12-1-2017

A Survey of Security Patterns for Use in Software Development

Catherine Ann Gibbs

A Survey of Security Patterns for Use in Software Development

by

Catherine Ann Gibbs

An Honors Capstone
submitted in partial fulfillment of the requirements
for the Honors Certificate
to

The Honors College

of

The University of Alabama in Huntsville

1 December 2017

Honors Capstone Director: Dr. Letha H. Etzkorn
Professor of Computer Science

[illegible]	29 NOV 17
Student (signature)	Date
[illegible]	11/29/17
Director (signature)	Date
[illegible]	11-29-17
Department Chair (signature)	Date
[illegible]	11-30-17
Honors College Dean (signature)	Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

CATHERINE A GIBBS
Student Name (printed)

[illegible]
Student Signature

29 NOV 17
Date

Table of Contents

Abstract

Computer scientists are the creative minds behind software development. Unfortunately, only a select few will get to develop something purely new. The vast majority of software practitioners will work to adapt existing software. Dealing with legacy software has many challenges. One of which is security holes in the code. As a computer language develops, patterns emerge and become coding practices. From a security perspective, these are called prescriptive security patterns (good programming practices) and anti-patterns (bad programming practices).

There are two ways in which prescriptive security patterns and anti-patterns can be used. One way is to use them while one is building software (*forward engineering*), to make sure the software is developed correctly. The other way is to use them to determine whether already-existing software is secure or not (*reverse engineering*). The former is what we are learning in our computer science curriculum. But, the latter is difficult to do manually because software can be millions of lines of code, particularly in the case of legacy systems which may have been added to every year for many years.

The purpose of this study is to review what research is being conducted regarding security patterns and provide possible areas for future work. It is hoped that this will assist in a thesis topic for follow-on work at the master's level.

Introduction

Software is continuing to grow in sheer size and complexity. There is more interconnection between electronic devices and various networks, to include the internet, than ever before. Proliferation of mobile devices, cloud services, and distributed networks have blurred the traditional lines of computing and networks with other realms of telecommunication services.

Increases of known attacks and security systems breaches have all caused software security to shift from an afterthought to key development parameter in the software lifecycle. The traditional “walled-garden” security concepts are ineffective because it is no longer possible to build a wall between your system or network and the outer world (Hamid, Gürgens, Fuchs 2016, 109).

We need a better way to secure these complex systems. Computer scientists are the creative minds behind software development. However, not all computer scientists are also security experts. This does not mean we are doomed to failure. Security patterns can help address security issues in software in a more holistic way.

Use of Patterns in Software Development

Patterns in software development are descriptions on how to solve common or reoccurring problems. They grow out of the best practices in the industry and, if properly implemented, have a positive effect on software quality (Bunke 2015, 3). There are several types of patterns in use in the software community. Analysis patterns are used to create conceptual models, design and architectural patterns are used to build software and security patterns are used to ensure secure systems (Fernandez 2013, 8).

In general, patterns provide three main advantages. First, the solution grows from the best practices of the industry. This is to say that the solution is time tested and proven to help. Second, all the advantages and disadvantages of the pattern's approach to a problem are known. This mean all aspects of the patterns can be considered while building the design. Third, using patterns creates a common terminology to simplify discussions (Heyman 2008, 1156).

Although there is no standard for pattern descriptions each must address the context, problem, forces, and solution (Dwivdei 2015, 3). Some of the best-known examples of software patterns are the collection of design patterns introduced to the software development community in 1994 by the Gang of Four (GoF) to solve recurring object-oriented programming issues. The other well-known software patterns publication is the book "Pattern-Oriented Software Architecture" known as POSA by Buschmann, originally published in 1996. Their popularity and impacts still heavily influence software development today.

Security Patterns

Security patterns are another subset of the patterns. Security patterns are like the design patterns described above in that they also provide a description to solve recurring problems in software development. But security patterns tend to be larger in scope than design patterns (Fernandez 2013, 6). So, choosing the appropriate patterns and implementing them into the software can be a significant challenge. Also, not all security patterns deal purely with software. There are five different application domains for security patterns (Bunke 2015, 3). In addition to the software application domain, security patterns can pertain to enterprise, cryptographic, user, and network application domains.

One of the primary advantages of security patterns is that it can bridge the gaps between software developers and security experts. But, they can do much more. Fernandez noted that, they could assist security experts apply security in a more systematic way as well as build new applications, audit complex applications and reengineer legacy software (Fernandez 2013, xix).

Security patterns were originally introduced in 1997 by Joseph Yoder and Jeffrey Barcalow. Their research discussed the concept of security patterns for application security. They stated a primary need for the framework was that developers are more focused on satisfying the customer's desires than on securing the system (Yoder and Barcalow 1997, 2). And, as such, security is generally an afterthought instead of a primary design concern. Their work introduced 7 patterns: Secure Access Layer, Single Access Point, Check Point, Roles, Limited View, and Full View with Errors.

These patterns were designed to work together to provide security framework for applications. This is generally how security patterns are implemented. No pattern is really mutually exclusive. A single pattern may solve a "low-level" problem. More typically, a combination of patterns are used to solve a "high-level" problem (Nguyen 2015, 247). This is why security patterns are described as a framework. By incorporating a security framework in the design, the details could be implemented late in development or expanded easily as required. But, if the framework is left out of the initial design, it could become a large issue to incorporate the security patterns later in the software development (Yoder and Barcalow 1997, 27).

The primary goal of their work was to harden software or infrastructure against common attacks and misuse. This continues to be the driving motivation of security pattern designs today. The patterns introduced by Gof, POSA and Yoder and Barcalow were just a starting point for pattern development. Today several hundred patterns have been published in various book and journals. In a recent publication search over 415 software patterns were found, of which 166 can be considered security patterns (Bunke 2014, 1).

Adoption Issues

Overall, security patterns have not been as well adopted across the software community. They are found more in academia than in the software industry. There are several reasons attributed to this.

Security patterns were published in different publications spanning several years. Design patterns were published as a single textbook and is usually taught as a course in computer science curriculum (Bunke 2015, 2). There is still no standard for what a pattern description should contain. This has led to security patterns to be published at various times under different names, but describing the identical issue and similar solutions (Bunke 2015, 3). Security patterns overall have less actual code implementation examples. The lack of working examples and automated developer tools to assist with implementation is another reason security patterns are less adopted than design patterns (Hamid 2016, 110). Lack of security pattern adoption has also been attributed to the cross disciplines of security specialist and software development. Finally, the security pattern testing is difficult to apply metrics and systematically test if the security pattern is implemented correctly (Duncan and Muijnck-Hughes 2014, 1).

Paper Structure

The aim of this paper is to give a survey of the existing research regarding security patterns to improve software development or software security. Section 2 focuses on the review process and overall findings. Section 3 presents a detailed analysis of the current research in security patterns. Section 4 discusses future work. Section 5 concludes the study.

Literature Review

Security patterns have been a popular area of research for 20 years. Earlier research focused on describing specific patterns or introducing new patterns. As patterns increased research revolved around organizing and selecting appropriate patters for various environments. There is still much discussion on how to standardize the descriptions across the community. Another area of research is on the implementation concepts and examples. These tend to be design oriented and very few have a fully developed implementation with code examples. To ease the difficulties of implementation some researchers have focused on developing automated tools. Overall, the Internet of Things (IoT) and cloud computing has provided the stimulus needed to re-examination of the use of security patterns in many of the above mentioned research areas.

Although many researchers have published works about security patterns, two stood out: Eduardo B. Fernandez and Michaela Bunke. Eduardo B. Fernandez, a professor of computer science at Florida Atlantic University, seems to be the principal champion of security patterns in software development. He has authored or co-authored numerous works as well as two books on the subject. Michaela Bunke, from Bremen University in Germany, has published some of the most in-depth work research works promoting understanding and use of security patterns over the past few years.

Research Approach

The approach used to conduct this survey was to search for research papers that discussed the use of security patterns in software development or application security in the past five years.

The search was limited to the following digital libraries: ACM Digital Library, IEEE Xplore Digital Library, and Science Direct Library.

A very simple inclusion and exclusion criteria was used. Any articles that discussed the use of security patterns in software development or application security was included. Any article that focused on design patterns or architectural patterns instead of security patterns was excluded. Also excluded were any articles that were not in English and any article that did not actually deal with software. Although, no articles were excluded based on date of publication, the focus was to find relevant research in the period of 2013 to mid-2017.

Overall Findings

	2013	2014	2015	2016	2017	Total
New Patterns		4	2	3		9
Organization of Patterns		3	2	3		8
Modeling and Metrics	3	2		1		6
Implementation		2	5	2	1	10
Tools/Automation	2	1	1	2		6
Other			2			2
Total Research Papers	5	12	12	11	1	41

Basic Trends

The amount of research in the field has stayed consistent over the last few years. In reviewing the various research papers, I noticed that some researchers view security from a systems viewpoint. Others from a secure coding or software viewpoint. And, still others view security from more of an architecture viewpoint. Many researchers identified the same requirements: standardized descriptions; implementation validation, meta-models or frameworks; code examples; and tools. There is still a lot of work to be done in the security pattern field.

Detailed Analysis

41 articles were reviewed. To simplify the analysis, the papers were broken into the following categories: new pattern descriptions; organization of security patterns; modeling and metrics; implementation; tools or automation; and other.

New Patterns

New patterns continue to be introduced or expanded upon. The most recent years are no exception. There were nine new patterns introduced in the past five years. Two discuss new patterns for Internet of Things or cloud environments. Several others expand on existing patterns or purposed a unique twist on the concept of security pattern use. Below is a summary description of each.

Jose Carlos Ciria and colleagues purposed a new twist on the authentication security pattern (Ciria et al.2014). Their concept was to strengthen the authentication pattern by basing it on the capability of an entity to satisfactorily account for its history in relation to the Internet of Things (IoT). Authentication is the process of an entity proving that it is who it claims to be. Usually authentication processes group things by factors - something you have, something you know, or something you are. The more factors required, the stronger the authentication process. This work discusses the concept of self-management and traceability of a smart system to be authenticated by extending the context of what it knows about itself. It is a specialization of the Authenticator Abstract Security Pattern proposed by Fernandez.

Eduardo B. Fernandez purposed a work that describes how to adjust security patterns to make them more specific to cloud environments (Fernandez 2015). This work builds a reference architecture to evaluate cloud security. As threats are identified, the corresponding security pattern can be applied to the reference architecture to mitigate the threat. The model is still abstract and is not a direct implementation, but it provides the basis to understand how security patterns fit into complex cloud architectures.

Fernandez and colleagues also purposed in another work the new Context-enhanced access control pattern that expands authorization so that entities access to data is restricted to relevant functions or context (Fernandez 2014).

Tobias Rauter and colleagues introduce 2 new patterns: Integrity Protection and Integrity Attestation to protect a software module (Rauter et al. 2015). These are described in a client server scenario. The first pattern allows a software module to proactively protect itself. The second pattern protects against forgery by enabling a software module to prove it maintains an integrity state with a challenge – response procedure.

Tobias Rauter and colleagues went on to introduce 2 more patterns: Static Integrity Properties and Dynamic Integrity Properties (Rauter et al. 2016). These are described in terms of a complex system which is made up of many components. By analyzing fault taxonomies of complex systems for integrity violation that lead to faults, the team builds upon their previous work to create two new security patterns. These are designed to ensure the integrity of a component before it is used in a complex system.

Mohamed Adel Sheta and colleagues propose a new security pattern to detect spyware (Sheta et al. 2016). The new pattern combines data mining techniques and design patterns to create a framework to detect and classify spyware. The focus of this paper was on the anti-spyware experiment rather than security patterns or the framework created, but it is very interesting. The framework is composed of 3 layers. On the lowest level is a design pattern, the middle layer is the security pattern and the final layer is the anti-spyware layer. The anti-spyware was run on several files and the results compared.

Research by Rohini Sulatycki and Eduardo B. Fernandez, built upon other security pattern research, specifically, on misuse patterns and threat patterns (Sulatycki and Fernandez 2015). Two new threat security patterns are described. Threat patterns describe the steps of an attacker from the point of view of the attacker. Then they go on to introduce countermeasures and forensic traces. The new patterns describe attacks based on security misconfiguration and data exposure.

Kazi Zakia Sultana and colleagues considered the relationship between coding patterns and security defects (Sultana et al. 2016). This very unique study shows that certain Nano-patterns are present in vulnerable code. Therefore, screening code for the Nano-patterns can improve vulnerability testing. The team hopes to grow the patterns and granularity of the vulnerabilities in future works.

Xiaohong Yuan and colleagues describe their ongoing effort to create a tool called pull attack patterns called Common Attack Pattern Enumeration and Classification (CAPEC) for software development (Yuan et al. 2014). It is the teams view that to create more secure software, developers must think like attackers. Currently, they are build the database of patterns and mapping them to Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege (STRIDE) threat categories. This is a new way to create and use security patterns.

Organization of Patterns

As larger and larger sets of security patterns are developed by security experts, the need to classify and analyze them has also grown. There are several different ideas on how this should be accomplished. Some are very novel approaches, but the most comprehensive is Michaela Bunke's work which maps a large set of existing security pattern descriptions. Below is a summary of the eight papers on organization of security patterns.

Priya Anand and colleagues' paper discussed the categorization of security patterns based on vulnerabilities to help software developers select appropriate security patterns and identify missing patterns (Anand, Ryoo, Kazman 2014).

Michaela Bunke, published a research work that compared existing security pattern descriptions against common GoF and POSA design pattern templates (Bunke 2014). The stated goal of the work was to encourage the software community to reach an agreement on a core description sections. The research built on a previous works which collected a literature review of security patterns from 1997 to mid-2012. Analysis showed that pattern description contained anywhere from three to over 15 sections and provided a detailed mapping of the characteristics of each pattern description.

Anas Motii and colleagues propose a security pattern selection method based on classification according to properties and application domain (Motii et al. 2015). The process is guided by security risk management.

Yasser M. Hausawi's research focused in a broader sense on improving the selection of patterns and tools between security, usability and usable-security to ensure design concepts map to actual requirements (Hausawi and Allen 2014). To illustrate the ideas, a detailed analysis of a choice-based authentication system is modeled with a trade-off matrix to identify conflicts between security and usability.

Pooname Ponde contributed two almost identical papers on security pattern organization using statistical analysis of clusters (Ponde, Shirwaikar, Kreiner 2016), (Ponde, Shirwaikar, Gore 2016). The complex analysis of existing pattern descriptions that uses hierarchical clustering analytical tool to create groups of similar data. First a list patterns based on a chosen set of attributes were collected. Then hierarchical clusters were applied to extract groups of related patterns. The resulting groups were cut into high-level and low-level subcategories and the relationships were then interpreted based on the dominant attributes of each cluster.

Loukmen Regainia and colleagues propose a methodology of classification for security patterns by establishing a relationship among security weaknesses, security principles and security patterns (Regainia, Salva, Ecuhcurs 2016). This identifies groups of patterns that work in combination to remove a given weakness. The research is limited to the analysis of web applications. It includes 185 software weaknesses from the Common Weakness Enumeration (CWE) database, 66 security principles and 26 security patterns.

Like the other works by Anton V. Uzunov and colleagues this research outlines not just a detailed security framework using security patterns but a meta-model for engineering security methodologies (Uzunov, Falkner, Fernandez 2015). This work is designed to be a “fit-to-purpose” organization for software development that can be tailored to specific needs of a project. The work utilizes many of the ideas from the related field of method engineering. It is unique approach to finding an overarching framework to develop secure software using security patterns.

Modeling and Metrics

The ability to validate that a security y pattern is implemented correctly has ever been an elusive goal. The need to assure that software is built securely is becoming ever more important. Below are quick summaries of the current six works in this area.

Ishbel Duncan and Jan de Muijnck-Hughes purposed the development of a dynamic testbed to quantitatively measure and compare security patterns (Duncan and Muijnck-Hughes 2014). Their work is guided by the premise that user feedback cannot provide the level of trust in security patterns. The software community needs assurance that the patterns are defined in such a way that erroneous variation is removed or minimized and quantified metrics can help. Essentially, they purpose to merge existing code testing and evaluation techniques with higher level pattern design indicators. It discusses the software life-cycle, pattern complexity and current evaluation methods.

Brahim Hamid and colleagues purposed a new framework for the specification and validation of security patterns (Hamid, Gürgens, Fuchs 2015). The framework would provide two complimentary representation: a semi-formal meta-model and a formal model in a prefix-closed formal language for security patterns. A conceptual model is presented for a smart power grid related to cars.

Takanori Kobashi and colleagues discussed a method to validate security pattern applications (Kobashi et al. 2013). First threats and vulnerabilities in the target system are identified. Then the testbed validates that the security pattern is properly applied and assess if the vulnerabilities are resolved. These are limited to the design phase only and uses UML diagrams for the validation.

Vaise Patu's research discusses software security assurance as the ability to present an assurance to a system's stakeholders that it meets security and dependability requirements (Patu and Yamamoto 2013). The work outlines how to begin developing a dependability case, called D-case against the various security patterns available for use in a system. It briefly discusses manual vs automatic security pattern selection techniques in terms of validity and scalability as well as security pattern repositories.

Paul Rimba described his on-going a PhD research about building secure applications with security patterns (Rimba 2013). It identifies a gap between the generic security pattern and the underlying platform specifications. The work is focused on platforms that adhere to a capability-based security model. Specifically, the research models the designs using the Serscis Access Modeller (SAM). The stated goal of the research is to develop a collection of reusable design fragments for security patterns. Each fragment would be evaluated for feasibility and correctness. Then tested using a security property analysis.

Masatoshi Yoshizawa and colleagues propose an evaluation method for testing security design patterns (Yoshizawa et al. 2014). Their concept is that one of the biggest issues in using security patterns is the improper implementation. Therefore, a test template is used to observe internal processing to evaluate the system in the initial stages of implementation. Borrowing from the concept of test-driven development, this can validate if the security pattern has been appropriately applied. The examples are created using AspectJ.

Implementation

This category describes design implantation models. None show any code implementations, but many elaborately explain the designs. Here again we see several that deal with IoT, cloud service or distributed architectures. A summary of each of the 10 research papers in this category is below.

Priya Anand and colleagues discuss cloud security issues. It recommends a cloud security framework that is based on the use of security patterns (Anand, Ryoo, Kim 2015). They stress that there is currently no classification or organization of security patterns for cloud computing environments. As a case in point, they select a set of patterns and designed one such pattern-based security framework for cloud computing environment. The model developed is focused on cloud security challenges at the network level.

Ashish Kumar Dwivedi and colleagues purposed how to use security patterns to add security features in a Service-Oriented Architecture (SOA) in their research (Dwivedi, Kumar, Rath 2015). SOA is an architecture in which different components share information using web services. The paper does an impressive job of conceptually laying out the architectural model based on SOA architecture designs from OASIS group and showing how security patterns fit into the mix. An example implementation is given in the form of an online banking service with six security patterns implemented – Identification and Access Management, Check Point, Data Confidentiality, Policy, Proxy-based Firewall, and Secure Channel.

Eduardo Fernandez and Monge also modeled a cloud-based architecture implementation with security patterns (Fernandez, Monge 2014). The paper discusses the basic cloud architecture and interaction of its components. The detailed architectural representation builds upon previous work’s Cloud Access Security Brokers (CASBs) as security enforcement points between consumers and service providers.

Wen-Tin Lee and Po-Jen Law also discussed implementing security patterns in and IoT environment (Lee and Law 2017). The work briefly outlined how to implement secure directory, secure logger exception manager patterns to address unsecure application data and secure adapter and input validation patterns to address unsecure wireless. It has some nice diagrams but does not really discuss the merits of how these implementations reduce or resolve security issues.

Santiago Moral-Garcia and colleagues purpose how to implement security patterns across a whole organization by using an Enterprise security pattern (Moral-García et al. 2014). This integrates security patterns more comprehensively to handle a greater number of threats. The proposed solution is a model-driven architecture example comprised of a computationally independent model, a platform independent model, a platform specific model and a product dependent model. The paper walks through each model in relation to Software as a Service (SaaS) concept. It makes some good point about the limitation of implementing security patterns on an individual basis.

Anas Motii and colleagues describe the implementation of a model-based approach for selecting between security patterns alternatives relative to cyber-physical systems (Motii et al. 2016). The work is centered on a meta-model for describing Security and Dependability patterns called the System and Software Engineering Pattern Meta-model (SEPM).

Hans-Joachim Hof created a short poster that illustrate examples of security patterns not used properly (Hof and Socher 2016).

Phu H. Nguyen and colleagues purpose a new model-driven security (MDS) framework based on a system of security design patterns (SoSPa) (Nguyen et al. 2015). This weaves multiple security patterns into the design of a system. The idea is to detect and resolve conflicts and inconsistencies among the applied security solutions during the design phase. The meta-model extends reusable aspect models (RAM) which is what the paper intricately describes with use cases.

Anton V. Uzunov and colleagues designed a methodology for distributed systems called ASE (Uzunov, Fernandez, Falkner 2014). This work describes in detail the analysis and design of an example system called SHAring and collaborative Editing environment (SHARED). The example system is used to model most of the core distributed systems security concerns.

Anton V. Uzunov and colleagues also designed an authorization framework, called a security solution frame, for distributive collaborative systems (Uzunov, Fernandez, Falkner 2015). This builds upon previous security patterns. Specifically, the "product" pattern and security micro-process patterns.

Tools and Automation

Most research on the application of security patterns focus on the design level and use UML for design implementations. Very few provide any mechanisms for implementing full integration of a security pattern from concept to final code. The six papers that do provide some code or automation research for security patterns in recent years are listed here. A summary of the each is below.

Rahma Bouaziz and colleague outlined an engineering process called SCRIP (SeCurity PatteRn Integration Process) to provide a guideline for integrating security patterns into component-based models (Bouaziz, Kallel, Coulette 2013). The stated purpose was to help software developers improve capabilities to deploy security pattern solutions.

Bouaziz went on to build upon the SCRIP tool and introduced a SCRIStudio tool which gives developers a way to implement security patterns in component based applications based on UML language (Bouaziz, Kammoun 2016).

Takanori Kobashi and colleagues built upon their earlier research to implement a design tool TESEM (Test Driven Secure Modeling Tool) which supports pattern application in the design modeling (Kobashi et al. 2015). The tool verifies whether the security patterns are properly applied and resolves the vulnerabilities specified. To achieve this, the research team added Object Constraint Language (OCL) expressions to current security patterns. The testbed then created a script to execute the model using UML diagram format.

Mehdi Mirakhorli and colleagues presented a tool in 2014 called Archie. It is specific to the Java Language and is designed as a plugin for Eclipse IDE (Mirakhorli et al. 2014). The open-source tool in available from Github. The work discusses how to trace linkage between the tactics, design models, and source code. It is built around the concept of a Tactic Traceability Pattern (TTP). Once mapped, the sourcecode is monitored for any changes. When a change is detected, a notification is sent.

Joseph Near and Daniel Jackson created a tool called SPACE (Security PAttern CheckEr) which uses access control patterns to identify application-specific security errors in source code (Near and Jackson 2016). The implementation uses of a custom-built catalog of access control patterns created for Ruby on Rails web applications. The patterns constrain the relations of generic RBAC model to only allow certain behaviors. The team describes this as an abstracted "whitelist" of sorts for data access. The tool was then used on the 50 most watched open-source Rails applications on Github and the results analyzed.

Adel Ziani and colleagues propose Pattern-based System Engineering (PBSE) for the development of Resource Constrained Embedded Systems(RCES) (Ziani et al. 2013). The work includes a design framework dealing with modeling and patterns, and a repository for platform implementation. The Gaya repository is implemented using Java and Eclipse CDO Server technology. Although this is a heavy engineering perspective the overall approach is like other security pattern research

Other

The last two research works do not fit neatly into any of the above categories but are excellent research examples.

Michaela Bunke discussed adoption issues with security pattern in software engineering (Bunke 2015). This paper compared the degree of maturity of security patterns by comparing them to design patterns in terms of pattern terminology, classification, description form, and usage in the software lifecycle using UML diagrams and code examples. This work goes into detail on the use of security patterns in the various phases of software life cycle

Koen Yskout and colleagues conducted a study to confirm or refute the alleged benefits of security patterns (Yskout, Scandariato, Joosen 2015). The study was designed to see if the use of security patterns creates more secure software and if the use of security patterns helps improve designer productivity. Using a catalog of 36 security patterns; 30 teams were given an initial design of a banking system asked to improve the security through 7 tasks. Half of the teams are given catalog of patterns; the other teams are not. The work and solutions were evaluated for effectiveness and productivity.

Future Work

One purpose for this work was to explore the possible research opportunities in security patterns in relation to software development. The results were surprising. It shows that there are many venues of research possible for the area of security patterns. It also shows that any implementation project is probably going to be very labor intensive.

One overarching themes that was seen in over a third of the papers was the need for a framework of security patterns. This echo's back to the first descriptions of security patterns by Joseph Yoder and Jeffrey Barcalow. Security patterns are rarely singletons. Instead several are combined to achieve a security feature. I can envision a security pattern solution that works like a web application framework do now. The developer would just specify the type of security features desired and the security framework would "plug-in" most of the code necessary to implement it.

Another common reference was to IoT, cloud or distributed collaborative system security solutions. The use of security patterns in this area is an expanding new research venue across many categories. Currently, the research is focused on new pattern description and design implementation illustrations. But, larger projects are coming soon.

The need for functional code examples that introduce correct security pattern application is needed across all programming languages. This area seems to still be in its infancy. The only actual code example in the last five years were for Ruby on Rail or Java and even these were really limited. This area would be excellent for a project, but also riskier and larger in scope of work necessary to complete.

Conclusion

The proliferation of mobile devices, cloud services and distributed networks have blurred the traditional computing lines. Software is continuing to grow in sheer size and complexity as is the amount of data collected about users. This connectivity and usage has become an integral part of our daily lives. This has also led to an increase in known hacking attacks and system security breaches. Vulnerable software can no longer be tolerated, better ways to secure complex systems are needed.

In looking into the use of security pattern for software development, we can see many basic trends, perceived benefits to using security pattern, and problems in adoption of the use of security patterns in software development. The perceived benefits of using security patterns are: time-tested solutions to recurring problems, pattern capabilities and limitation are known in advance, creates a common terminology, allows non-security specialist to implement security solutions. But, security patterns have not been adopted as well as design patterns have regarding software development. There are many reasons for this, but the issue that stuck out the most was that security patterns do not tend to be single patterns. More generally, multiple patterns are used in combination to create a security framework. This adds complexity to the design and implementation combinations possible, making it more difficult to use security patterns.

There is much more work to be done in the security pattern research fields. Research is actively ongoing in many areas to include new pattern designs, organization of patterns, design implementations, testing, metrics, tool and automation. But, the benefits of applying security to software in a systematic and holistic way are worth the efforts.

Reference List

Anand, Priya, Jungwoo Ryoo, and Hyoungshick Kim. "Addressing Security Challenges in Cloud Computing — A Pattern-Based Approach." *2015 1st International Conference on Software Security and Assurance (ICSSA)*, 2015.

Anand, Priya, Jungwoo Ryoo and Rick Kazman. "Vulnerability-Based Security Pattern Categorization in Search of Missing Patterns," *2014 Ninth International Conference on Availability, Reliability and Security*, Fribourg, 2014, pp. 476-483.

Bouaziz, Rahma and Slim Kammoun. "SCRIStUDIO: A security pattern integration tool," *2016 International Conference on Information Technology for Organizations Development (IT4OD)*, Fez, 2016, pp. 1-6.

Bouaziz, Rahma, Slim Kallel, Bernard Coulette. "An Engineering Process for Security Patterns Application in Component Based Models." *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on*, 2013, pp 231-236.

Bunke, Michaela. "Software-security patterns: degree of maturity." *Proceedings of the 20th European Conference on Pattern Languages of Programs - EuroPLoP 15*, 2015.

Bunke, Michaela. "On the description of software security patterns." *Proceedings of the 19th European Conference on Pattern Languages of Programs*, 2014.

Ciria, José Carlos, Eladio Domínguez, Inés Escario, Ángel Francés, María Jesús Lapeña, and María Antonia Zapata. "The history-based authentication pattern." In *Proceedings of the 19th European Conference on Pattern Languages of Programs* (EuroPLoP '14). ACM, 2014, New York, NY, USA, Article 30, 9 pages.

Duncan, Ishbel, and Jan De Muijnck-Hughes. "Security Pattern Evaluation." *2014 IEEE 8th International Symposium on Service Oriented System Engineering*, Oxford, 2014, pp. 428-429.

Dwivedi, Ashish Kumar, and Santanu Kumar Rath. "Incorporating Security Features in Service-Oriented Architecture using Security Patterns." *ACM SIGSOFT Software Engineering Notes 40,* no. 1 (2015): 1-6.

Fernandez, Eduardo B. *Security patterns in practice: designing secure architectures using software patterns.* Chichester: Wiley, 2013.

Fernandez, Eduardo B., Nobukazu Yoshioka, and Hironori Washizaki. "Patterns for security and privacy in cloud ecosystems." *2015 IEEE 2nd Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE)*, Ottawa, ON, 2015, pp. 13-18.

Fernández, Eduardo B., A. Raúl Monge, René Carvajal, Oscar Encina, Juan Hernández, and Paulina Silva. "Patterns for content-dependent and context-enhanced authorization." *Proceedings of the 19th European Conference on Pattern Languages of Programs - EuroPLoP 14*, 2014.

Fernandez, Eduardo B., and Raul Monge. "A security reference architecture for cloud systems." *Proceedings of the First International Conference on Dependable and Secure Cloud Computing Architecture - DASCCA 14*, 2014.

Hamid, B., S. Gürgens, and A. Fuchs. "Security patterns modeling and formalization for pattern-based development of secure software systems." *Innovations in Systems and Software Engineering* 12, no. 2 (2015): 109-40.

Hausawi, Yasser M., and William H. Allen. "Usablity and security trade-off." *Proceedings of the 2014 ACM Southeast Regional Conference on - ACM SE 14*, 2014.

Heyman, Thomas, Riccardo Scandariato, Christophe Huygens, and Wouter Joosen. "Using Security Patterns to Combine Security Metrics." *2008 Third International Conference on Availability, Reliability and Security,* 2008.

Hof, Hans-Joachim, and Gudrun Socher. "Poster." *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks - WiSec 16*, 2016.

Kobashi, Takanori, Masatoshi Yoshizawa, Hironori Washizaki, Yoshiaki Fukazawa, Nobukazu Yoshioka, Takano Okubo, and Haruhiko Kaiya. "TESEM: A Tool for Verifying Security Design Pattern Applications by Model Testing." *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, 2015.

Kobashi, Takanori, Nobukazu Yoshioka, Takao Okubo, Haruhiko Kaiya, Hironori Washizaki, and Yoshiaki Fukazawa. "Validating Security Design Patterns Application Using Model Testing." *2013 International Conference on Availability, Reliability and Security*, 2013.

Lee, Wen-Tin, and Po-Jen Law. "A case study in applying security design patterns for IoT software system." *2017 International Conference on Applied System Innovation (ICASI)*, 2017.

Mirakhorli, Mehdi, Ahmed Fakhry, Artem Grechko, Matteusz Wieloch, and Jane Cleland-Huang. "Archie: a tool for detecting, monitoring, and preserving architecturally significant code." *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014*, 2014.

Moral-García, Santiago, Santiago Moral-Rubio, Eduardo B. Fernández, and Eduardo Fernández-Medina. "Enterprise security pattern: A model-driven architecture instance." *Computer Standards & Interfaces* 36, no. 4 (2014): 748-58.

Motii, Anas, Brahim Hamid, Agnès Lanusse, and Jean-Michel Bruel. "Guiding the selection of security patterns based on security requirements and pattern classification." *Proceedings of the 20th European Conference on Pattern Languages of Programs - EuroPLoP 15*, 2015.

Motii, Anas, Brahim Hamid, Agnes Lanusse, and Jean-Michel Bruel. "Guiding the Selection of Security Patterns for Real-Time Systems." *2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS),* 2016.

Near, Joseph P., and Daniel Jackson. "Finding security bugs in web applications using a catalog of access control patterns." Proceedings of the 38th International Conference on Software Engineering - ICSE 16, 2016.

Nguyen, Phu H., Koen Yskout, Thomas Heyman, Jacques Klein, Riccardo Scandariato, and Yves Le Traon. "SoSPa: A system of Security design Patterns for systematically engineering secure systems." *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS),* 2015.

Patu, Vaise, and Shuichiro Yamamoto. "Identifying and Implementing Security Patterns for a Dependable Security Case -- From Security Patterns to D-Case." *2013 IEEE 16th International Conference on Computational Science and Engineering*, 2013.

Ponde, Poonam, Shailaja Shirwaikar, and Christian Kreiner. "An analytical study of security patterns." *Proceedings of the 21st European Conference on Pattern Languages of Programs - EuroPlop 16*, 2016.

Ponde, Poonam, Shailaja Shirwaikar, and Sharad Gore. "Hierarchical Cluster Analysis On Security Design Patterns." *Proceedings of the International Conference on Advances in Information Communication Technology & Computing - AICTC 16*, 2016.

Rauter, Tobias, Andrea Höller, Johannes Iber, and Christian Kreiner. "Patterns for software integrity protection." *Proceedings of the 20th European Conference on Pattern Languages of Programs - EuroPLoP 15,* 2015.

Rauter, Tobias, Andrea Höller, Johannes Iber, and Christian Kreiner. "Static and dynamic integrity properties patterns." *Proceedings of the 21st European Conference on Pattern Languages of Programs - EuroPlop 16*, 2016.

Regainia, Loukmen, Sebastien Salva, and Cedric Ecuhcurs. "A classification methodology for security patterns to help fix software weaknesses." *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA),* 2016.

Rimba, Paul. "Building high assurance secure applications using security patterns for capability-based platforms." *2013 35th International Conference on Software Engineering (ICSE),* 2013.

Sheta, Mohamed Adel, Mohamed Zaki, Kamel Abd El Salam El Hadad, and H. Aboelseoud M. "Anti-spyware Security Design Patterns." *2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC),* 2016.

Sulatycki, Rohini, and Eduardo B. Fernandez. "Two threat patterns that exploit "security misconfiguration" and "sensitive data exposure" vulnerabilities." *Proceedings of the 20th European Conference on Pattern Languages of Programs - EuroPLoP 15,* 2015.

Sultana, Kazi Zakia, Ajay Deo, and Byron J. Williams. "A Preliminary Study Examining Relationships Between Nano-Patterns and Software Security Vulnerabilities." *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC),* 2016.

Uzunov, Anton V., Eduardo B. Fernandez, and Katrina Falkner. "A Comprehensive Pattern-Driven Security Methodology for Distributed Systems." *2014 23rd Australian Software Engineering Conference*, 2014.

Uzunov, Anton V., Eduardo B. Fernandez, and Katrina Falkner. "Security solution frames and security patterns for authorization in distributed, collaborative systems." *Computers & Security* 55 (2015): 193-234.

Uzunov, Anton V., Katrina Falkner, and Eduardo B. Fernandez. "A comprehensive pattern-oriented approach to engineering security methodologies." *Information and Software Technology* 57 (2015): 217-47.

Yoder, Joseph and Jeffrey Barcalow. "Architectural Patterns for Enabling Application Security." Proceedings PLOP'97, Accessed November 28, 2017, https://www.idi.ntnu.no/emner/tdt4237/2007/yoder.pdf

Yoshizawa, Masatoshi, Takanori Kobashi, Hironori Washizaki, Yoshiaki Fukazawa, Takao Okubo, Haruhiko Kaiya, and Nobukazu Yoshioka. "Verifying Implementation of Security Design Patterns Using a Test Template." *2014 Ninth International Conference on Availability, Reliability and Security,* 2014.

Yskout, Koen, Riccardo Scandariato, and Wouter Joosen. "Do Security Patterns Really Help Designers?" *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015.

Yuan, Xiaohong, Emmanuel Borkor Nuakoh, Jodria S. Beal, and Huiming Yu. "Retrieving relevant CAPEC attack patterns for secure software development." *Proceedings of the 9th Annual Cyber and Information Security Research Conference on - CISR 14*, 2014.

Ziani, Adel, Brahim Hamid, Jacob Geisel, and Jean-Michel Bruel. "A model-based repository of security and dependability patterns for trusted RCES." *2013 IEEE 14th International Conference on Information Reuse & Integration (IRI),* 2013.