Software Security Testing of an Online Banking System

A Unique Research Experience for Undergraduates and Computer Teachers

Dianxiang Xu
National Center for the Protection of the Financial Infrastructure
Dakota State University
Madison, SD 57042, USA
1-605-256-5694
dianxiang.xu@dsu.edu

ABSTRACT

This paper presents a unique summer project for a group of undergraduate students and high school computer teachers to gain research experiences in the area of cybersecurity. The students and teachers were selected from the participants in the NSF REU and RET programs at the host institution. Through the research on security testing of a real-world online banking system, the students and teachers have not only learned about the cutting-edge security testing techniques, but also made publishable contributions to the research base. The two collaborating graduate assistants served as an immediate role model for the undergraduates and an indirect role model for high school students through the teachers. With the help from the graduate assistants, the students and teachers were able to work effectively toward achieving their research objectives. The internal competition helped the participants get a better sense of achievement and satisfaction. The research experiences also prepared the teachers with the necessary knowledge for introducing cybersecurity topics (e.g., secure programming) into future classroom activity. As such, the project described in this paper provides a model summer program for undergraduate and/or K-12 teachers to gain research experiences.

Categories and Subject Descriptors

D.2.5 [**Testing and debugging**]: Testing tools (e.g., data generators, coverage testing). D.4.6 [**Security and protection**]: Access Controls. K.6.5 [**Security and protection**]: Authentication, Unauthorized access.

General Terms

Security.

Keywords

Cybersecurity, security testing, software testing, access control, security attacks, mutation analysis.

1. INTRODUCTION

The widespread applications of Internet-based information systems have raised serious concerns about cybersecurity. Cybersecurity involves policies, procedures, methods, and techniques that protect and defend information and information systems by ensuring availability, integrity, authentication, confidentiality and non-repudiation. The cybersecurity workforce is key to assuring adequacy of security measures to protect and defend its information and information systems. According to the 2010 white paper of the CSIS commission on cybersecurity for the 44th Presidency [7], the United States is facing "a human capital crisis in cybersecurity." The U.S. Bureau of Labor statistics projects "there will be over 155,000 job openings for individuals with these important (cybersecurity) skills between now and 2018 - the 2nd fastest job growth rate in the United States"[1]. To strengthen the future cybersecurity environment, we need to motivate our young generations to develop a career interest in this important area. In particular, we need high quality educators to expand cybersecurity education at early age. Career planning and preparation during high school and postsecondary studies have an important impact on next generation workforce.

To meet the educational needs of cybersecurity, the author, in collaboration with his colleagues, has received grants from the US National Science Foundation (NSF) to establish the REU (Research Experiences for Undergraduates) and RET (Research Experiences for Teachers) Sites in Cybersecurity at Dakota State University. As South Dakota's information technology university, DSU is a national Center of Academic Excellence in Information Assurance Research and Education designated by the US National Security Agency and the US Department of Homeland Security. DSU offers undergraduate and graduate degrees in information assurance. In 2009, DSU established the National Center for the Protection of the Financial Infrastructure sponsored by the State of South Dakota, in partnership with Citibank, Wells Fargo, First Premier, the Federal Reserve, FDIC, and DHS. The goal of this center is to address the major security issues within the technical, business, legal and policy contexts of banking and finance through education and research.

In this paper, we present a project that is aligned with the center's mission as well as the objectives of the REU and RET sites. Specifically, the project involved two undergraduate students from the 2012 REU program and four high-school computer teachers from the 2012 RET program. Under the supervision of the faculty mentor (the author) and in collaboration with two graduate assistants, the students and teachers investigated the security testing of a real-world online banking system. They have not only learned about the cutting-edge security testing techniques, but also made contributions to the research base, which resulted in the publication of an international conference

SIGCSE '13, March 6–9, 2013, Denver, Colorado, USA.

[1] http://mnc3.advanceitmn.org/

paper [13]. The collaborating graduate assistants served as an immediate role model for the undergraduates and an indirect role model for high school students through the teachers. An internal competition was held to help the participants get a better sense of achievement and satisfaction.

The remainder of this paper is organized as follows. Section 2 gives an introduction to the REU and RET Sites. Section 3 describes the technical background about the research project through which the students and teachers gained the research experiences. Section 4 introduces the project design. Section 5 presents the results of the project. Section 6 concludes this paper.

2. The REU and RET Sites in Cybersecurity

2.1 The REU Site

The REU Site in Cybersecurity is a three-year (2010, 2011, 2012) project that provides a diverse pool of motivated undergraduate students with a competitive research experience in cybersecurity. Each year, the REU site offers a nine-week summer program for nine undergraduate students to develop confidence as researchers and scientists. To assure the diversity of participants, 5-6 students (more than 50%) are selected from other institutions and 3-4 students are from DSU. Recruits are primarily junior or senior undergraduates majoring in computer science, computer and network security, software engineering, information systems, and related areas.

During the eight-week summer program, each student works with a faculty mentor on a carefully designed research project. In the first two weeks, students focus on developing projects. Tutorials and workshops are provided to help students establish necessary background knowledge and learn about how to conduct cybersecurity research. Students also work with their individual advisors, read literature, and design their projects. From week 3 through week 8, students carry out their projects under the supervision of their mentors. They also are required to attend the weekly seminars by faculty and invited industry speakers and report their weekly progress. During week 9, students prepare a written report in the format of an IEEE conference paper and an oral presentation modeled after a computer science conference.

Through the research experiences, the students are expected to improve their ability to formulate and solve a research problem, increase their communication proficiency, and establish a professional relationship with their faculty mentor. Some of them will be motivated to pursue graduate studies.

2.2 The RET Site

The RET Site in Cybersecurity is a three-year (2012, 2013, and 2014) project that provides in-service high school computer teachers across the State of South Dakota with an experience of conducting cutting-edge research in cybersecurity. It offers an eight-week summer program for ten teachers to conduct research with DSU faculty and continue to help them implement research-based activities in their classrooms throughout the subsequent academic year. During the summer program, the teachers are paired up to work with a DSU faculty mentor and his research team (graduate and undergraduate assistants) on a research project. Following the summer program, the teachers continue to develop learning materials for integrating their research experiences into their classroom activity.

Similar to the REU program, the schedule of the RET program consists of three phases: planning, execution, and presentation. In the first two weeks, teachers focus on developing project plans. They are paired up to work with their mentor and group members, read literature, and design their projects. From week 3 through week 7, they carry out their projects under the supervision of their mentor. During week 7, they prepare a written report in the format of an IEEE conference paper, an oral presentation modeled after a computer science conference, and a plan for curriculum development. The summer program also offers a three-credit course *Special Topics in Cybersecurity*. Through the integration of their research experiences and findings into classroom activity in the subsequent academic year, the high school teachers can expose their students to cybersecurity and motivate them to develop a career interest. A benefit for the faculty mentors is that they can disseminate the findings of their research programs to non-traditional audiences (high school teachers) and increase their understanding of high school computer technology education so as to improve their own teaching.

3. SOFTWARE SECURITY TESTING

In this section, we introduce the technical background of the research project. In addition to the basic concepts, we describe the advanced techniques that the students and teachers would learn from the research experience as well as the open research issues to which they could contribute.

Software security is a major source of cybersecurity risks. Many attacks against a company's network come at the application layer. They bypass the security mechanisms at the crypto, protocols and systems layers (e.g., firewall and intrusion detection). While secure coding is important, software security requires a lot more beyond coding [11]. For example, a large portion of the security flaws uncovered during Microsoft's "security push" in 2002 were closely related to design-level problems [4]. Principled use of assurance techniques needs to be considered throughout software development processes. Unfortunately, there is a lack of well-trained workforce for secure software development.

Software testing is a dominant practice for quality assurance of computer software. It aims at finding programming errors by executing a program with test cases, including test inputs and test oracles (i.e., expected results). A test fails if the actual result of its execution is different from its expected result. Software testing is an expensive and time-consuming activity, accounting for 50% or more of the total development costs. To create test cases, there are two basic strategies: black box and white box. The former views the system under test (SUT) as a block box and derives test cases from a requirements specification. The latter creates test cases based on the SUT's implementation (e.g., source or executable code). In this project, we focus on model-based security testing, a black box testing strategy that generates tests from security models (e.g., threat models and access control models).

Software security testing aims at finding vulnerabilities that may be exploited to violate security goals, such as confidentiality, integrity, and availability. Generally, it involves two perspectives: (a) testing whether or not a SUT has enforced its security policies, and (b) testing whether or not a SUT is subject to security attacks. Both are important for security assurance.

3.1 Testing Security Policy

Security policy plays an important role in security systems. For example, an access control policy regulates who can access what resources under what circumstances. It is a fundamental

mechanism for preventing malicious or accidental violation of security requirements (e.g., confidentiality and integrity). While the specification of an access control policy can be supported by powerful verification techniques, a correctly specified policy may be implemented incorrectly for various reasons, such as programming errors, omissions, misunderstanding of the policy, and intricate interplay between business logic and access control policy [14]. The defects in an incorrect access control implementation may result in serious security problems, such as unauthorized accesses and escalation of privileges. Therefore, it is important to reveal the potential discrepancy between the policy specification and the actual system implementation.

On the other hand, traditional access control metrics are inadequate for assuring system accountability, which depends on "security procedures being faithfully carried out by computer systems and by human users" [5]. The actions in a security procedure range from administrative operations to business functions. They are often an obligatory part of the humans' job responsibilities. As such, specification and monitoring of obligation policy has recently gained increasing attention [10]. An obligation policy grants access "with strings attached" - access is granted with some obligations, i.e., behavioral constraints. Existing work on obligation policy has focused on policy languages for specifying and monitoring obligation requirements. In a computer software system, however, a correctly specified obligation policy may be implemented incorrectly. Existing work on testing obligation policies takes an implementation-based approach, targets those systems that specify obligation policy as rules in a database or input file, and derives test cases from the rules. In fact, few real-world applications have yet implemented obligations as part of security policy. In this project, we aim at a model-based approach to testing access control with obligatory constraints. It is independent of how the access control and obligation polices are implemented.

3.2 Testing Security Attacks

Security attacks typically result from unintended behaviors or invalid inputs. Testing for security is hard in that it is impossible to test a real-world program against all invalid inputs. Hence, it is highly desirable to develop automated (or partially automated), cost-effective testing techniques for detecting software vulnerabilities. Although existing work has provided useful recipes for specific types of vulnerabilities, more structured and automated approaches to systematic security testing are much needed. To address this issue, we have recently developed a tool-supported approach to automated security test generation with formal threat models [15]. Threat models are represented as Predicate/Transition (PrT) nets in MISTA [12], an automated model-based test generation tool developed by the author and adopted by industry. PrT nets are high-level Petri nets, a well-studied method for modeling and verifying distributed systems. When used for threat modeling, PrT nets are more expressive than attack trees (the most popular yet informal notation for threat modeling) because, for example, they can capture both control flows and data flows and complex attacks with partially ordered actions. Threat models can provide a basis for effective security testing because they describe security threats from the standpoint of how the adversary would attack or exploit a system. This meets the need of security testing to target the "presence of an intelligent adversary bent on breaking the system" [9].

There are a great variety of security attacks, depending on the given application. Testing security attacks in this project focused on SQL injection attacks and cross-site scripting (XSS) attacks. They have been the top security threats against web applications in recent years. Because many web applications make SQL queries to their database according to user-provided inputs, such as user name and password, a SQL injection attack tries to gain unauthorized access to a web application by injecting SQL query/command into user inputs. XSS attacks inject malicious client-site script into web pages viewed by other users so as to gain unauthorized access to the information (e.g., sensitive page content and session cookie) maintained by the browser on the user's behalf.

3.3 Security Mutation

Evaluating the effectiveness of a testing technique is a challenging issue in software testing. Because testing aims at finding bugs, a widely applied evaluation approach is mutation analysis, which creates mutants of a subject program by injecting faults [6]. The faults injected into the mutants mimic the bugs that likely occur in the development process. A mutant is said to be killed if at least one of the tests created by the given testing technique reports a failure. Usually, the percentage of mutants killed by the tests is a good indicator of the testing effectiveness. Key to mutation analysis is the fault model, which is a classification of bugs in the given programming language. Fault models used in traditional mutation testing research are often syntax-oriented and faults are injected by making syntactic changes to a target program or specification, such as modification of && (and) to || (or) in a condition. Obviously, syntactical mutants are unlikely security vulnerabilities because they have not taken the semantics into consideration. To evaluate our approach to automated security testing with formal threat models, we have created security mutants according to the common vulnerabilities of web applications (e.g., SQL injection and XSS) and C++ programs (buffer overflows and logic errors) [15].

Mutation analysis has been applied to the testing of access control and obligation policies. Le Traon et al., [8] created the mutants of role-based access control rules using five types of mutation operators: replacing permission rule with prohibition, replacing prohibition rule with permission, changing role, changing context, and adding a rule. Bertolino et al. [1] proposed a mutation-based approach to testing the PolPA authorization system that is based on the usage control and history-based access control model. The Policy Decision Point (PDP) is tested with the mutants of policy rules created by applying mutation operators to the policy rules. Elrakaiby et al. [2] also proposed a mutation-based approach to testing obligation policy enforcement, where obligation rules are independent of access control rules. In the above work, the access control or obligation policy is specified as rules in a database or input file and the test cases are derived from the rules. The mutants created from the syntax of access control and obligation rules in essence represent data or configuration errors, rather than programming bugs. From the mutation analysis perspective, this project considers both data error and programming bugs.

4. PROJECT DESIGN

In this section, we introduce the objective of the project, the organization of the project team, and the research tasks.

4.1 Objective

The main objective of the project was to provide an opportunity for the selected REU students and RET teachers to gain an experience of conducting cutting-edge research in cybersecurity.

Through the experience, the students and teachers will not only learn about the advanced security techniques, but also contribute to the research base. To achieve this objective, the project was motivated by the author's newly developed techniques for threat-model-based security testing and model-based access control testing. These techniques had just been accepted for publication at prestigious venues when this project was first conceived [14] [15].

To make the project more interesting and engaging, a real-world application is critical. It can help researchers better formulate their research problems and evaluate their solutions. This consideration has lead to the adoption of Cyclos (http://project.cyclos.org/), an open source real-world online banking system being used by many organizations and communities all over the world. Using the online banking system is aligned with the goal of another funded project that aims at building a model online banking system as a research infrastructure for information assurance research and education. Banking security affects our daily life. Nowadays, more and more people rely on the Internet and mobile devices to manage financial activities. As the money is stored on and moved around the Internet, the intelligent criminal would migrate from the physical crime to the less dangerous options available online. Although the banks have made constant progress and innovation, the security threats to online banking are even greater. Falk et al. examined the web sites of 214 U.S. financial institutions and found that 76% of them suffered from at least one design flaw [3]. On the other hand, all participants of this project as a banking customer have some basic knowledge about banking and are familiar with Java, the programming language in which Cyclos is implemented. All RET teachers teach Java Programming I, Java Programming II, and AP Java Programming. Most of them also teach web design.

Cyclos is an open source online banking platform developed by STRO (Social Trade Organization). The main features include Internet banking, mobile banking, debit/credit cards and POS (Points Of Sale), e-commerce, and system administration and configuration. Cyclos is implemented based on MySQL, Tomcat, and several other Java frameworks, such as Spring (Application development framework for enterprise Java), Hibernate (for the storage and retrieval of Java domain objects via Object/Relational Mapping), and Struts (for creating web applications that utilize the Model-View-Controller architecture).

4.2 Project Team and Tasks

The project team consisted of a faculty member (the author), two REU students, four RET teachers, and two graduate assistants. The REU students were juniors from Pomona College and George State University, majoring in Computer Science. The RET teachers were from three public schools and one private school in South Dakota. The graduate assistants were third-year and first-year doctoral students with a research focus on information assurance. About two months before the REU program started, the author decided to use Cyclos as a real-world application for his research group and assigned the graduate assistants to begin learning about Cyclos. When the REU students and RET teachers joined the group, the graduate students had obtained a basic understanding about the Cyclos's functionality. For example, they were able to revise the source code as needed, re-compile and re-deploy Cyclos (this is critical to the mutation analysis) and present the basic features of Cyclos.

As Cyclos is a complex online banking platform, we chose to limit the scope of our study to the objects and activities related to loans, such as granting, viewing, repaying, and discarding a loan, creating a loan group, and adding a member of a loan group. This allowed us to focus on a manageable section of the system, which nevertheless remained broad enough to provide interesting results. Loan groups, a construct popular in the microfinance community in which a group of users are collectively responsible for a single loan, added an extra layer of complexity. In addition, we focused on the following four permission groups (i.e., roles): (a) *Full Member:* ordinary account holder; (b) *Full Broker:* broker who has read access to their clients' accounts; (c) *System Administrator*; (d) *Account Administrator:* admin with a subset of the System Administrator's permissions.

According to the research tasks, the project team was divided into the following four groups:

- Access control testing group. This group was responsible for validating Cyclos's access control policies by creating access control tests and performing the tests against the Cyclos implementation (Refer to Section 4.1 for technical background). It consisted of the third year doctoral student and one REU student;
- Obligation testing group. This group was responsible for defining obligation policies and designing and performing obligation tests (Refer to Section 4.1 for technical background). It consisted of two RET teachers;
- Attack testing group. This group was responsible for designing and performing SQL injection attacks and XSS attacks against Cyclos (Refer to Section 4.2 for technical background). It consisted of two RET teachers.
- Mutation analysis group. This group was responsible for creating security mutants of Cyclos (Refer to Section 4.3 for technical background). The mutants would be used by the testing groups for evaluating the effectiveness of their tests. The mutation analysis group consisted the first-year doctoral student and one REU student.

The 2012 REU and RET programs were held from May 28 through July 27 and from June 4 through July 27, respectively. The REU program started one week earlier than the RET program, but they ended in the same week. Before the summer programs started, the author contacted the REU students and the RET teachers and introduced reading materials for them to prepare for the summer program. The research groups were not formed until the first week of the RET program.

Beyond the research project, the REU students and RET teachers attended the following program-wide activities, which were helpful for them to perform the research project.

- A weekly three-hour class "Special Topics in Cybersecurity". This course was designed specifically for the REU and RET programs. It provides an overview of fundamental cybersecurity principles, including security goals, applied cryptography, network security, operating system security, software security, web security, malware, and digital forensics.
- Seminars such as "Conducting Publishable Research in Cybersecurity" and "Ethics in Research".
- Field trips to the Symantec (a well-known security company) site in Minneapolis and the Network and

Security Department of USGS/EROS (Earth Resources Observation Systems) data center.

- Weekly REU or RET group meetings with the program director (i.e., the author).

The activities for this particular project included performing individual research tasks, holding group discussions, and attending weekly team meetings. During weekly team meetings, each member or group was required to update their progress, discuss the main research issues, and present a work plan for the next week. A common task for all project participants was to understand the related functional features and security goals of Cyclos. To this end, we built a model of the loan accounts according to the various access control activities and obligatory actions by the four roles. The model was represented by a PrT net in MISTA. Part of the model is included in the appendix.

To assure the objectivity of mutation testing, the mutation analysis group worked independently of the testing groups. In other words, they should not share with the testing groups any information about how they created the security mutants. The testing groups should not share with the mutation analysis group any information about how their tests were created. A few weeks later, we realized that the mutation analysis of Cyclos is too challenging. It requires a very good understanding about the complex Java source code of Cyclos as well as various plausible security vulnerabilities in several Java frameworks (Spring, Hibernate, Struts). Thus, we decided to limit the mutation analysis to access control policy and hold a competition between the access control testing group and the mutation analysis group. As an agreement negotiated by the two groups, the access control testing group would be the winner if their tests kill 60% or more of the mutants created by the mutation analysis group, otherwise the mutation analysis group would be the winner. Each member of the winning group will receive a 32GB USB drive as the prize.

5. RESULTS

In the first 3-4 weeks, all groups focused on understanding the related functionality and security requirements of Cyclos and learning about the principles of relevant security techniques (e.g., security testing with threat models and access control testing). Then they started creating their own artifacts.

The access control testing group created 47 access control tests according to the loan activities and access control metrics related to full member, full broker, account administrator, and system administrator. Each of these tests consists of a manually executable test script and an oracle value at one or more steps with the script. The test scripts are specified at the user interface level and thus performed from the user's perspective.

Due to the limited time and the complexity of Cyclos, the mutation analysis group focused on the mutation of access control rules. They created 10 security mutants by injecting access control faults into the relevant code or configuration. 5 of them have a programming error (e.g., missing access control check) and the rest involve incorrect privileges in the configuration database.

The access control testing group won the competition because their access control tests killed 90% of the security mutants created by the mutation analysis group. In fact, their tests would have killed the remaining mutant (mutant #5) if they had checked the oracle values carefully. In this mutant, the incorrect privilege for members has led to an unexpected change of the member's user interface. The access control testing group did not notice this change. In fact, this reflects one of the major reasons that software bugs go undetected – "looking, but not seeing".

The obligation testing group formulated five detailed obligation requirements and created scripts for testing these obligation requirements. Each of the scripts may yield multiple concrete tests with different parameters. A sample obligation is that, after an administrator has granted a loan to a member, the system should create a loan account for the member and the member should repay the loan within a period of time, otherwise a late fee will be charged (the duration and the amount of late fee are loan parameters, which are determined when the loan is granted).

The attack testing group identified numerous user input fields that are potentially subject to SQL injection and XSS attacks. For example, user name and password are two input fields, which lead to SQL queries to the database. We need to test whether SQL injection attacks can be launched through these inputs.

Based on the research activities, the REU students and RET teachers produced the following deliverables: (a) each REU student or RET pair made two presentations to the entire REU or RET group – one was about their project plan and the other was about their research result; (b) each REU student or RET pair wrote a paper on their project in the IEEE Proceedings format. (c) each RET teacher made a presentation on their teaching experience and a plan to integrate cybersecurity topics into their classroom activities. For example, most of them would introduce the concept of secure programming into their programming class.

In brief, the project has successfully achieved its goals. First, each REU student and RET teacher has learned about one of the specific security testing techniques. These techniques were newly developed and accepted for publication in prestigious venues. Second, most of them were able to make research contributions, which resulted in a paper accepted for publication and presentation by the 2013 *International Conference on Computing, Networking and Communications* [13]. To the best of our knowledge, the obligation testing was the first attempt to apply a model-based approach to the generation of obligation tests. The mutation analysis introduced access control bugs into the source code of Cyclos. This is different from the existing work on access control mutation that created mutants of access control rules according to their syntactic components.

There are several lessons that we have learned from this project. First, as the summer programs were short, preparation before the summer programs could be very helpful. The research project involved several subjects that were new to the participants. Getting some background knowledge related to the research project before the summer program would accelerate the warm-up process. Second, the graduate assistants played an important role in the project. They provided professional help to the REU students and RET teachers so that the students and teachers could work more effectively toward achieving their research goal. The graduate assistants also served an immediate role model for the undergraduates and an indirect role model for high school students through the teachers. Third, teamwork with both collaboration and competition made the research experience more effective and interesting. Through collaboration, the participants obtained a common understanding about the relevant functionality and security requirements of Cyclos. Through competition, they had a better sense of urgency, achievement, and satisfaction. Last, but not the least, Cyclos, as a real-world online banking system, intrigued the students' and teachers' interest and curiosity to investigate security issues.

6. CONCLUSIONS

We have presented a unique summer research project that involved undergraduate students from outside the host institution and high school computer teachers. Under the supervision of the faculty mentor and in collaboration with graduate assistants, they investigated several security testing issues of a real-world online banking system. Through the research activities, they have not only learned about advanced security testing techniques but also make publishable contributions to the research base. As the results were very encouraging, this project can be used as a model of summer programs for undergraduate students and/or K-12 teachers to gain research experiences.

7. ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation (NSF) under grants CNS 1004843, CNS 1200648, and CNS 1123220.

8. REFERENCES

[1] Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E., Martinelli, F., Mori, P. 2012. Testing of PolPA authorization systems. In *Proc. of 7th International Workshop on Automation of Software Test* (Zurich, Switzerland, June 2-3, 2012). AST'12. 8-14.

[2] Elrakaiby, Y., Mouelhi, T., Le Traon, Y. 2012. Testing obligation policy enforcement using mutation analysis. In *Proc. of the 7th International Workshop on Mutation Analysis* (Montréal, Canada, April 2012). 673-680.

[3] Falk, L., Prakash, A., Borders, K. 2008. Analyzing websites for user-visible security design flaws. In *Proc. of the Symposium on Usable Security and Privacy* (Pittsburg, USA, July 2008). SOUP'08. 117-126.

[4] Hoglund, G. and McGraw, G. 2004. *Exploiting Software: How to Break Code.* Addison-Wesley.

[5] Irwin, K., Yu, T., Winsborough, W.H. 2006. On the modeling and analysis of obligations. In *Proc. of the 13th ACM Conf. on Computer and Communications Security* (Chicago, USA, October 2006). CCS'06. 134-143.

[6] Jia, Y. and Harman, M. 2010. An analysis and survey of the development of mutation testing. *IEEE Trans. on Software Engineering*, 37, 5 (October 2010). 649-678.

[7] Langevin, J.R., McCaul, M.T., Charney, S. Raduege, H., Lewis, J.A. 2010. *A Human Capital Crisis in Cybersecurity: A White Paper of the CSIS commission on Cybersecurity for the 44th Presidency*, Center for Strategic & International Studies.

[8] Le Traon, Y., Mouelhi, T., Pretschner, A., and Baudry, B. 2008. Test-driven assessment of access control in legacy applications. In *Proc. of the First IEEE International Conference on Software, Testing, Verification and Validation* (Lillehammer, Norway, April 2008). ICST'08. 238-247.

[9] Potter, B., Allen, B., Mcgraw, G. 2004. Software security testing. *IEEE Security & Privacy*, (Sept. 2004). 32–36.

[10] Xu, C., Fong, P. W. L. 2012. The specification and compilation of obligation policies for program monitoring. In *Proc. of the 7th ACM Symposium on Information, Computer and Communications Security* (Seoul, South Korea, May 2012). ASIACCS'12.

[11] Xu, D. 2009. Software security, *Wiley Encyclopedia of Computer Science and Engineering*, B. W. Wah (Editor-In-Chief), Volume 5, John Wiley & Sons, Inc., Hoboken, NJ. 2703-2716.

[12] Xu, D. 2011. A tool for automated test code generation from high-level Petri nets. In *Proc. of Petri Nets'11* (Newcastle upon Tyne, UK, June 2011), LNCS 6709, 308-317.

[13] Xu, D., Sanford, M., Liu, Z., Emry, M., Brockmueller, B., Johnson, S., To, M. 2012. Testing access control and obligation policies. In *Proc. of the 2013 International Conference on Computing, Networking and Communications* (San Diego, January 2013). ICNC'13.

[14] Xu, D., Thomas, L., Kent, M., Mouelhi, T., and Le Traon, Y. 2012. A model-based approach to automated testing of access control policies. In *Proc. of the 17th ACM Symposium on Access Control Models and Technologies* (Newark, USA, June 2012), SACMAT'12. ACM. 209-218.

[15] Xu, D., Tu, M., Sanford, M., Thomas, L., Woodraska, D., and Xu, W. 2012. Automated security test generation with formal threat models. *IEEE Trans. on Dependable and Secure Computing*. 9, 4 (July/August 2012), 525-539.

Appendix. Partial test model of loan accounts in Cyclos [13]