Then we propose a generic architecture that meets the privacy requirements. Finally, we give the details of our implementation.

Example of anonymization in the United States

In the United States, the Social Security Administration uses a "*Tricryption Engine*" (TE) to protect medical data. [1] The TE is a large encryption and automated key management system. It encrypts data with a per-call generated cryptographic key, encrypts the key and encrypts the link between the data and the key. The full process is the following:

- Sensitive data to be encrypted are selected by the user, and a request for encryption is sent to the TE.
- A randomly generated, symmetric session key is created and a random key ID is created.
- The session key is encrypted.
- The encrypted key and its key ID are stored in a key database (DB).
- The key ID is encrypted, producing a hidden link.
- The personal data are encrypted, using the session key.
- The encrypted data and the link are returned to the user.
- The encrypted data and the session key used to encrypt them are completely separated, both physically and logically, and the link between them is hidden.

The Tricryption Engine includes a fully integrated administration module, which supports the administration of the Tricryption system, including managing user authentication and authorization.

Example of anonymization in Switzerland

The Swiss Federal Office for Statistics (SFSO) is responsible for collecting the country's medical data. To preserve patient privacy, the SFSO is working with the Swiss Federal Section of Cryptography (SFSC). [2] The resulting analysis concludes that while it is not necessary to know the patient's identity within a medical record, it is imperative that SFSO has the capability link a patient's multiple medical records.

First, identifying data (such as date of birth, sex, last and first name) are replaced by a fingerprint, called anonymous linking code: *fingerprint = H (ID-Data)*, with *H* being a secure hash function. [3] Before transmitting medical data to the SFSO, the hospital generates a session key c; this key is then used to encrypt the fingerprint during the transmission: $IDEA\{fingerprint\}_c$; a public key cryptosystem (RSA) is used to transmit the session key $RSA\{c\}_E$ using the SFSO public key *E*.

After reception, "*c*" is retrieved by using the SFSO private key *D*; the encrypted fingerprints are then decrypted, and uniformly re-encrypted by the symmetric key *K* of the SFSO. They become the anonymous linking codes used as personal codes. The key *K* is distributed among several trusted persons, using Shamir's secret sharing technique.

Example of anonymization in Germany

The German National Cancer Registry gathers medical statistics related to German cancer cases. The procedure of the population-based cancer registration is realized in two steps by two institutions. [4] In the first stage, the Trusted Site accumulates the data recorded by doctors. The Trusted Site anonymizes these data by an asymmetric procedure, in this case, a hybrid IDEA-RSA encoding. The identifying data is encrypted with an IDEA session key, generated randomly. The IDEA key is encoded by a public RSA key. A control number (a pseudonym) is then generated by using a one-way hash function on various identifying attributes and a symmetrical ciphering algorithm (IDEA).

To allow data coming from the different federal lander to be linked, the pseudonym generation procedure and the key are unique ("Linkage Format"). The Trusted Site transfers both the encrypted patient-identifying data and the epidemiological plaintext data to the Registry Site. The latter stores the record in the register database and brings together different records belonging to the same patient. After this matching of data, a random number is added to the pseudonym and the result is symmetrically encrypted by IDEA ("Storage Format"). To match new records, the control numbers must be deciphered back from the "Storage Format" to the "Linkage Format."

Example of anonymization in France

French hospitals [5] transform the patient's identity by using a one-way hash function *H*. Actually, two keys have been added before applying *H*. The first pad, *k1*, is used by all senders of information: "$Code_1=H\ (k_1|Identity)$", and *k2* used by the recipient: "$H\ (k_2 \mid Code_1)$". The aim of *k1* (resp. *k2*) is to prevent dictionary attacks by a recipient (resp. a sender).

However, this protocol seems both complex and risky: the secret key should be the same for all information issuers and remains the same over time. Moreover, these keys must always remain secret: if a key is compromised, the security level is considerably reduced. It is very difficult to keep a key that is widely distributed secret for a long time. Hence, new keys must be generated and distributed periodically. The same applies when the algorithm (or the key length) is proven not sufficiently robust any more. But, how can we link all the information concerning the same patient before and after changing the algorithm/key? If this problem occurs, the only possible solution is to apply another cryptographic transformation to the entire database, which may be very costly.

The characteristics of these four examples are summarized in Table 1.

Country	Purpose	Technique
USA	Social security data processing	Secret keys (Tricryption)
Germany	Statistics	Hybrid encryption + hashing
Switzerland		
France	Linking medical data for evaluation purposes	Symmetric keys + hashing

Table 1 Summary of existing solutions.

Analytic approach

Most of the solutions presented above have been developed empirically and concern only one specific use. In this work, we develop a *generic solution* that could be easily adapted (and parameterized) to satisfy the requirements of a particular system. For this reason, we believe that before calling for technical or organizational solutions, we should first define a systematic methodology.

The starting step is to understand the standardized terminology used in this area. [6] *Anonymity* can be defined as the state of being not identifiable within a set of subjects. *Pseudonymity* adds accountability to anonymity. *Unlinkability* between items (*e.g.*, users initiating operations, patient's data) means that it is not possible to distinguish if these items are related or not. *Unobservability* ensures that a user may use a resource (or service) without others being able to observe that the resource/service is being used.

Traditionally, the privacy analysis process studies the request (needs to be satisfied) and suggests a *response* (mechanisms to implement). To be systematic and to avoid attacks by inferences, our methodology suggests some intermediary steps: identify the privacy *objectives* and specify the privacy *requirements*.

Indeed, the *privacy needs* represent the user's expectations; they depend on the system, the environment, *etc.*; and generally, their form is neither very explicit nor very simple to formalize. Thus, a great effort should be done to express the *needs* clearly. For instance, in healthcare systems, it is important to identify directly and indirectly nominative data that are to be anonymized. For example, for a particular study, instead of providing the full address (resp. the profession), is it sufficient to provide the region code (resp. the socio-professional category)? This kind of question will naturally lead us to study what we call the privacy objectives.

We define the *privacy objective* according to one of the three following properties, applied to the anonymization function:

- **Reversibility**—hiding data by encryption. In this case, from encrypted data, it is always possible to retrieve the corresponding original nominative data.
- **Irreversibility**—the property of anonymization. The typical example is a one-way hash function. Once replaced by anonymous codes, the original nominative data are no longer recoverable.
- **Inversibility**—this is the case where it is, in practice, impossible to re-identify the person, except by applying an exceptional procedure restricted to duly authorized users. This exceptional procedure must be done under surveillance of a highly trusted authority like the medical examiner, the inspector-doctor or a trustworthy advisory committee. This authority can be seen as the privacy guarantor. Actually, it is a matter of a pseudonymisation according to the common criteria terminology. [7]

Afterwards, the analysis is carried on by studying the privacy requirements, taking into account the possible attacks, the environment, *etc.* We identify two kinds of privacy requirements: the "*linkability*" and the "*robustness*" requirements.

Linkability allows associating (in time and in space) one or several pseudonyms to the same person. Linkability can be temporal (*e.g.*, always, sometimes, never) or geographic (*e.g.*, international, national, local).

The robustness requirements concern illicit disanonymization. We distinguish robustness to reversion from robustness to inference. The *reversion robustness* concerns the possibility of inverting the anonymization function, for example if the used cryptographic algorithms are not strong enough. The *inference robustness* concerns data disanonymization by means of unauthorized computation, *e.g.*, by inference.

Once we have specified the privacy requirements, it is time to choose and characterize the most suitable solutions. In particular, we have to specify the *type* of solution to develop (organizational procedure, cryptographic algorithm, one-way function); the *plurality* of the solution to implement (simple, double or multi- anonymization); and the *interoperability* of the solutions that must be combined: transcoding (manually) translating (mathematically) or transposing (automatically) several anonymization systems into another one.

A Generic Solution

Even if the suggested anonymization procedure (see figure 1) is discussed through healthcare examples, it is adaptable to non-medical areas, such as demographic studies or E-commerce.

Transformations carried out by hospitals

In hospitals, three kinds of databases can be distinguished: administrative, medical, and anonymized databases. Each anonymized database contains the information for a particular project. A

The solution that we propose offers several benefits. First, it is fine-grained, generic enough and easily adaptable to different sector needs.

project is a program or a study intended for statistical, epidemiological, therapeutic research, or medico-economical data processing.

The transition from a medical database to an anonymized one requires the application of two transformations (*T1, T2*).

T1 consists of calculating "*IDApat|Proj*," an anonymous identifier per person and per project. "*IDproj*" is the project identifier, while "*IDpat*" is the permanent patient anonymous identifier (a random number). We suggest that *IDpat* is held under the patient's control, *e.g.*, on his personal medical smart card.

In the hospital, when transfering data into anonymous databases, the user (*i.e.*, the healthcare professional) sends IDproj to the card. The card already contains *IDpat*. By supplying his card, the patient gives his consent for his data to be exploited as part of this project. The *T1* procedure, run within the smart card, consists of applying a one-way hash function (*e.g.*, SHA-2) to the concatenated set (*IDproj* | *IDpat*):

(T1) $IDApat|Proj = H(IDproj \mid IDpat)$

Nevertheless, the transformation *T1* does not protect against attacks where attackers try to link data held by two different hospitals. To make this clearer, let us take an example where Paul has been treated in the hospitals $Hosp_A$ and $Hosp_B$. In each of these two hospitals, Paul has consented to give his data to the project $Proj_a$. Let us assume that Bob, a $Hosp_B$ employee, knows that the fingerprint X (=$IDAPaul|Proj_a$) corresponds to Paul, and that Bob obtains (illicitly) access to the anonymous database held by $Hosp_A$ and concerning $Proj_a$. In this case, the malicious user Bob can easily establish the link between Paul and his medical data (concerning $Proj_a$) held by $Hosp_A$ and $Hosp_B$.

To face this type of attacks, a cryptographic asymmetric transformation (**T2**) is added. Thus, before setting up the anonymous databases (specific to each project), the hospital encrypts (using an asymmetric cipher) the fingerprint *IDApat|Proj* with the encryption key Ks_{hosp} specific to the hospital; (the notation "{M}K" indicates that M is encrypted with key K):

(**T2**) $IDAhosp(pat|Proj) = \{IDApat|Proj\}Ks_{hosp}$

If we take again the previous scenario, the malicious user Bob cannot re-identify the patients because he does not know the decryption key Kp_A. Ks_{hosp} and Kp_{hosp} are a key pair of a public key cryptosystem, but that does not mean that Kp_{hosp} is really public. It is known only by the project processing centers and by the hospital's security officer (who knows also Ks_{hosp}, of course).

Basically, the anonymous databases intended to one or several projects are periodically sent by hospitals to

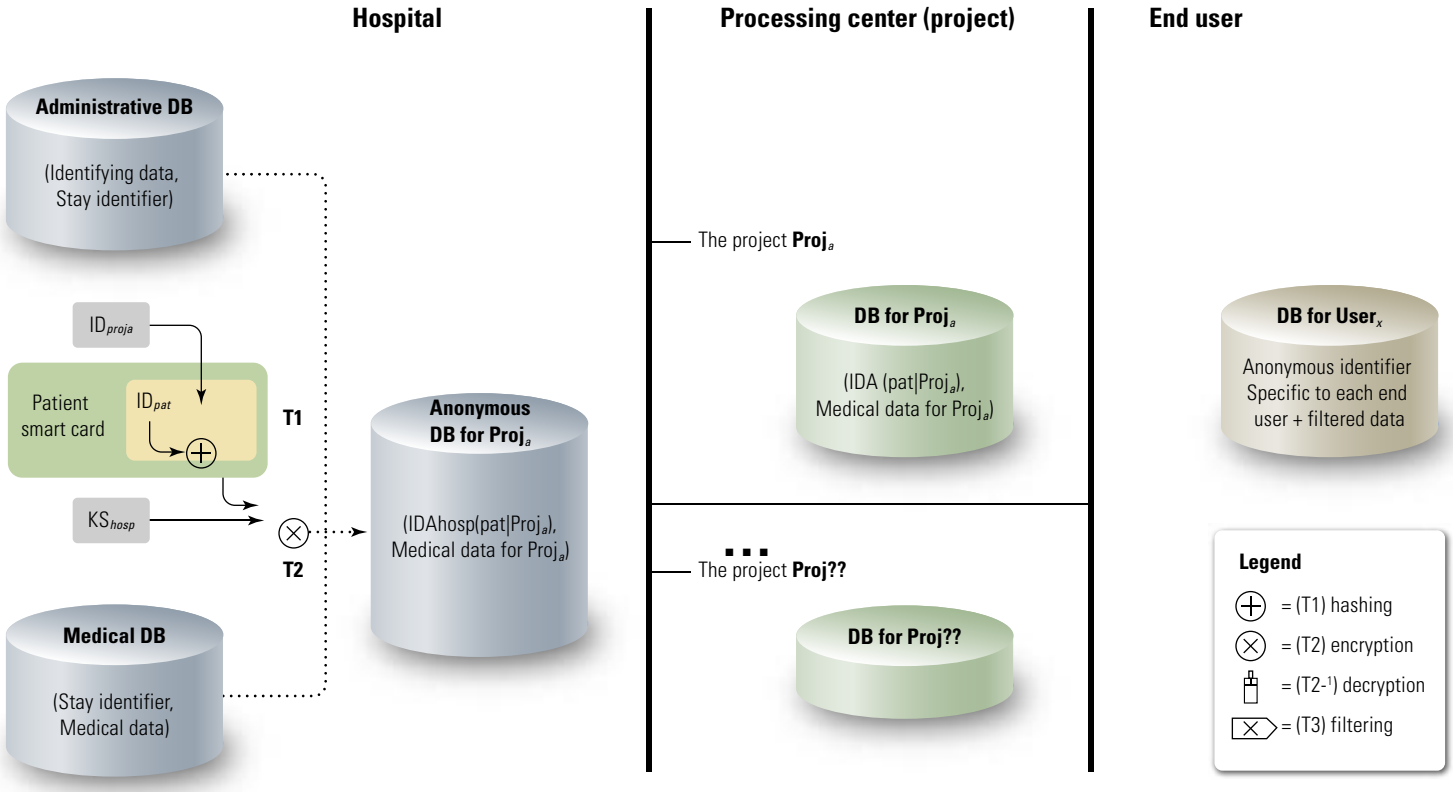


Figure 1 The suggested anonymization procedure.

Second, using smartcards (that are sufficiently tamper-resistant) helps to keep the sensitive data (*e.g.*, the anonymous patient identifier) secret and to protect the critical processes. Moreover, the secret as well as the anonymization is held under the patient's control.

processing centers. A processing center could be an association, an office for medical statistics, or a research center.

Transformations carried out by processing centers

When anonymized data are received from a hospital by a processing center, these data undergo transformations that depend on *IDAproj|pat* and on Ks_{hosp}. Every center decrypts received data by using Kp_{hosp}:

$$[IDAhosp(pat|Proj)]Kp_{hosp} = [\{IDApat|Proj\}Ks_{hosp}]Kp_{hosp}$$

$$= IDApat|Proj \quad (\mathbf{T3})$$

Note that since the resulting data are associated to *IDApat|Proj*, each project can link data corresponding to the same patient, even if they come from different hospitals.

Before their distribution to the final users (statistics organizations, web publishing, press, *etc.*), the anonymized data can undergo a targeted filtering (data aggregation, data impoverishment, *etc.*). If, in addition, the security objective is to forbid file linking, it is advisable to apply another anonymization (*e.g.*, by MD5) with a secret key *Kutil|proj* generated randomly:

$$IDApat|util = H(IDApat|Proj \mid Kutil|proj)$$

In accordance with the needs, this transformation can provide different linkability properties:

- If the aim is to allow full-time linking, *Kutil|proj* has to be stored by the processing center and reused for each distribution to the same user.
- Conversely, if the center wishes to forbid the user to link data distributed by the center at different times, the key is randomly generated just before each distribution.

Implementation

For the implementation, it is advisable to use cards that support cryptographic procedures (smartcards). Kits provided by smartcard manufacturers (*e.g.*, the "JCardManager" interface provided in the Gemexpresso RAD III kit) offer a complete environment for smartcard software development and contain an interface that makes the communication easier with the smart card. With such a kit, programming is done in the standardized language "*Javacard*." Javacard is a reduced-API Java. Even if this API provides most characteristics of Java (*e.g.*, exceptions, constructors, inheritance, unidimensional arrays), it has some restrictions, since the primitive types are limited to bytes, shorts, and Booleans; it does not support cloning, threads, garbage collection, *etc.*

Conclusion

The solution that we propose offers several benefits. First, it is fine-grained, generic enough and easily adaptable to different sector needs. Second, using smartcards (that are sufficiently tamper-resistant) helps to keep the sensitive data (*e.g.*, the anonymous patient identifier) secret and to protect the critical processes. Moreover, the secret as well as the anonymization is held under the patient's control. Indeed, the personal data can appear in a particular database only if, by supplying his card, the patient gives his consent to use his sensitive data as part of a project. Third, the solution resists dictionary attacks that could be run in various organizations: hospitals, processing centers, or end users. Fourth, the identifiers are located in different places and the keys are held by different persons. Thus, even if a certain *IDpat* (corresponding to Paul, for example) is disclosed, only Paul's privacy would be endangered (but not all the patients' privacy, as it is the case in the French procedure, for instance).

Furthermore, according to the security needs of the studied cases, we suggest to complement our solution by other technical and organizational procedures. In particular, the access to data has to be strictly controlled; a well-defined security policy must be implemented by appropriate security mechanisms (hardware and/or software). Reference [8] suggests a security policy and an access control model (*Or-BAC: Organization-Based Access Control*) that are suitable to collaborative systems. ■

Acknowledgements

This study has been partially supported by the project MP6 of the French National Research Network on Telecommunications (RNRT) and by the project Privacy and Identity Management for Europe (PRIME) of the European Programme FP6/IST.

References

1. Vasic O., Using Tricryption to Protect Privacy, Tech. Report ERUCES Data Security, 11 February 2005, available at <http://www.eruces.com/images/stories/pdf/ERUCES_EFE_Protecting_Privacy_%20Final.pdf>.
2. Jeanneret J.P., Jacquet-Chiffelle D.O., "How to Protect Patient's Rigottes to Medical Secret in Official Statistics," Information Security Solutions Europe Conference (ISSE), London, UK, September 2001, pp. 26-28.
3. A one way hash function, also known as a message digest function, fingerprint function or compression function, is a mathematical function that takes a variable-length input string and converts it into a fixed-length binary sequence. Furthermore, a secure one-way hash function is designed in such a way that it is hard to find a string that hashes to a given value, or to find two strings that produce the same hash value.
4. Blobel B, "Clinical Record Systems in Oncology," in Personal Medical Information – Security, Engineering and Ethics, Anderson R.J. (Editor), Springer-Verlag, pp. 39–56, 1997.
5. Quantin C., Bouzelat H., Allaert F.A., Benhamiche A.M., Faivre J. and Dusserre L., "How to ensure data security of an epidemiological follow-up," International Journal of Medical Informatics, vol. 49, No 1, 1998, pp 117-122.
6. Pfitzmann A., Köhntopp K., "Anonimity, Unobservability, and Pseudonymity – A Proposal for Terminology," Int. Workshop on Design Issues in Anonymity and Unobservability, Berkley, USA, July 25-26, 2000, Springer.
7. International Standard ISO/IEC 15408-2, Information technology — Security techniques — Information criteria for IT security — Part 2: Security functional requirements, 248 p., 2nd edition, October 2005.
8. Abou El Kalam A., Balbiani P., Benferhat S., Cuppens F., Deswarte Y., Saurel C., Trouessin G. "OrBAC," 4th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy'03), Como, Italy, 4-6 June 2003, IEEE Computer Society Press, pp. 120-131.

About the Author

Anas Abou El Kalam | is an assistant professor at the Bourges Higher National School of Engineering. He is in charge of the security option and head of the Computer Science Department. His current research interests are in privacy, intrusion detection systems, smart cards security, security policies, and models. He obtained his PhD in December 2003 at INP (National Polytechnic school) and LAAS-CNRS (Laboratory for Analysis and Architecture of Systems).

Yves Deswarte | is Directeur de Recherche at LAAS-CNRS in Toulouse, France. His research interests are in distributed system security and fault-tolerance. He has authored or co-authored more than 80 international publications in these areas. He has been consulting for several European companies, and for SRI International in the USA. He is member of Institute of Electrical & Electronics Engineers (IEEE), ACM and Society of Electricity, Electronics, and Technologies of Information and Communication (SEE). He is the IEEE-CS representative at IFIP TC-11 (Technical Committee on Security and Protection in Information Processing Systems).

Removing Security through Obscurity from Software Watermarking

by Christopher O. Jones, Robert F. Mills, and Richard A. Raines

The research leading to this paper was sponsored by the Air Force Research Laboratory Anti-Tamper/Software Protection Office.

Software piracy, the act of illegally copying and distributing software, is an ongoing problem for software vendors within the United States and internationally. Recent estimates show that piracy costs software vendors nearly $15 billion dollars each year. [1] Therefore, security professionals are presently researching various software watermarking techniques as one of many means to be used as tools to combat software piracy. Software watermarking involves inserting some sort of copyright data, or in some cases, a unique fingerprint, often taking the form of a customer's ID number. This data is then embedded within the software's code.

Current watermarking systems protect the watermark by hiding it, using security through obscurity. In this paper we propose to use watermarking for Digital Rights Management (DRM), removing security through obscurity from software watermarking and using cryptographic style security instead, and introduce a new software watermarking method using those criteria. By "cryptographic style security" we mean that we are not attempting to hide that the watermark is present. Rather, we put our emphasis on making the watermark as robust and unbreakable as possible.

Using Watermarking for Digital Rights Management

Software watermarking provides two key services for DRM. First, it proves ownership by implanting a copyright into the software, and second, it provides a means to track individual copies of software, thereby discouraging theft and unauthorized distribution.

Proof of ownership

The watermark embedded in a piece of software can serve as proof of ownership. A valid watermarking system *proves* ownership rather than simply identifying the author. If the attacker knows that the author has the ability to prove ownership for a piece of software, an attacker is less likely to claim it as his own. Current research focuses on mobile code, specifically *java* bytecode, which is widely distributed. It is a simple matter to disassemble/decompile the bytecode. [2] Reusable code modules are a common practice in modern software engineering. These modules can be stolen from online *Java*-based applications and then used in the attackers' software. Proving ownership allows the author to keep attackers from claiming stolen code modules as their own.

This proof of ownership is similar to the copyright notice found in printed materials, but in a more robust form. While the printed copyright can be physically clipped out of a book, the book may still be usable. However, if a software watermark has been embedded appropriately, unauthorized removal or modification of the watermark should render the program useless.

Software Fingerprinting

Software fingerprinting is a form of watermarking in which multiple copies of the same piece of software are embedded with separate watermarks, uniquely marking each copy of the software. The watermark not only demonstrates proof of ownership, but also serves as a way to trace the origin of a copy of the software once stolen. For example, suppose a company releases 100 copies of a beta version of a new product. Each copy of the software can be fingerprinted with a unique watermark. If a copy of the software is stolen and illegally distributed, not only can proof of ownership be determined, but the software's fingerprint can be extracted to find the software's point of origin. This allows the theft's source to be identified as well.

The motion picture industry has begun to take advantage of media fingerprinting as a means to curb piracy of movies. The Motion Picture Association of America recently banned screener copies of movies from being sent to critics and reviewers. It had been shown that these screener copies are stolen and copied (a relatively easy matter for digital media) by the critics or their assistants leading to the films being put on various peer-to-peer networks on the Internet. A secure media fingerprinting method was used to identify uniquely each copy of *Kill Bill: Vol 1* and *Seabiscuit* that were sent to critics. When they were inevi-

tably pirated over the Internet, their watermarks were extracted. Investigators knew which copies had been stolen and were able to track the theft of the movies leading to the prosecution of Russell Sprague, who received the movies from actor Carmine Caridi, a member of the Academy of Motion Picture Arts and Sciences. [3]

Software vendors could take a similar approach when releasing software. Each copy of the software would be embedded with its own unique watermark. The watermark could contain specific user information or the vendor could maintain a database listing the user information and the assigned watermarks. If pirated copies of the software are made available, it would be possible to track the origin of the theft and prosecute the guilty parties.

Threat Model

Software watermarking attacks can be divided into four categories: *subtractive, distortive, additive,* and *collusive*. A subtractive attack attempts to remove the watermark completely from the program, while maintaining functionality. A distortive attack is one in which an attacker performs numerous transformations to the program until a watermark has been corrupted without distorting the program's functionality. A common distortive attack is achieved when an attacker performs various code obfuscation techniques to a watermarked application. An additive attack attempts to add another watermark to the program, making it impossible to prove which watermark was added to the program first.

Collusive attacks are unique to software fingerprinting and can be used to detect and remove fingerprints. In a collusion attack, an attacker compares two copies of the same program, each with a separate, unique fingerprint. A differential analysis can be used to locate within the code or executable where the fingerprint is stored and will allow an attacker to remove or perhaps alter a watermark. [4]

These attacks are classified using two threat models, manual and automated attacks. A manual attack is defined as a determined attack by a human reverse engineer for an extensive time. Automated attacks are performed by automated tools that are effectively used against entire classes of watermarks. Most research has focused on defeating automated attacks, conceding that "no software protection scheme will withstand a determined manual attack." [5]

An accepted standard for a watermark attack resistance is that any removal or modification of the watermark will "break" the program, rendering it unusable. Embedding the watermark should not change the output of the program. Further, the effort required to attack and break the watermark should exceed the effort required for independent development of the same software.

Security through Obscurity

Security through obscurity refers to the axiom of assuming that a potential attacker won't have knowledge about the algorithm or key used.

History of Security through Obscurity

Security through obscurity was first debunked in cryptography by Auguste Kerckhoffs in *La Cryptographie Militaire* in 1883: "The security of a cryptosystem must not depend on keeping secret the crypto-algorithm. The security depends only on keeping secret the key." [6] Many crypto-systems are trivial to decode, once the method of encryption is known.

Instead of relying on the security of the algorithm itself, modern cryptographic algorithms depend in the security of the encryption keys and the rigorous mathematical difficulty of reverse engineering one way functions for the security of those keys. Little or no effort is expended in hiding that encryption occurs or in hiding actual method of encryption. The important thing is to keep the *keys* secure.

The prevalence of open source software takes this argument further, claiming that open source software is necessary to produce truly secure applications. By releasing the source code, there is no secrecy. Everyone has access to the software, and all known vulnerabilities are made public. Users are therefore aware of any risk or threat, even if the necessary patch is not yet available.

Current Software Watermarking Security

Current watermark research uses *stealth,*

data rate, and *resilience* to *attack* as metrics to measure the strength of software watermarking techniques. Stealth refers to how well hidden the watermark is within the program. Data rate relates the amount of data that can be embedded in a watermark to program size. Resilience refers to how well the watermark resists attack. Much like the engineer's conundrum (trade-off between quality, speed, and cost), software watermarks "exhibit a trade-off between these three metrics in that a high data rate implies low stealth and resilience." [7] Increasing the strength of one metric results in a decrease in the others.

This emphasis on the stealth of the watermark is made at the cost of its possible resilience to attack. The current state-of-the-art in software watermarking is Collberg and Thomborson's dynamic graph-based watermarking algorithm. [7] Much of the current research in software protection (watermarking, obfuscation, and tamper-resistant code) has been combined into a single tool, SandMark. [8] Developed at the University of Arizona for studying software watermark, SandMark performs software watermarking, tamper-proofing, and code obfuscation of *Java* bytecode. It implements several static and dynamic watermarking techniques, along with code obfuscation and optimization algorithms that can be used as automated attacks to verify, validate, and compare the watermarking techniques.

As noted in [7], software watermarking is achieved by embedding a structure *w* into a program *P* as follows. Four functions describing the operations of embedding, extracting, recognizing, and attacking the watermarks are given as follows:

- $embed(P,w,key) \rightarrow P_w$
- $extract(P_w, key) \rightarrow w$
- $recognize(P_w, key, w) \rightarrow [0.0,1.0]$
- $attack(P_w) \rightarrow P'_w$

It is critical that *w* can be reliably located and extracted even after *P* has been subjected to code transformations (such as translation, optimization and obfuscation). Further, *w* should be stealthy, should allow for a high data rate, and its presence in *P* should not adversely affect the performance of *P.*

Most important, *w* has a mathematical property that allows the author to prove ownership. [7] Some watermarking algorithms use a key, which is required to insert or recognize the watermark. The Collberg-Thomborson algorithm encodes the watermark in graph structures that are hidden in the program's heap space during runtime. The watermark is so well hidden that the *recognized* function can only give the probability that the watermark is present, as opposed to other methods using an *extracted* function that is able to give a yes or no Boolean response to the presence of the watermark.

A secret key is needed to embed or recognize the watermark. If the key is published, security is compromised. Current watermarking systems depend on the location of the watermark to be hidden from any potential attacks. Once the location of the watermark is known, removing or modifying it is a minor task. We see placing too much emphasis on hiding the watermark, rather than making it robust or tamperproof, as a major security vulnerability.

Using Cryptographic Style Security for Software Watermarking

We argue that the watermarking scheme and the watermark values should be publishable without compromising the security of the watermark. A model similar to modern cryptography is desirable. The watermarking algorithm can be published, even the actual watermark. The security of the watermark is then dependent on the robustness of the embedding of the watermark and the difficulty of its removal while maintaining program integrity, not finding the watermark.

Integrate the Watermark with Program Control Flow

The watermark must be integrated into the program in such a manner that any attempt to remove or modify the watermark will break the program. The watermark must be easy to embed, but difficult to remove or modify. Most watermarking schemes integrate the watermark with a vital part of the software.

Attempts to remove or modify the watermark will modify this vital part of the software, breaking the program. A successful attack must leave the program flow intact. The control flow for a program is a vital component of the program. Modifications to control flow break the program, either causing it to crash, exit suddenly, function improperly, or enter an infinite loop.

Branch Flattening

Branch flattening takes the control flow of the program and flattens it, transforming the program into a large switch/case statement. The switch sends program control flow to the appropriate code block using a go-to pointer. Each code block ends by updating the case variable and sending control flow back to the switch. This process is shown in Figure 1.

Branch flattening has been studied as an obfuscation technique to obstruct static analysis of programs. [9] Obfuscating the control flow prevents the attacker from being able to modify the program control, specifically once the watermark has been embedded. Branch flattening was also used as one of the first static software watermarking techniques by Davidson and Myhrvold. [10] They proposed to encode a watermark based on the ordering of the case statements. This watermark is based on the ordering and structure of the code, but it has a low resilience to attack, since a simple reordering of the case statements modifies the watermark with no adverse affect on the program.

Strong Watermarking Variants

A stronger method of protection involves embedding the watermark into the case statements of a program after branch flattening has been performed. The watermark is not hidden, but any attempt to modify or remove it will disrupt the program control flow, thereby breaking the program. This method is further described by Wilson and Sattler in. [11]

Researchers at the Air Force Institute of Technology have developed a proof-

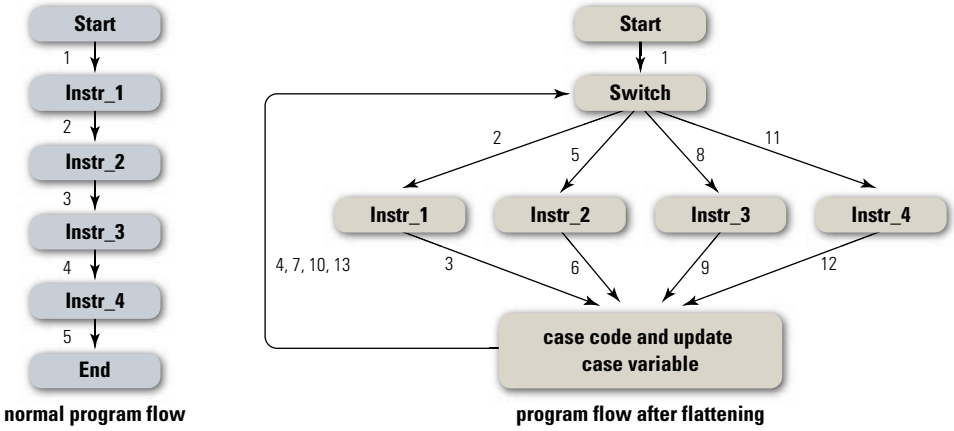


Figure 1 Program Flow after Flattening

of-concept tool to implement the Wilson-Sattler Algorithm. The program is written in *Java*, but operates on c source code. It operates on branch flattened c code, embedding a watermark into the case statements as described above. Our proof-of-concept tool supports the following methods of software fingerprinting:

- **Basic Substitution**—The basic method uses simple substitution to implant the watermark. The watermark values are substituted in for the case values. As the case values are evaluated in random order, the watermark is placed in random cases.
- **A+B Substitution**—The mathematical operation C = A + B is used to update the case variable, where C is the case value and A is a watermark value.
- **Polynomial Substitution**—The polynomial method uses a polynomial for the update operation with six terms. The watermark is embedded within the polynomial. The terms of the polynomial consist of a coefficient, base, and exponent. Each case variable can have multiple unique updates and multiple functions can be used to update the same case variable.
- **Pointer Aliasing**—A global integer array is declared in the C code where the operands for the case variable update operation are stored. A simple addition operation is used for the case variable update, using operands from the global integer array.
- **Case statements versus if statements**—A switch block is equivalent to a series of if statements. By converting the case statements in the switch block to if statements, the case update values can be variables. This allows more complex additions to be made to increase watermark resilience.
- **Conjunctive Normal Form**—Conjunctive normal form, or CNF, is a Boolean expression expressed as a series of clauses ANDed together. Each clause is made up of a series of one or more literals ORed together. The expression is a conjunction of disjunctions of literals. A literal is a variable or the negation of a variable. In *k*-CNF, each clause in the expression has *k* literals.

Preliminary analysis shows that the methods using pointer aliasing, if statements, and conjunctive normal form are highly robust, but they introduce measurable cost in program run time and size. Further research needs to be done on increasing robustness against manual runtime analysis, and combining the tool with existing software protection measures, code modification and anti-debugger protection for example.

Conclusion

History has shown that security through obscurity does not provide security. Current watermarking research depends on a secret key for security, hiding the watermark for protection. If the key is discovered and the watermark found, it is by default vulnerable. On the contrary, we have argued that the focus of the research should nonstealthy, robust techniques. This shift would lead to increased robustness and security in future software watermarking methods. The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.■

References

1. T. R. Sahoo and C. Collberg, "Software watermarking in the frequency domain: Implementation, analysis, and attacks." Technical Report TR04-07, 2004.
2. J. Palsberg, S. Krishnaswamy, M. Kwon, D. Ma, Q. Shao, and Y. Zhang, "Experience with software watermarking," Department of Computer Science, Purdue University, 2000. [Online]. Available: *http://www.cs.ucla.edu/ palsberg/paper/acsac00.pdf*
3. "Man pleads guilty to film piracy," BBC News World Edition, April 2003. [Online]. Available: *http://news.bbc.co.uk/2/hi/entertainment/3621293.stm*
4 D. Boneh and J. Shaw, "Collusion secure fingerprinting for digital data," Transactions on Information Theory, IEEE, pp. 1897–1905.
5 C. Collberg and C. Thomborson, "Watermarking, tamperproofing, and obfuscation - tools for software protection," IEEE Transactions on Software Engineering, August 2002.
6 A. Kerckhoffs, "translated from la cryptographie militaire," January 1883. [Online]. Available: *http://petitcolas.net/fabien/kerckhoffs/*
7. C. Collberg, C. Thomborson, and G. Townsend, "Dynamic graph-based software watermarking," Technical Report TR04-08, April 2004. [Online]. Available: *ftp://ftp.cs.arizona.edu/reports/2004/TR04-08.pdf*
8. "Sandmark: A tool for the study of software protection algorithms," Department of Computer Science, The University of Arizona. [Online]. Available: *http://www.cs.arizona.edu/sandmark/*
9. C. Wang, J. Hill, J. Knight, and J. Davidson, "Software tamper resistance: Obstructing static analysis of programs," Department of Computer Science, University of Virginia, May 2000. [Online]. Available: *http://www.cs.virginia.edu/ techrep/CS-2000-12.pdf*
10. R. Davidson and N. Myhrvold, "Method and system for generating and auditing a signature for a computer program," US Patent, no. 5559884, Assignee: Microsoft Corporation, September 1996.
11. K. Wilson and J. Sattler, "Software control flow watermarking," US Patent Application, Assignee: AFRL/SNT Software Protection Center, 2003.

▷▷ *Authors Bio page 28*

Vulnerability Analysis of J2ME CLDC Security

by Mourad Debbabi, Mohamed Saleh, Chamseddine Talhi, and Sami Zhioua

The research leading to this paper was possible thanks to funding and scientific collaboration with Alcatel Research and Innovation (R&I) Security Group.

Java 2 Micro-Edition (J2ME) Connected Limited Device Configuration (CLDC) is the platform of choice when it comes to running mobile applications on resource-constrained devices (cell phones, set-top boxes, etc.). The intent of this paper is twofold: First, we study the security architecture of J2ME CLDC. Second, we provide a vulnerability analysis of this Java platform. The analyzed components are Virtual machine, CLDC API and MIDP (Mobile Information Device Profile) API.

With the proliferation of mobile, wireless and Internet-enabled devices (*e.g.*, personal digital assistant (PDA), cell phones, pagers, *etc.*), *Java* is emerging as a standard execution environment because of its security, portability, mobility and network support features. The platform of choice in this setting is J2ME CLDC. [6, 12]

The typical most widely deployed J2ME CLDC platform consists of several components that can be classified into virtual machine, APIs and tools. The virtual machine is the Kilobyte Virtual Machine (KVM). [7, 13] The APIs are CLDC [17] and MIDP. [10, 12] The tools are the pre-verifier and the *Java* code compacter. KVM is an implementation of the *Java* Virtual Machine (JVM). [4] CLDC provides the most basic set of libraries and virtual-machine features for resource-constrained, network-connected devices. MIDP is a layer on top of CLDC configuration. It extends the latter with more specific capabilities, namely, networking, graphics, security, application management, and persistent storage. The preverifier checks all the *Java* classes to enforce object, stack, and control-flow safety. This is done offline and the result is stored as attributes in the compiled program units. The *Java* code compactor (JCC) is in charge of the romizing process. The latter is a feature of KVM that allows it to load and link *Java* classes at startup. The idea is to link these classes offline, then create an image of these classes in a file, and finally to link the image with KVM.

With the large number of applications that could be available for *Java*-enabled devices, security is of paramount importance. We present here a careful study of J2ME CLDC security aspects with the purpose of providing a security evaluation for this *Java* platform.

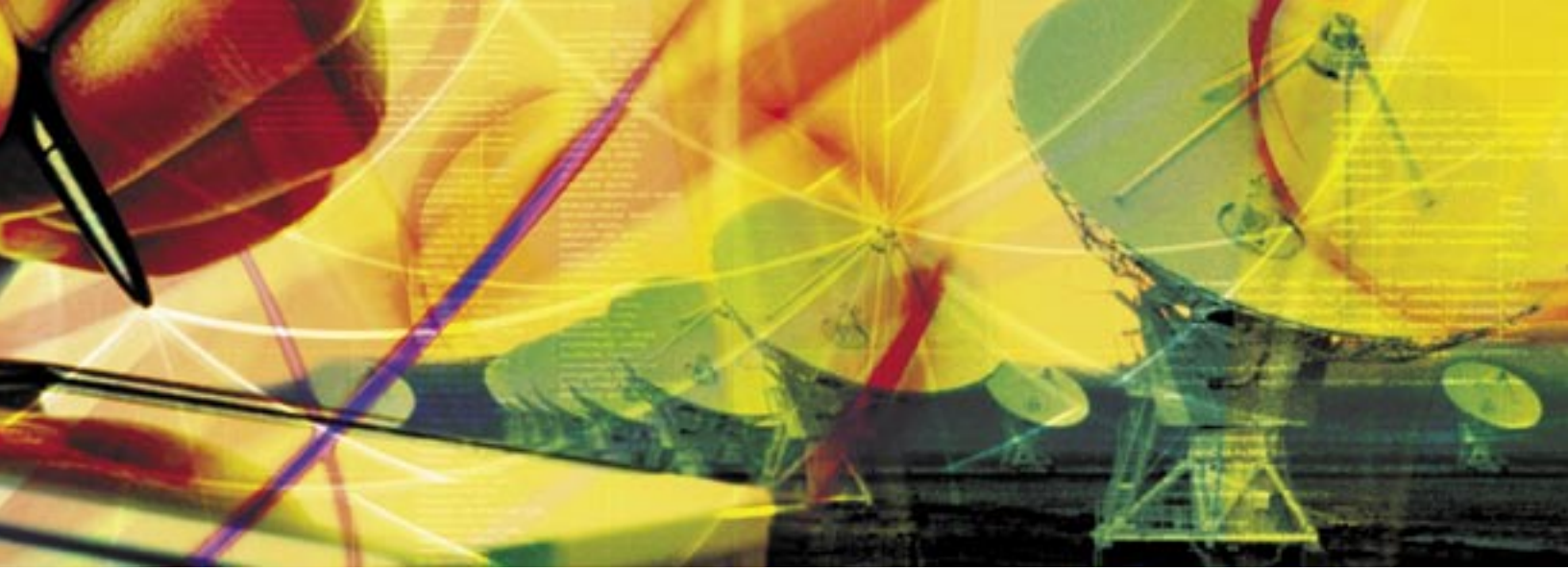
In section 2, we present the main security architecture of J2ME CLDC. In section 3, we list the results of the vulnerability analysis by starting with the previously reported flaws. Finally, section 4 concludes the paper.

J2ME CLDC Security Architecture

Applications developed for the J2ME CLDC platform are called **MIDlets**. They are downloaded to the device in the form of two files: the *Java* Archive (**JAR**), and the *Java* Application Descriptor (**JAD**). The JAR is an archive file that contains class files and any other supporting files. The JAD, on the other hand, is a text file that contains several attributes like the MIDlet name and MIDP version needed to run the MIDlet. The software entity on the device that is responsible for MIDlet management, such as downloading, installing, running, *etc.*, is called the **Application Management System** (AMS), or the *Java* **Application Manager** (JAM).

The security of the J2ME CLDC platform can be categorized into *low-level* security, *application* security, and *end-to-end* security:

- In general, the role of the low-level security mechanisms is to ensure that class files loaded into the virtual machine do not execute in any way that is not allowed by the *Java* virtual machine specification. [5]
- Application-level security means that "*Java* applications can access only those libraries, system resources, and other components that the device and the *Java* application environment allow it to access." [17]
- End-to-end security has a larger scope involving secure networking.



In J2ME CLDC platform, low-level and application security are addressed in CLDC, while MIDP addresses *application* and *end-to-end* security.

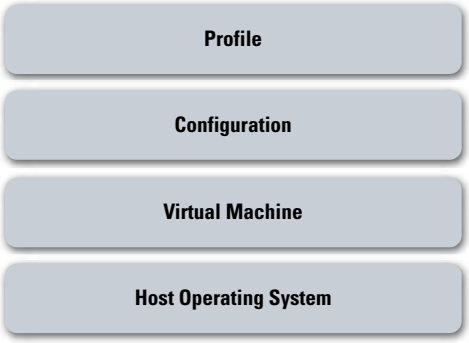


Figure 1 High-Level J2ME CLDC Architecture

CLDC Security

Here, we discuss low-level and application security.

Low-Level Security

Low-level security in CLDC is mainly based on type safety mechanisms. Since conventional class file verification is not ideal for resource-constrained devices, class files are *pre-verified* on the development platform, before being installed on the device. The on-device verifier then performs only a linear scan of the bytecode. The details of the J2ME CLDC verification process can be found in the *CLDC Byte Code Typechecker Specification*.[1]

Application-level Security

The CLDC application security is mainly ensured by adopting a *sandbox model*, by protecting system classes, and by restricting dynamic class loading.

MIDP Security

In the following, we present the security architecture of MIDP 1.0 and MIDP 2.0.

MIDP 1.0 Security

Application security in MIDP 1.0 is based on the *Java* sandbox model. The sandbox security model provided by MIDP 1.0 (and CLDC) is different from the conventional *Java* sandbox model. In fact, no *Security Manager* or *Security Policies* (as for J2SE/EE) are used for access control.

MIDP 2.0 Security

The difference between MIDP 1.0 security and MIDP 2.0 security is that in MIDP 2.0, accessing sensitive resources (APIs and functions) is not totally prohibited. Instead, MIDP 2.0 controls access to protected APIs by granting *permissions* to *protection domains* and binding each MIDlet on the device to one protection domain. Moreover, MIDP 2.0 introduces the ability to share record stores between MIDlet suites and provides end-to-end security by allowing secure networking using HTTPS protocol.

Sensitive APIs and Protection Domains

The sensitive APIs in MIDP 2.0 are all those related to connectivity in addition to the **PushRegistry** class that allows the automatic launching of a MIDlet at the occurrence of certain events. Access to sensitive APIs is protected by permissions. In this regard, a protection domain defines a set of permissions, and for each permission, the protection domain defines the level of access to the API protected by the permission. The level of access can be either *Allowed* or *User* (user has to be asked for permission). For instance, the *Trusted* domains have all permissions in the allowed mode.

Trusting MIDlet Suites

The procedure for determining to which protection domain a MIDlet should belong is device-specific. Some devices might trust only MIDlet suites obtained from certain servers. Others authenticate MIDlet suites using the Public Key Infrastructure (PKI).

Persistent Storage Security

The storage unit in J2ME CLDC is the *record store*. Each MIDlet suite can have one or more record stores. For MIDP 1.0, record stores were not allowed to be shared among MIDlet suites. In MIDP 2.0, sharing of record stores is allowed. The sharing mode can be set to **read-only** or **read/write**.

End-to-end Security

MIDP 2.0 specification mandates that HTTPS be implemented to allow secure connection with remote sites.

Vulnerability Analysis

In this section, we present our vulnerability analysis of J2ME CLDC security. We start by listing the most important previously reported flaws.

Previously Reported Flaws

Siemens S55 SMS

In late 2003, the Phenoelit hackers group [11] has discovered that the Siemens S55 phone has a vulnerability that makes the device send SMS messages without the user's authorization. This vulnerability is because a race condition during which the *Java* code can overlay the normal permission request with an arbitrary screen display.

Problems on Sun's MIDP RI

The Bug Database of Sun Microsystems contains hundreds of problems about J2ME CLDC. However, few are related to security. Those can be found in Bug 4802893: RI checks sockets before checking permissions [16], Bug 4959337: RSA Division byZero [15], and Bug 4824821: Return value of midpInitializeMemory is not checked.[14]

In the sequel, we present the vulnerabilities our investigation discovered.

Networking Vulnerabilities

MIDP SSL Vulnerability

To establish a secure connection with remote sites (HTTPS), MIDP uses the SSLv3.0 protocol. The implementation is based on KSSL [2] from Sun Labs. During the SSL handshake, the protocol has to generate random values to be used to compute the master secret. The latter is then used to generate the set of symmetric encryption keys. Hence, generating random values that are unpredictable is an important security aspect of SSL. The method **PRand.generateData** is used in MIDP to generate pseudo-random data. However, to update the seed of the pseudo-random number generator, only the system time is used. Hence, in order to obtain the random value generated by the client, all the attacker has to do is guess the precise system time (in milliseconds) at the moment of the pseudo-random value computation. This allows the attacker to guess a narrow interval of the correct system time. Afterwards, it remains only to try all possible values in that interval.

Unauthorized SMS Sending Vulnerability

The Phenoelit hackers group [11] has discovered that the Siemens S55 phone has a vulnerability that allows malicious code on the device to send SMS messages without the user's authorization. The idea is to fill the screen with different items when the device is asking the user for SMS permission. In this way, the user unwittingly will approve sending SMS messages under the assumption that he is answering a different question. The key point in this attack is that only the screen is overwritten. The button's (soft buttons) behavior is not changed and it is still about the SMS message permission. Since its publication, this flaw was always bound to Siemens S55 phones. However, nothing was said about its applicability to other phones. We found out that some other Siemens phones (by testing phone emulators) are vulnerable to SMS authorization attack. By checking the APIs of all these phones, we found that the SMS APIs are almost the same, which explains our findings. Sun reference implementation of MIDP is not vulnerable to this attack. Indeed, when the device asks the user for permission, MIDP RI prevents any modification to the screen until an answer is received. This is achieved by **preemptDisplay** method, which locks access to the display until the user provides an answer, then the display is unlocked by the **doneDisplay** method.

Storage System Vulnerabilities

The storage unit in J2ME CLDC is the *record store*. Each MIDlet suite can have one or more record stores. Record stores are identified by a unique full name, which is a concatenation of the vendor name, the MIDlet suite name, and the record store name. Figure 2 shows the structure of the storage system. The part of the *Java* platform responsible for manipulating the storage is called the Record Management System (RMS).

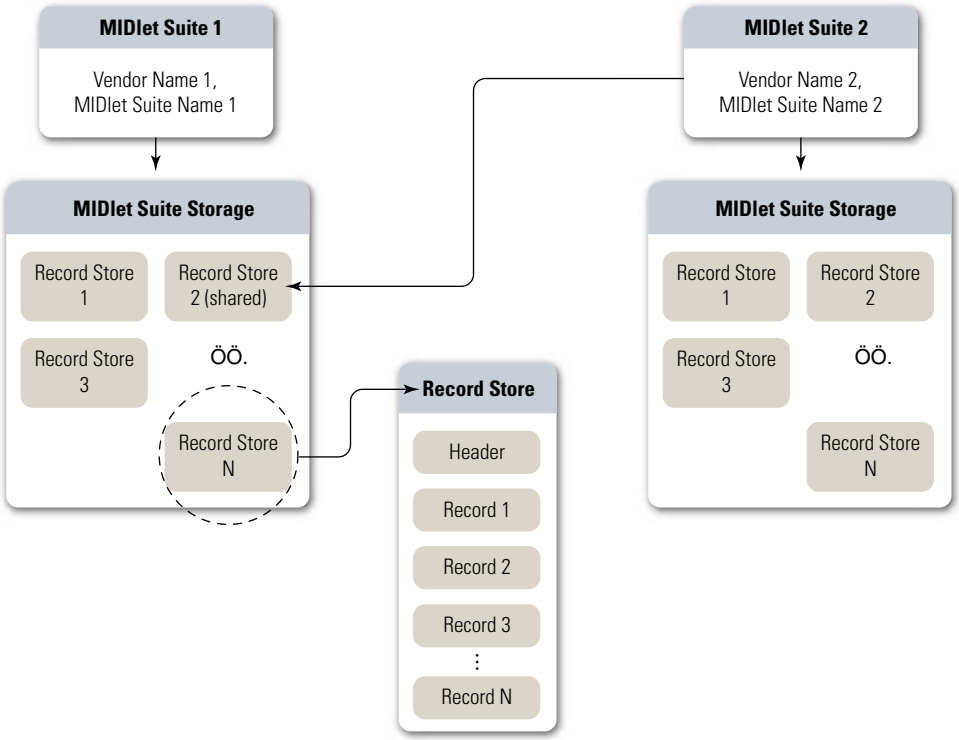


Figure 2 Record Stores in MIDP.

For MIDP 1.0, record stores were not allowed to be shared among MIDlet suites. In MIDP 2.0, sharing record stores is allowed. The MIDlet suite that created the record store can choose to share it or not. Moreover, the sharing mode can be set either to **read-only** or **read/write**. Detailed analysis of the RMS using MIDP specifications and Sun's reference implementation revealed the vulnerabilities listed below.

Unprotected Data Vulnerability
Data in record stores are not protected against malicious attacks. There is no mention in the specification of protecting sensitive user data, for example, with encryption and/or passwords. Data can be vulnerable to any attack from outside the RMS, such as when transferring data to or from a backup device. Moreover, the whole storage system in MIDP can be accessed from any file browsing application on the device. An example of such access can be performed using the FExplorer software, which is an application for Series 60 Nokia phones. It is worth noting that the SATSA API provides tools that can be used to protect data by cryptographic operations. However, such a protection is not part of the standards and is left to the applications.

Managing the Available Free Persistent Storage Vulnerability
When a MIDlet needs storing information in the persistent storage, it can create new records. Since the persistent storage is shared by all Midlets installed on the device, restrictions must be made on the amount of storage attributed to each MIDlet. This is motivated by embedded devices have limited memory resources. Otherwise, one cannot prevent a MIDlet from getting all the available space on the persistent storage of the device.

Unprotected Internal APIs Vulnerability
MIDP APIs provide the capabilities (methods) MIDlet programmers need to develop mobile applications. However, these are high-level APIs, which are designed to ease programming tasks. Therefore, they use helper low-level APIs that call native methods to deal with the device hardware. These low-level APIs have more privileges and less restrictions when dealing with the device hardware. Accordingly, it is crucial from the security standpoint to make sure that only high-level MIDP APIs could access these low-level privileged methods. In other words, developers should not have direct access to these low-level APIs. We discovered that the reference implementation of J2ME CLDC does not enforce this security measure. Figure 3 illustrates how the different levels of APIs are accessed.

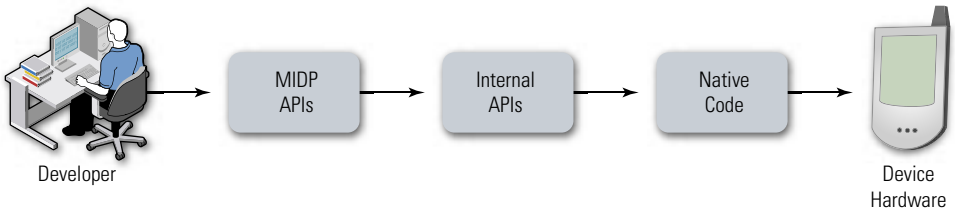


Figure 3 Various Levels of Abstraction in APIs for Mobile Devices.

In the storage system of MIDP, **RecordStore** one high-level API provides the functionalities the developer needs to manipulate record stores. This class also checks for access rights before doing such actions. For instance, no MIDlet is allowed to delete a record store of another MIDlet. There is another class, **RecordStoreFile**, that does not check for access rights. It should only provide services to the **RecordStore** class. However, we were able to directly access the methods of this class and succeeded to develop a MIDlet that was able to delete the record store of another MIDlet.

Retrieving and Transferring JAR Files from a Device
A typical scenario for downloading a MIDlet is to connect to a mobile application provider. Once a MIDlet is installed on the device, the user should be able to perform two kinds of operations, namely, execution and un-installation of the MIDlet. If, in addition, the user has the capability to transfer the MIDlet to another device, this might result into a breach of the intellectual property rights of the MIDlet provider. In our experiments, we succeeded in transferring MIDlets from one device to another, using an application available for Series 60 phones. [8] For instance, the FExplorer software [3] makes it possible to navigate through files and MIDlets installed on the device. It also provides options to send these files to other phones. It is important to note, however, that MIDlets protected by Digital Right Management (DRM) cannot be transferred from one device to the other (protection should be at least in the forward lock mode. [9]

Retrieving and Transferring MIDlet Persistent Data
Using FExplorer software, it is possible to transfer MIDlet persistent data from one device to another. Indeed, on Nokia 3600 phone, the **rms.db** file that holds all MIDlet persistent data is in the same location as JAD and JAR files and can be transferred following the same steps. Moreover, DRM protection does not cover **rms.db** files. Even if the MIDlet is DRM protected, the **rms.db** file can be transferred because DRM protection holds only for JAR files. [9]

KVM Vulnerabilities

Buffer Overflow Vulnerability
Buffer overflow is a well-known problem and may result in many security breaches. It occurs when the application does not perform bounds checking on data before copying it into a buffer. This can happen, for instance, when an application tries to overwrite a certain buffer with data from a larger buffer using the **strcpy** function (for C code). In this case, some values in the execution stack may be overwritten. Among these values, it is possible to overwrite the return address of the current function. By overwriting this address, the

attacker will be able to execute the code that he wants. By inspecting the source code of KVM, we identified a memory overflow vulnerability. The vulnerable code in **native.c** is the following:

```
sprintf(str_buffer," Method %s :: %s not found",
    className, methodName(thisMethod));
```

This code throws an exception if a native method is declared in a class file without implementing it elsewhere. The code does not check the size of the message that will be stored in **str_buffer**. Knowing that **str_buffer** is an array of 512 characters, it is clear that the code might result in a strange behavior if the size of the string to be stored in it exceeds 512 characters. On the Motorola V600 phone, the virtual machine crashes when the buffer in question is overflowed. One part of this string is the name of the invoked native method. Since no restrictions are imposed by the virtual machine on the size of method or field names, we wrote a simple *Java* program that declares a native method name counting 2,000 characters. This native method is declared without giving any implementation to force the throw of the exception. When the exception is thrown, the native method name overflows **str_buffer**, causing the overwriting of more than 1,500 characters in the memory segment.

Conclusion and Future Work

In this paper, we provide a careful study of J2ME CLDC security. We started by presenting the security architecture of this *Java* platform, followed by our evaluation of the underlying security model. In our study, we investigated the existing implementations of the platform to look for vulnerabilities. Actual phone models were tested and an important set of attack scenarios were designed and executed. We showed that the J2ME CLDC security model needs some refinements (*e.g.*, permissions and protection domains). Moreover, we demonstrated the presence of some vulnerabilities exist in the RI of MIDP 2.0 (*e.g.*, SSL implementation). Some phones were also shown to be vulnerable to security attacks like the Siemens SMS attack, while other phones followed a restrictive approach in implementing the J2ME CLDC platform. With this study in hand, improvements can be performed to harden the J2ME CLDC security. Therefore, the security model can be improved by fixing the discovered vulnerabilities and by proposing new security extensions. ■

References

1. G. Bracha, T. Lindholm, W. Tao and F. Yellin. CLDC Byte Code Typechecker Specification. *http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html*, January 2003.
2. Vipul Gupta and Sumit Gupta. KSSL: Experiments in Wireless Internet Security. Technical Report TR-2001-103, Sun Microsystems, Inc, Santa Clara, California, USA, November 2001.
3. D. Hugo. FExplorer Web Site. *http://users.skynet.be/domi/fexplorer.htm.*
4. G. Steele J. Gosling, B. Joy and G. Bracha. The *Java* Language Specification Second Edition. The *Java* Series. Addison-Wesley, Boston, Mass., 2000.
5. T. Lindholm and F. Yellin. The *Java* Virtual Machine Specification (Second Edition). Addison Wesley, April 1999.
6. Sun MicroSystems. Connected, Limited Device Configuration. Specification Version 1.0, *Java* 2 Platform Micro Edition. Technical report, Sun MicroSystems, California, USA, May 2000.
7. Sun MicroSystems. KVM PortingGuide. Technical report, Sun MicroSystems, California, USA, September 2001.
8. Nokia. Series 60 Platform. *http://www.nokia.com/nokia/0,8764,46827,00.html.*
9. OMA. Implementation Best Practices for OMA DRM v1.0 Protected MIDlets, May 2004.
10. J. Van Peursem. JSR 118 Mobile Information Device Profile 2.0, November 2002.
11. Phenoelit Hackers Group. *http://www.phenoelit.de/, 2003.*
12. Roger Riggs, Antero Taivalsaari, Mark VandenBrink, and Jim Holliday. Programming wireless devices with the *Java* 2 platform, micro edition: J2ME Connected Limited Device Configuration (CLDC), Mobile Information Device Profile (MIDP). Addison-Wesley, Reading, MA, USA, 2001.
13. T. Sayeed, A. Taivalsaari, and F. Yellin. Inside The K Virtual Machine. *http://java.sun.com/javaone/javaone2001/pdfs/1113.pdf*, Jan 2001.
14. Bug 4824821: Return value of midpInitializeMemory is not checked. *http://bugs.sun.com/bugdatabase/viewbug.do?bugid=4824821*, February 2003.
15. Bug 4959337: RSA Division byZero. *http://bugs.sun.com/bugdatabase/viewbug.do?bugid=4959337*, November 2003.
16. Bug 4802893: RI checks sockets before checking permissions. *http://bugs.sun.com/bugdatabase/viewbug.do?bugid=4802893*, January 2004.
17. A. Taivalsaari. JSR 139 J2ME Connected Limited Device Configuration 1.1, March 2003.

About the Author

Dr. Mourad Debbabi | is a Professor and the Associate Director of the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, Canada. He is also leading the CSL (Computer Security Laboratory) at Concordia University. He is the Specification Lead of four Standard JAIN (Java Intelligent Networks) Java Specification Requests (JSRs) dedicated to the elaboration of standard specifications for presence and instant messaging. Dr. Debbabi holds PhD and MSc degrees in computer science from Paris-XI Orsay, University, France. He published more than 100 research papers in journals and conferences on computer security. He can be reached at debbabi@ciise.concordia.ca.

Mohamed Saleh | is a PhD student at Concordia Institute for Information Systems Engineering (CIISE), Concordia University, Montreal, Quebec, Canada. He is a member of the Computer Security Laboratory (CSL) at CIISE. He is pursuing a Ph.D. thesis on the specification and analysis of security protocols using game semantics. He can be reached at m_saleh@ece.concordia.ca.

Chamseddine Talhi | is a researcher at CSL (Computer Security Laboratory) research group at Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, Canada. He holds an MSc degree from Constantine University, Algeria. He is pursuing a PhD thesis on the security of embedded Java platforms. He can be reached at talhi@ciise.concordia.ca.

Sami Zhioua | is a former researcher of CSL (Computer Security Laboratory) research group at Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, Canada. The main topic of his research activities is the acceleration of Java in the context of embedded systems. He can be reached at zhioua@ciise.concordia.ca.

Dr. Ying Zhu

by Ron Ritchey

This article continues our series profiling members of the IATAC Subject Matter Expert (SME) program. The SME profiled in this article is Dr. Ying Zhu. Dr. Zhu has been an Assistant Professor at the Department of Computer Science at Georgia State University since 2003 and an active member of the GSU Computer Science Department's *Hypermedia and Visualization Lab*. His research focus is computer graphics, bioinformatics, and software visualization—reflected in his numerous published journal articles and whitepapers presented at ACM SIGGRAPH and IEEE conferences. [1] Dr. Zhu has become increasingly interested in issues of computer security though, and he has responded by extending his research into the application of graphics and visualization to solving IA-related problems. Specifically, his IA research has focused on the feasibility of graphical passwords and the visualization of network and security management devices.

In *Graphical Passwords: A Survey*, co-authored in 2005 with fellow GSU student, Xiaoyuan Suo, and SIGGRAPH member, G. Scott. Owen, Dr. Zhu introduced concepts and assessed technologies for implementing graphical passwords. [2] In this important study, Dr. Zhu attempts to answer two predominant questions that arise when implementing graphical password authentication mechanisms:

1. Are graphical passwords as secure as text-based passwords?
2. What are the major design and implementation issues for graphical passwords?

This study, conducted on a select group of GSU students, attempts, in part, to survey human responses to graphical passwords in an effort to minimize human error and related security compromises often associated with committing alternative text-based passwords to memory. The study identifies multiple techniques for implementing graphical passwords (based on either pattern/picture recognition or pattern/picture reproduction) and contrasts them against text passwords in terms of susceptibility to attacks (*e.g.* brute force, dictionary, social engineering).

In conjunction with fellow Associate Professor and GSU colleague, Dr. Raheem Beyah, Director of the GSU *Communications Assurance and Performance (CAP) Group,* [3] Dr. Zhu is also attempting to weigh human responses to a variety of graphical visualization tools that could be used to quickly assess security events, network traffic patterns, and alarms. Like the Graphical Password study, one objective is to simultaneously simplify the visualization process while effectively reducing errors (*e.g.* false alarms, false reporting, *etc.*) resulting from poor human judgment. By varying graphical nuances such as color schemes, reporting patterns, and icon/object types, Dr. Zhu and Dr. Raheem are also hoping to strike a balance between functionality and the level of subject acceptance to the tools. Dr. Raheem and the CAP Group have conducted additional research into IA and performance analysis, including studies related to Rogue Access Point Detection, Network Card Identification, Resource Utilization, Prediction using Traffic Analysis, and Host Identification using Traffic Analysis, respectively.

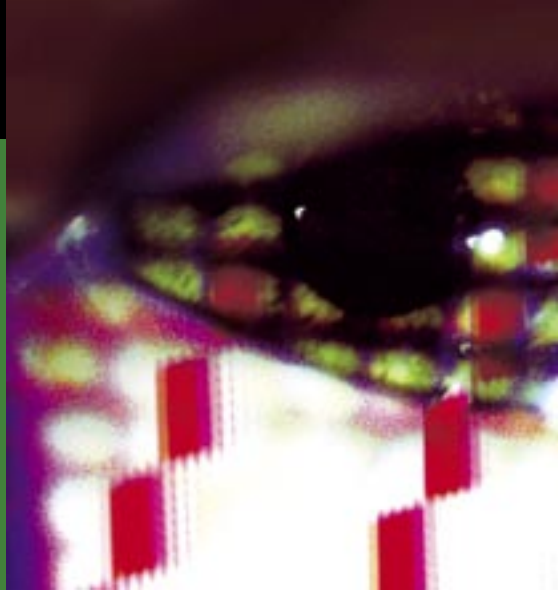
If you have a technical question for Dr. Zhu or other IATAC SMEs, please contact iatac.dtic.mil. The IATAC staff will assist you in reaching the SME best suited to helping you solve the challenge at hand. If you have any questions about the SME program or are interested in joining the SME database and providing technical support to others in your domains of expertise, please contact iatac@dtic.mil , and the URL for the SME application will be sent to you.

References

1. A selected list of Dr. Zhu's publications can be found at *http://www.cs.gsu.edu/~cscyiz/ying_zhu_publications.htm*
2. *Graphical Passwords, A Survey,* X. Suo, Y. Zhu, G. Owen, 21st Annual Computer Security Applications Conference, December 2005. *http://www.acsac.org/2005/papers/89.pdf*
3. View the website at *http://www.cs.gsu.edu/rbeyah/home.htm*

A Survey of Graphical Passwords

by Xiaoyuan Suo, Ying Zhu, and G. Scott Owen



Currently, the most common computer authentication method is text-based password. Studies have shown that users tend to pick short passwords or passwords that are easy to remember [1], which can also be easily guessed or broken. On the other hand, passwords that are hard to guess or break are often hard to remember. Studies also showed that since users can only remember a limited number of passwords, they tend to write them down or will use the same passwords for different purposes.

Graphical password schemes have been proposed as a possible alternative to text-based schemes, motivated partially by humans remembering pictures better than text. [2] In addition, if the number of possible pictures is sufficiently large, the password space of a graphical password scheme may exceed that of text-based. Because of these (presumed) advantages, there is a growing interest in graphical passwords. In addition to workstation and web log-in applications, graphical passwords have also been applied to automatic teller machines (ATM) and mobile devices.

In this paper, we conduct a survey of the existing graphical password techniques. We divide the graphical passwords into two categories: recognition based and recall based methods. We will also discuss the security and design issues in this area.

The survey

Recognition-Based Techniques

Using recognition-based techniques, a user is presented with a set of images and the user passes the authentication by recognizing and identifying the images he or she selected during the registration stage.

Dhamija and Perrig's graphical authentication scheme [3] was among the earliest graphical passwords (Figure 1). In their system, the user will be required to identify the pre-selected images to be authenticated. The user studies showed that the graphical password had a 90 percent success rate, while the equivalent text-based password had a 70 percent success rate. However, the average log-in time is longer than the traditional approach. Since the display is crowded with pictures, the process of selecting is slow.

To deal with the shoulder-surfing problem, Sobrado and Birget [4]

Figure 1 Random images used by Dhamija and Perrig [3]

developed several graphical password techniques. In one of the algorithms, the system will display a number of pass-objects (pre-selected by user) among many other objects. To be authenticated, a user needs to recognize pass-objects and click inside the convex hull formed by all the pass-objects. (See figure 2) To make the password hard to guess, Sobrado and Birget suggested using 1,000 objects, which makes the display very crowded and the objects almost indistinguishable. But fewer objects may lead to a smaller password space.

"Passface" is a technique developed by Real User Corporation. [5] The user will be asked to choose four images of human faces from a face database as their future password. (See figure 3) The user is authenticated if he or she correctly identifies the four faces after several rounds. The technique is based on the assumption that people can recall human faces easier than other pictures. User studies by Valentine [6, 7] showed that Passfaces are very memorable over long intervals. Comparative studies conducted by Brostoff and Sasse [8] showed that Passfaces had only a third of the login failure rate of text-based passwords. Their study also showed that the Passface-based log–in process took longer than text passwords; therefore, the effectiveness of this method is still uncertain. Davis, *et al.* [9] studied the graphical passwords

Figure 2 A shoulder-surfing resistant graphical password scheme [4]

Figure 3 An example of Passfaces (source: www.realuser.com)

created using the Passface technique and found obvious patterns among the chosen faces. This makes the Passface password somewhat predictable. This problem may be alleviated by arbitrarily assigning faces to users, but doing so would make it hard for people to remember the password.

Recall-Based Techniques

Using recall-based techniques, a user is asked to reproduce something that he or she created or selected earlier during the registration stage. We will discuss two types of picture password techniques: reproducing a drawing and repeating a selection.

Reproduce a Drawing

Jermyn, *et al.* [10] proposed a technique, called "Draw-a-secret (DAS)", which allows the user to draw their unique password. (See figure 4) A user is asked to draw a simple picture on a two-dimensional grid. The coordinates of the grids occupied by the picture are stored in the order of the drawing. During authentication, if the user's redraw touches the same grids in the same sequence, then the user is authenticated. Jermyn, *et al.* suggested that given reasonable-length passwords in a 5 X 5 grid, the full password space of DAS is larger than that of the full text password space.

Thorpe and van Oorschot [11] analyzed the memorable password space of the graphical password scheme by Jermyn *et al.* [10]. They introduced the concept of graphical dictionaries and studied the possibility of a brute-force attack using such dictionaries. They defined a length parameter for the DAS type graphical passwords and showed that DAS passwords of length 8 or larger on a 5 x 5 grid may be less

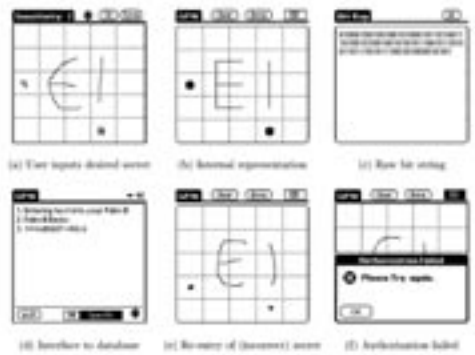


Figure 4 Draw-a-Secret (DAS) technique [10]

susceptible to dictionary attack than textual passwords. Further studies by Nali and Thorpe [12] showed that certain symmetries (*e.g.* crosses and rectangles), letters, and numbers were common among DAS passwords, indicating that users choose graphical passwords with predictable characteristics.

Syukri, *et al.* [13] proposed a system in which authentication is conducted by having the user draw their signatures using a mouse. (See figure 5) According to the paper the rate of successful verification was satisfying. The biggest advantage of this approach is that one's signature is hard to fake and easy to remember. However, not everybody is familiar with using a mouse as a writing device. One possible solution to this problem would be to use a pen-like input device, but such devices are not widely used, and adding new hardware to the current system can be expensive. Such a technique is more useful for devices such as

Figure 5 A signature is drawn by mouse [13]

a personal digital assistant (PDA), which may already have a stylus.

Repeat a sequence of actions

Blonder [14] designed a graphical password scheme in which a password is created by having the user click on several locations on an image. During authentication, the user must click on the approximate areas of those locations. The image can assist users to recall their password. Passlogix [15] has developed a graphical password system based on this idea. In their implementation (see figure 6), users must click on various objects in the image in the correct sequence and within a predefined boundary to be authenticated.

Figure 6 A recall-based technique developed by Passlogix [15]

The "PassPoint" system, by Wiedenbeck, *et al.* [16, 17] extended Blonder's idea by allowing users to click on any pixels on an image. Because any picture can be used and because a picture may contain hundreds of thousands of memorable points, the possible password space is considerably large.

Discussion

Is a graphical password better than text-based password?

There is still no clear answer to this question. Many user studies in our survey have confirmed that people can recall graphical passwords more reliably than text-based passwords over a long time. This seems to be the main advantage of graphical passwords. Some graphical password techniques have been shown to provide a password space similar to or larger than that of text-based passwords. [10, 11, 18]

There is little study on the possible techniques for breaking graphical passwords. As a result, there is still no concrete evidence to prove whether a graphical password is more or less secure than a text-based password. Recognition-based graphical passwords tend to have smaller password spaces than the recall-based methods, and therefore seem more vulnerable to attacks. In addition, studies on the Passface technique have shown that people often choose weak and predictable graphical passwords [9], a serious problem typically associated with text-based passwords. Nali and Thorpe's study [12] revealed similar predictability among the graphical passwords created with DAS technique. [10] Much more research efforts are needed to understand the nature of graphical passwords created by real-world users.

Major design and implementation issues of graphical passwords

Password space—A large password space is necessary to defend against guess-based attacks. For recognition-based methods, one solution is to have several rounds of verifications. But this will make the log-in process longer and tedious. Another solution is to deploy large number of decoy-images. This would also slow down the log-in process. In addition, this solution is not suitable for mobile devices because of very limited user interface space. For "reproduce-a-drawing" methods, possible solutions include maintaining a large canvas, reducing the tolerance of error, and requiring users to draw complex pictures. However, this may result in sophisticated and perhaps overly sensitive recognition programs that generate lots of false negatives. For "repeat-a-sequence" methods, the solution is to use a highly detailed image and provide large number of potential click points. Users are also required to click on many points to generate a long password. The drawback, however, is that users may have difficulty memorizing the many point of clicks.

Shoulder-surfing resistance—At this point, only a few recognition-based techniques are designed to resist shoulder-surfing. None of the recall-based based techniques are considered should-surfing resistant.

Storage—Graphical passwords require much more storage space than text-based passwords. Tens of thousands of pictures have to be maintained in a centralized database. Network transfer delay is also a concern for graphical passwords, especially for recognition-based techniques in which hundreds of pictures may need to be displayed for each round of verification.

Usability—A major complaint among the users of graphical passwords is that the password registration and log-in process take too long, especially in recognition-based approaches.

Conclusion

We have conducted a survey of existing graphical password techniques. More details can be found in our earlier publication. [19] A comparison of current graphical password techniques is presented in Table 1.

Our preliminary analysis suggests that it is more difficult to break graphical passwords using the traditional attack methods such as brute force search, dictionary attack, or spyware. Much more research and user studies are needed for graphical password techniques to achieve a higher level of maturity and usefulness. ■

Techniques	Usability		Security Issues	
	Authentication Process	Memorability	Password Space	Possible Attack Methods
Text-based password	Type in password; can be very fast	Depends on the password. Long and random passwords are hard to remember.	94^K (there are 94 printable characters excluding SPACE, K is the length of the password). The actual password space is usually much smaller.	Dictionary attack, brute force search, guess, spyware, shoulder surfing, ***etc.***
Perrig and Song [20]	Pick several pictures out of many choices. Takes longer to create than text password	Limited user study showed that more people remembered pictures than text-based passwords.	N!/K!(N-K)! (N is the total number of pictures; K is the number of pictures in the graphical password.)	Brute force search, guess, shoulder-surfing
Sobrado and Birget [4]	Click within an area bounded by pre-registered picture objects; can be very fast	Can be hard to remember when large numbers of objects are involved	N!/K!(N-K)! (N is the total number of picture objects; K is the number of pre-registered objects.)	Brute force search, guess
Man, ***et al.*** [21] Hong, ***et al.*** [22]	Type in the code of pre-registered picture objects; can be very fast	Users have to memorize both picture objects and their codes. More difficult than text-based password	Same as the text-based password	Brute force search, spyware
Passface [5]	Recognize and pick the pre-registered pictures; takes longer than text-based password	Faces are easier to remember, but the choices are still predictable.	N^K (K is the number of rounds of authentication; N is the total number of pictures at each round.)	Dictionary attack, brute force search, guess, shoulder surfing
Jansen ***et al.*** [23]	User registers a sequence of images; slower than text-based password	Pictures are organized according to different themes to help users remember.	N^K (N is the total number of pictures, K is the number of pictures in the graphical password. N is small because of the size limit of mobile devices)	Brute force search, guess, shoulder surfing
Takada and Koike [24]	Recognize and click on the pre-registered images; slower than text-based password.	Users can use their favorite images; easy to remember than system assigned pictures	(N+1)^K (K is the number of rounds of authentication; N is the total number of pictures at each round)	Brute force search, guess, shoulder surfing
Jermyn, ***et al.*** [10], Thorpe and van Oorschot [11, 25]	Users draw something on a 2D grid	Depends on what users draw. User studies showed the drawing sequence is hard to remember.	Password space is larger than text-based password, but the size of DAS password space decreases significantly with fewer strokes for a fixed password length.	Dictionary attack, shoulder surfing
Syukri, ***et al.*** [13]	Draw signatures using mouse. Need a reliable signature recognition program	Very easy to remember, but hard to recognize	Infinite password space	Guess, dictionary attack, shoulder surfing
Goldberg ***et al.*** [26]	Draw something with a stylus onto a touch sensitive screen.	Depends on what users draw	Infinite password space	Guess, dictionary attack, shoulder surfing
Blonder [14], Passlogix [15], Wiedenbeck, ***et al.*** [16, 17]	Click on several pre-registered locations of a picture in the right sequence.	Can be hard to remember	N^K (N is the number of pixels or smallest units of a picture; K is the number of locations to be clicked on)	Guess, brute force search, shoulder surfing

Table 1 Comparison of Major Graphical Password Techniques

References

1. A. Adams and M. A. Sasse, "Users are not the enemy: why users compromise computer security mechanisms and how to take remedial measures," Communications of the ACM, vol. 42, pp. 41-46, 1999.
2. R. N. Shepard, "Recognition memory for words, sentences, and pictures,"Journal of Verbal Learning and Verbal Behavior, vol. 6, pp. 156-163, 1967.
3. R. Dhamija and A. Perrig, "Deja Vu: A User Study Using Images for Authentication," in Proceedings of 9th USENIX Security Symposium, 2000.
4. L. Sobrado and J.-C. Birget, "Graphical passwords," The Rutgers Scholar, An Electronic Bulletin for Undergraduate Research, vol. 4, 2002.
5. RealUser, "*www.realuser.com.*"
6. T. Valentine, "An evaluation of the Passface personal authentication system," Technical Report, Goldsmiths College, University of London 1998.
7. T. Valentine, "Memory for Passfaces after a Long Delay," Technical Report, Goldsmiths College, University of London 1999.

8. S. Brostoff and M. A. Sasse, "Are Passfaces more usable than passwords: a field trial investigation," in People and Computers XIV - Usability or Else: Proceedings of HCI. Sunderland, UK: Springer-Verlag, 2000.
9. D. Davis, F. Monrose, and M. K. Reiter, "On user choice in graphical password schemes," in Proceedings of the 13th Usenix Security Symposium. San Diego, CA, 2004.
10. I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. D. Rubin, "The Design and Analysis of Graphical Passwords," in Proceedings of the 8th USENIX Security Symposium, 1999.
11. J. Thorpe and P. C. v. Oorschot, "Graphical Dictionaries and the Memorable Space of Graphical Passwords," in Proceedings of the 13th USENIX Security Symposium. San Diego, USA,: 2004.
12. D. Nali and J. Thorpe, "Analyzing User Choice in Graphical Passwords," Technical Report, School of Information Technology and Engineering, University of Ottawa, Canada May 27 2004.
13. A. F. Syukri, E. Okamoto, and M. Mambo, "A User Identification System Using Signature Written with Mouse," in proceedings of the Third Australasian Conference on Information Security and Privacy (ACISP): Springer-Verlag Lecture Notes in Computer Science (1438), 1998, pp. 403-441.
14. G. E. Blonder, "Graphical passwords," in Lucent Technologies, Inc., Murray Hill, NJ, U. S. Patent, Ed. United States, 1996.
15. Passlogix, "*www.passlogix.com.*"
16. S. Wiedenbeck, J. Waters, J. C. Birget, A. Brodskiy, and N. Memon, "Authentication using graphical passwords: Basic results," in Human-Computer Interaction International (HCII 2005). Las Vegas, NV, 2005.
17. S. Wiedenbeck, J. Waters, J. C. Birget, A. Brodskiy, and N. Memon, "Authentication using graphical passwords: Effects of tolerance and image choice," in Symposium on Usable Privacy and Security (SOUPS). Carnegie-Mellon University, Pittsburgh, 2005.
18. J.-C. Birget, D. Hong, and N. Memon, "Robust discretization, with an application to graphical passwords," Cryptology ePrint archive 2003.
19. X. Suo, Y. Zhu, and G. S. Owen, "Graphical Password: A Survey," in Proceedings of Annual Computer Security Applications Conference (ACSAC). Tucson, Arizona: IEEE, 2005.
20. A. Perrig and D. Song, "Hash Visualization: A New Technique to Improve Real-World Security," in Proceedings of the 1999 International Workshop on Cryptographic Techniques and E-Commerce, 1999.
21. S. Man, D. Hong, and M. Mathews, "A shoulder-surfing resistant graphical password scheme," in Proceedings of International conference on security and management. Las Vegas, NV, 2003.
22. D. Hong, S. Man, B. Hawes, and M. Mathews, "A password scheme strongly resistant to spyware," in Proceedings of International conference on security and management. Las Vergas, NV, 2004.
23. W. Jansen, S. Gavrila, V. Korolev, R. Ayers, and R. Swanstrom, "Picture Password: A Visual Login Technique for Mobile Devices," National Institute of Standards and Technology Interagency Report NISTIR 7030, 2003.
24. T. Takada and H. Koike, "Awase-E: Image-based Authentication for Mobile Phones using User's Favorite Images " in Human-Computer Interaction with Mobile Devices and Services vol. 2795 / 2003 Springer-Verlag 2003 pp. 347 - 351
25. J. Thorpe and P. C. v. Oorschot, "Towards Secure Design Choices for Implementing Graphical Passwords," in Proceedings of 20th Annual Computer Security Applications Conference (ACSAC). Tucson, USA.: IEEE, 2004.
26. J. Goldberg, J. Hagman, and V. Sazawal, "Doodling Our Way to Better Authentication," in Proceedings of Human Factors in Computing Systems (CHI). Minneapolis, Minnesota, USA.: ACM, 2002.

About the Author

Xiaoyuan Suo | is a computer science master's student at Georgia State University. Her research interest is in information security visualization. She may be contacted at xsuo@student.gsu.edu.

Ying Zhu | is an assistant professor of computer science at GSU. His research interest is in visualization and computer graphics. He may be contacted at yzhu@cs.gsu.edu.

G. Scott Owen | is a professor of computer science at GSU. He currently serves as the president of ACM SIGGRAPH. He is interested in computer graphics, visualization, and Human-Computer Interaction (HCI). He may be contacted at owen@siggraph.org.

▷▷ *Authors Bio's form page 17*

About the Author

Chris Jones | 1st Lt Christopher O. Jones, USAF, is currently stationed at RAF Lakenheath, England, where he is the executive officer for the 48th Maintenance Group, 48th Fighter Wing. He holds a BS degree in computer science from the United States Air Force Academy and a MS degree in computer science from the Air Force Institute of technology. He may be contacted at christopher.jones@lakenheath.af.mil.

Robert F. Mills | is an assistant professor of electrical engineering at the Air Force Institute of Technology (AFIT). He retired from the USAF in 2003 after serving 21 years in the communications career field. He holds a BS degree in electrical engineering from Montana State University, an MS in electrical engineering from AFIT, and a PhD in electrical engineering from the University of Kansas. He teaches and conducts research in computer security, network management, and Command, Control, Communications, Computers, & Intelligence (C4I) systems. He may be contacted at robert.mills@afit.edu

Richard Raines | Dr. Richard "Rick" Raines is the Director of the CISER at the AFIT. Dr. Raines holds a BS degree in Electrical Engineering from the Florida State University, an MS in Computer Engineering from AFIT, and a PhD in Electrical Engineering from Virginia Tech. He teaches and conducts research in information security and global communications. He may be contacted at richard.raines@afit.edu

Georgia State University Computer Information Systems and Security

by Ron Ritchey and Jennifer Kurtz

The Georgia State University (GSU) Department of Computer Information Systems (CIS) within the J. Mack Robinson College of Business [1] is the largest department focusing exclusively on information systems studies in the United States. Chaired by accomplished information systems security author, Dr. Richard Baskerville [2], the CIS department also serves as the official administrative home of both the Association for Information Systems (AIS) [3] and the International Conference on Information Systems (ICIS). [4]

The CIS department offers multiple degrees in Computer Information Systems (CIS), including BBA, MBA, MS, and PhD programs. Students that enroll in the MS, CIS program are able to take advantage of GSU courses that either directly or indirectly address security. For example, through the GSU "Management of Information Services" course (CIS 8100), students are exposed to management-oriented issues related to areas such as system performance, distributed computing, and systems-level security. A more rigorous security offering is provided through the "Security and Privacy of Information and Information Systems" course (CIS 8080), which is designed to develop knowledge and skills for implementing security within organizations. Coursework focuses on concepts and methods associated with planning, designing, implementing, managing, and auditing security at all levels and on all systems platforms, including worldwide networks. It additionally addresses issues associated with ethical uses of information and privacy considerations.

Beyond curriculum-based security education, GSU also hosts the CAP Research Group, headed by Professor Raheem Beyah. The CAP Group focuses on researching, developing, and implementing experimental algorithms in a simulated network environment that are designed specifically to promote network security. Research predominantly focuses on the two following main areas:

- **Device identification**—Includes techniques specific to wireless networks and network card types, as well as general approaches to identifying devices. Approaches to estimating device characteristics through traffic analysis are also being researched.
- **Techniques for detecting and combating rogue devices on networks**—Research areas include creating algorithms for detecting rogue access points and other malicious devices that have been placed on a network.

In alliance with the GSU *Hypermedia and Visualization Lab* [5], the CAP Group is also engaged in research efforts related to system and network device security visualization.

To date, the CAP Group has contributed multiple publications to the industry, including studies into Rogue Access Point Detection, Network Card Identification, Resource Utilization, Prediction using Traffic Analysis, and Host Identification using Traffic Analysis. Several of these publications have appeared in IEEE-sponsored proceedings, including the Information Assurance Workshop (IAW), Global Communications Conference (GLOBECOM), International Workshop on IP Operations and Management (IPOM), and International Conference on Communications (ICC), among others. The CAP Group is sponsored by the Georgia Tech "Facilitating Academic Careers in Engineering and Science (FACES)" Program [6] and the Georgia Tech Communications Systems Center. [7] ■

References

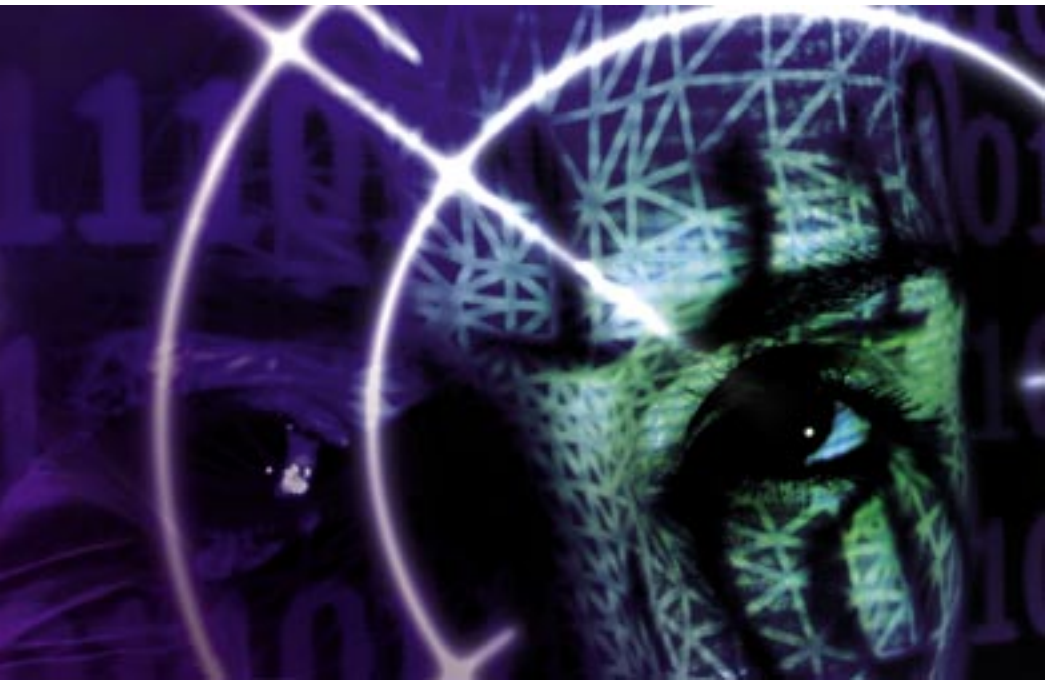
1. *http://www2.cis.gsu.edu/cis/about/index.asp*
2. View Dr. Baskerville's relevant security publications: *http://www.cis.gsu.edu/~rbaskerv/library/*
3. *http://plone.aisnet.org/*
4. *http://icisnet.aisnet.org//intro.shtml*
5. *http://hvl.cs.gsu.edu/*
6. *http://www.faces.gatech.edu/*
7. *http://www.csc.gatech.edu/tiddly/*

Letter to the Editor

Q ***In your Volume 8 Number 3 edition of the* IAnewsletter*, there was an article titled "DoD Information Assurance/Computer Network Defense (IA/CND) Enterprise-wide Solutions Steering Group or ESSG" that mentioned procuring something called a Host-Based Security System (HBSS). I was wondering if you could explain what this is and where the ESSG is in their procurement of it.***

A A Host-Based Security System (HBSS) is a host-based intrusion-prevention system that increases the difficulty for adversaries to compromise Department of Defense (DoD) hosts. This automated system will provide network security personnel a means of preventing, detecting, tracking, reporting, and remediating malicious activities and incidents on information systems across the DoD, DoD-related Intelligence Agencies, the National Guard, the Coast Guard, and the Reserves. The primary purpose of the HBSS is to support CND capabilities such as those listed in DoDI 8500.2.

On March 31, 2006, the Defense Information Systems Agency (DISA) purchased from industry, a means to develop and deploy an automated HBSS. Under this contract award, help desk support, classroom training, and virtual on-demand training will also be offered. The intent is to deploy the HBSS to selected DoD pilot sites to test, baseline, and develop policy. Should the ESSG determine that the system is refined and prepared for full operational capability, the Enterprise-wide license option of the contract will be executed and DoD enterprise-wide deployment will be initiated. Software and training for the HBSS is expected to be offered in the 4th quarter of CY2006. For more information, please do not hesitate to contact us at iatac@dtic.mil. ■



What lies ahead for the ESSG? In FY 2006, the ESSG will procure a Host-Based Security System (HBSS), a Tier 3 (Base/Unit-Level) Security Information Manager (SIM) tool, and a tool to help mitigate insider threats. The ESSG team is also working on hardening the Secret Internet Protocol Router Network (SIPRNet), mitigating the wireless threat, and procuring components of an Enterprise Sensor Grid (ESG) and User Defined Operational Picture (UDOP). ▷▷ *IAnewsletter 8.3*

FREE Products Order Form

Instructions: All IATAC LIMITED DISTRIBUTION reports are distributed through DTIC. If you are not registered DTIC user, you must do so prior to ordering any IATAC products (unless you are DoD or Government personnel). To register On-line: *http://www.dtic.mil/dtic/registration*. The *IAnewsletter* is UNLIMITED DISTRIBUTION and may be requested directly from IATAC.

Name _____ DTIC User Code _____

Organization _____ Ofc. Symbol _____

Address _____ Phone _____

_____ E-mail _____

_____ Fax _____

Please check one: ☐ USA ☐ USMC ☐ USN ☐ USAF ☐ DoD
☐ Industry ☐ Academia ☐ Government ☐ Other

Please list the Government Program(s)/Project(s) that the product(s) will be used to support: _____

_____

LIMITED DISTRIBUTION

IA Tools Reports (softcopy only)
☐ Firewalls ☐ Intrusion Detection ☐ Vulnerability Analysis

Critical Review and Technology Assessment (CR/TA) Reports
☐ Biometrics (soft copy only) ☐ Configuration Management ☐ Defense in Depth (soft copy only)
☐ Data Mining (soft copy only) ☐ IA Metrics (soft copy only) ☐ Network Centric Warfare (soft copy only)
☐ Wireless Wide Area Network (WWAN) Security ☐ Exploring Biotechnology (soft copy only)
☐ Computer Forensics* (soft copy only. DTIC user code MUST be supplied before these reports will be shipped)

State-of-the-Art Reports (SOARs)
☐ Data Embedding for IA (soft copy only) ☐ IO/IA Visualization Technologies (soft copy only)
☐ Modeling & Simulation for IA ☐ Malicious Code (soft copy only)
☐ A Comprehensive Review of Common Needs and Capability Gaps

UNLIMITED DISTRIBUTION

IAnewsletters Hardcopies are available to order. The list below represents current stock.
Softcopy back issues are available for download at *http://iac.dtic.mil/iatac/IA_newsletter.html*

Volumes 4		☐ No. 2	☐ No. 3	☐ No. 4
Volumes 5	☐ No. 1	☐ No. 2	☐ No. 3	☐ No. 4
Volumes 6	☐ No. 1	☐ No. 2	☐ No. 3	☐ No. 4
Volumes 7	☐ No. 1	☐ No. 2	☐ No. 3	☐ No. 4
Volumes 8	☐ No. 1	☐ No. 2	☐ No. 3	☐ No. 4
Volumes 9	☐ No. 1	☐ No. 2		

Fax completed form to IATAC at 703/984-0773

Calendar

August

Information Assurance Engineering
21–22 August 2006
Washington, DC
http://ttcus.com/newsite/defense-seminars/IAA_s06.html

LandWarNet Conference (formerly DOIM/AKM Conference)
22–24 August 2006
Greater Fort Lauderdale, Broward County Convention Center
Fort Lauderdale, FL.
http://www.afcea.org/events/landwarnet/default.asp

FBI HQ Security Awareness
30 August 2006
J. Edgar Hoover Building
Washington, DC
http://www.fbcinc.com/event.asp?eventid=Q6UJ9A00AJPF

Phoenix Tech-Security Conference
30 August 2006
Phoenix, AZ
http://dataconnectors.com/events/2006/08phoenix/agenda.asp

September

Disruptive Technologies Conference
06–07 September 2006
Hyatt Regency Capitol Hill
Washington, DC
http://www.ndia.org/Template.cfm?Section=6920&Template=/ContentManagement/ContentDisplay.cfm&ContentID=11983

Army Materiel Command Information Assurance Conference 2006
12–14 September 2006
The Alfred M. Gray Marine Corps Research Center
Quantico Marine Corps Base, VA
https://www.technologyforums.com/6AM/index.asp

2006 Biometrics Technology Expo
19–21 September 2006
Baltimore Convention Center
Baltimore, MD
http://www.biometricstechexpo.com/

IEEE SECON 2006
25–28 September 2006
Hyatt Regency
Reston, VA
http://www.ieee-secon.org/2006/index.html

October

24th Command, Control, Communications, Computers & Intelligence Systems Technology (C4IST)
03–05 October 2006
Barnes Field House
Fort Huachuca, AZ
http://www.laser-options.com/afcea/

InfoTech 2006
17–19 October 2006
Dayton Convention Center
Dayton, OH
http://www.afcea-infotech.org/index.htm

Tactical IA 2006
23–25 October 2006
Westin Arlington Gateway
Arlington, VA
http://idga.org/cgi-bin/templates/singlecell.html?topic=233&event=10704

International Security Conference (ISC) East Conference and Expo 2006
24–25 October 2006
Jacobs Javits Center
New York, NY
http://www.isceast.com/App/homepage.cfm?moduleid=42&appname=100036

IATAC
Information Assurance Technology Analysis Center
13200 Woodland Park Road, Suite 6031
Herndon, VA 20171

To change, add, or delete your mailing or e-mail address (soft-copy receipt), please contact us at the address above or call us at: 703/984-0775, fax us at: 703/984-0773 , or send us a message at: iatac@dtic.mil