

ENGINEERING SECURITY METHODOLOGIES FOR DISTRIBUTED SYSTEMS

by

Anton Victor Uzunov

May 2014

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN THE SCHOOL OF COMPUTER SCIENCE
UNIVERSITY OF ADELAIDE

This page is intentionally left blank.

Table of Contents

Abstract	ix
Declaration	x
Acknowledgements	xi
Introduction	1
Background	1
Problem Statement and Research Objectives	2
Solution and Brief Outline of Contributions	3
Thesis Structure	4
Chapter 1: Engineering security into distributed systems: A survey of methodologies	6
Prologue	6
1 Introduction	8
1.1 Scope and Organization	9
2 Taxonomy of Security Methodologies	10
2.1 Background: Key Ingredients of a Security Methodology	10
2.2 Classification Dimensions	11
2.2.1 Methodology Paradigm	11
2.2.2 Specificity	12
2.2.3 Modeling Language and Notation	12
2.2.4 Range of Security Properties	13
2.2.5 Use of Formal Methods and Verification	13
2.2.6 SDLC Stages Supported	13
2.2.7 Ease of Use	13
2.2.8 Tool Support	13
3 Survey of Model-Based Security Methodologies	14
3.1 (General) Model-Based Methodologies	14
3.1.1 Jürjens (MBSE / UMLsec)	14
3.2 Model-Driven Methodologies	15
3.2.1 Model Driven Architecture / Security (MDA/MDS) Methodologies	16
3.2.2 Aspect-Oriented Software Development (AOSD) Methodologies	22
3.2.3 Other Model-Driven Methodologies	26
3.3 Architecture-Driven Methodologies	26
3.3.1 Ren / Taylor	27
3.3.2 Ali / El-Kassas / Mahmoud	28
3.3.3 Other Architecture-Driven Approaches	28
3.4 Pattern-Driven Methodologies	29
3.4.1 Methodologies Using Security Components and/or Architectures as Patterns	30
3.4.2 Methodologies Using Security Patterns	35
3.5 Agent-Driven Methodologies	40
3.5.1 Mouratidis / Giorgini (Secure Tropos)	40
4 Evaluation	42
4.1 Criteria	43
4.2 Analysis Results	44
4.3 Discussion	45
5 Conclusion and Future Directions	46
References	49
Epilogue	60

Chapter 2: Securing distributed systems using patterns: A survey	61
Prologue	61
1 Introduction	63
2 Background, Related Work and Organization of Patterns	64
2.1 Brief History of Software and Security Patterns	65
2.2 Related Work on Security Patterns	65
2.3 Distributed Systems and their Security Concerns	66
2.4 Scope and Organization of Patterns in the Survey	67
3 Survey of Security Patterns for Distributed Systems	67
3.1 Security Contexts	68
3.2 Identity Management	68
3.3 Authentication	68
3.4 Authorization and Access Control	69
3.5 Secure Communications	70
3.6 Filtering	71
3.7 Authentication/Authorization Frameworks and Languages	72
3.8 Distribution Control	72
3.9 Processing	73
3.10 Monitoring	74
4 Evaluation of Surveyed Patterns	74
4.1 Classification	74
4.2 Analysis and Discussion	77
4.2.1 Quality	77
4.2.2 Concerns covered	77
4.2.3 Levels of abstraction	77
4.2.4 Summary	77
5 Methodologies for Applying Security Patterns	78
5.1 Introduction and Classification	78
5.2 Survey	78
5.2.1 Mature and/or larger-scale Methodologies	79
5.2.2 Young and/or smaller-scale Methodologies	80
5.3 Evaluation of Methodologies and Discussion	81
6 Conclusion and Future Directions	83
References	84
Epilogue	88

Chapter 3: A comprehensive pattern-oriented approach to engineering security methodologies	89
Prologue	89
1 Introduction	91
2 Background	92
2.1 Security methodologies	92
2.2 Flexible security methodologies	93
2.3 Towards engineering security methodologies	93
3 Engineering a security methodology: security process aspect	94
3.1 SPPF: a framework of security process patterns and pattern modifiers	94
3.2 Structure and presentation of SPPF	96
3.3 First SPPF level: generic life-cycle modifiers	96
3.3.1 Requirements Analysis (reqAn) Phase pattern	96
3.3.2 Design (des) Phase pattern	97
3.3.3 Implementation (impl) Phase pattern	97
3.3.4 Deployment (deploy) Phase pattern	97

3.4	Second SPPF level: generic security macro-process patterns	97
3.4.1	Overview	97
3.4.2	Security Requirements Determination (SecReq) Phase pattern	98
3.4.3	Countermeasure Introduction (CounterIntro) Phase pattern	100
3.4.4	Security Implementation (SecImpl) Phase pattern	103
3.4.5	Security Administration (SecAdmin) Phase pattern	104
3.4.6	Refine task modifier [refine]	105
3.5	Using SPPF to construct security process models	105
3.6	Engineering strategies for constructing security process models.....	105
3.6.1	“From-scratch” strategy: process pattern assembly	105
3.6.2	Base methodology tailoring	106
4	Engineering a security methodology: conceptual security framework aspect	106
4.1	Conceptual security framework meta-model	106
4.2	Engineering strategies for constructing conceptual framework models	107
4.2.1	Meta-model instantiation.....	107
5	Notation for representing methodology models.....	107
6	S-SMEP: a meta-methodology for engineering flexible security methodologies.....	110
6.1	Constructing S-SMEP	110
6.2	Methodology Initiation phase.....	111
6.2.1	Requirements elicitation and analysis (stage)	111
6.3	Methodology Design/Modeling phase	111
6.3.1	Selecting and organizing SPPF patterns (stage).....	111
6.3.2	Instantiating the conceptual framework meta-model (stage)	112
6.3.3	Selecting existing methodology elements for tailoring (stage).....	112
6.3.4	Verifying methodology model (stage)	112
6.3.5	Generalizing existing methodology elements (stage)	112
6.4	Methodology Construction phase.....	113
6.4.1	Instantiating SPPF patterns (stage)	113
6.4.2	Realizing conceptual framework model elements (stage).....	113
6.4.3	Documenting the methodology (stage)	113
6.4.4	Verification (stage).....	113
6.5	Methodology Deployment phase	114
7	Applying S-SMEP to engineer a security methodology	114
7.1	Methodology Initiation phase.....	114
7.1.1	Requirements Elicitation and Analysis	114
7.2	Methodology Design/Modeling phase	115
7.2.1	Selecting and organizing SPPF patterns (stage).....	115
7.2.2	Instantiating the conceptual framework meta-model (stage)	115
7.2.3	Selecting existing methodology elements for tailoring (stage).....	117
7.2.4	Verifying methodology model (stage)	118
7.3	Methodology Construction phase.....	118
7.3.1	Instantiating SPPF patterns (stage)	118
7.3.2	Realizing conceptual framework model elements (stage).....	118
7.3.3	Discussion	119
7.4	Feature analysis/screening of the approach.....	119
8	Related work	121
8.1	Security process patterns and method engineering	121
8.2	Engineering development methodologies using process patterns	122
8.3	Mining process patterns	122
8.4	Pattern-based process modeling notations	123
8.5	Relation to traditional SME approaches.....	123
8.5.1	Assembly-based SME in the context of security	123

8.5.2 Other SME approaches outside of security.....	123
8.6 Related development meta-methodologies	123
8.7 Security process frameworks	123
8.8 Security methodologies employing process modeling.....	124
9 Conclusion and future directions	124
9.1 Current and future evaluation	124
9.2 Additional research directions	124
9.3 Software support	125
Appendix A: Third SPPF level: abstract process fragments.....	125
References.....	125
Epilogue	130
Chapter 4: Framework for decomposing distributed software architectures.....	131
Prologue	131
I. Introduction.....	133
II. Conceptual framework for architectural decomposition.....	134
A. High-level modelling abstractions (first framework level)	134
B. Functionality decomposition layers (second framework level)	135
C. Technical realisation abstractions (third framework level)	136
III Process for using the framework.....	137
IV. Example	138
A. Incorporation of early security requirements (example).....	140
B. Determination of design-level security requirements (example)	140
V. Related Work	140
VI. Conclusion and Future work	141
References.....	142
Epilogue	143
Chapter 5: Pattern-based threat library and taxonomy for networked and distributed systems	144
Prologue	144
1 Introduction.....	146
2 Background and definitions	147
2.1 Threat patterns and pattern-based threat taxonomies.....	147
2.2 Architectural contexts for the threat patterns.....	149
2.2.1 Functionality decomposition layers	150
2.2.2 Technical realization abstractions.....	150
3 Base threat taxonomy for distributed systems	151
3.1 First level (security) threat patterns	152
3.1.1 Identity attacks	152
3.1.2 Network communications attacks	153
3.1.3 Network protocol attacks	153
3.1.4 Passing illegal data.....	154
3.1.5 Stored data attacks	154
3.1.6 Remote information inference	154
3.1.7 Loss of accountability	155
3.1.8 Uncontrolled operations	155
3.2 Second level (meta-security) threat patterns	156
3.2.1 Cryptography attacks	156
3.2.2 Countermeasure design	157
3.2.3 Configuration / administration	157
3.3 Specializing threat patterns	157

3.4 Instantiating threat patterns	158
4 Constructing a threat taxonomy for peer-to-peer systems	158
4.1 Background on peer-to-peer systems	158
4.2 Setting the architectural context.....	159
4.3 Threat pattern list	159
4.4 Threat taxonomies for other contexts.....	161
4.5 Pattern-based taxonomies as classification schemes.....	162
5 Related work	162
6 Conclusion and future work	163
References	163
Epilogue	166

Chapter 6: Security solution frames for distributed authorization and policy/rule

management.....	167
Prologue	167
1 Introduction	169
2 Security tactics, patterns and solution frames	170
3 Authorization solution frame	173
3.1 Conceptual authorization model pattern family	174
3.1.1 Abstract authorization (abstract security pattern)	175
3.1.2 Pattern modifiers	175
3.1.3 Relationships between modifiers	178
3.2 Enforcement architecture pattern family.....	178
3.2.1 Abstract architecture level.....	178
3.2.2 Solution design level	180
3.2.3 Practical implementation details level.....	181
3.3 Security process pattern family	182
3.3.1 Meta-authorization control (stage micro-process pattern)	183
3.3.2 Variant: Meta-authorization Control with Multi-level Authority Checking (stage micro-process pattern).....	183
3.4 Example implementation workflow for the authorization solution frame	183
4 Security information management solution frame	183
4.1 Policy management pattern family.....	184
4.1.1 Conceptual analysis level.....	184
4.1.2 Abstract architecture level.....	184
4.1.3 Practical implementation details level.....	186
5 Examples	187
5.1 UCON for collaborative applications.....	188
5.1.1 Conceptual model.....	188
5.1.2 Enforcement architecture	188
5.2 ZBAC for SOA-based systems	189
5.2.1 Conceptual model.....	189
5.2.2 Enforcement architecture	189
6 Related work	189
7 Conclusion and Future work	190
References	191
Epilogue	196

Chapter 7: Security solution frames for secure network communication, authentication and cryptographic key management.....

.....	197
Prologue	197
1 Introduction	199

2	Background	200
2.1	Cryptographic primitives and protocols.....	200
2.1.1	Cryptographic primitives	200
2.1.2	Security protocols	201
2.2	Security solution frames.....	201
3	Secure communications solution frame	204
3.1	Secure two-party communication pattern family.....	204
3.1.1	Secure message channel (abstract security pattern).....	204
3.1.2	Pattern modifiers	205
3.1.3	Solution design level.....	206
3.1.4	Practical implementation details level	208
3.2	Security process pattern family	208
3.2.1	Protocol design level.....	208
3.2.2	Network configuration level	209
3.2.3	Implementation workflow for the secure communications solution frame (phase micro-process pattern).....	209
4	Security information management solution frame.....	209
4.1	Key management pattern family	210
4.1.1	Conceptual analysis level.....	210
4.1.2	Abstract architecture level	210
4.1.3	Solution design level.....	212
4.1.4	Practical implementation details level	215
5	Identity management solution frame.....	216
5.1	Authentication pattern family	216
5.1.1	Conceptual analysis level.....	216
5.1.2	Abstract architecture level	217
5.1.3	Solution design.....	217
5.1.4	Practical implementation details	219
6	Analysis examples.....	220
6.1	Brief analysis: IBM Lotus Domino/Notes collaborative system	220
6.1.1	Identity Management (Authentication).....	220
6.1.2	Security Information Management (Key Management)	220
6.1.3	Secure communications	220
6.2	Detailed analysis: Plan9 distributed operating system.....	221
6.2.1	Identity Management (Authentication).....	221
6.2.2	Security Information Management (Key Management)	221
6.2.3	Secure communications	222
7	Related work	222
8	Conclusion and future work	223
	References.....	223
	Epilogue	226

Chapter 8: A framework and process for security methodology quality assessment and improvement..... 227

Prologue	227
1 Introduction.....	229
2 Background	230
2.1 Security methodologies.....	230
2.2 Engineering security methodologies.....	231
3 Methodology quality	233
3.1 General discussion	233
3.2 Quality framework overview	234

4	Quality framework and assessment/improvement process	236
4.1	Methodology quality factors (first framework level).....	236
4.2	Abstract criteria and metrics (second framework level, first sub-level)	236
4.3	Concrete criteria and metrics (second framework level, second sub-level).....	239
4.3.1	Correctness of construction (R).....	239
4.3.2	Completeness (C)	241
4.3.3	Comprehensiveness (V)	241
4.3.4	Usability (U).....	243
4.3.5	Assurance (S) and Adaptability (A)	244
4.3.6	Base criteria profile relationships.....	245
4.4	Quality assessment and improvement process (QAIP)	246
5	Assessing methodology quality.....	248
5.1	Project situation:a very brief description	248
5.2	Step 1: Creating situational criteria profiles for general and peer-to-peer distributed systems	248
5.3	Step 2: Constructing methodology models	250
5.3.1	The methodology of Ali et al.	251
5.3.2	SERENITY.....	252
5.3.3	The methodology of Uzunov et al.....	254
5.4	Steps 3 and 4: Measuring criteria and assessment results	257
5.4.1	Comparison and analysis of assessment results	259
5.4.2	Discussion	259
6	Improving methodology quality.....	260
6.1	Generic conceptual security artefacts.....	260
6.2	Abstract process fragments	260
6.3	The methodology of Uzunov et al.....	261
6.3.1	QAIP Step 5: Determining areas for improvement.....	261
6.3.2	QAIP Step 6: Re-engineering the methodology – requirements.....	261
6.3.3	QAIP Step 7: Re-engineering the methodology – design and construction.....	262
6.3.4	QAIP Step 8: Re-assessment.....	265
6.4	Applying QAIP to SERENITY and the methodology of Ali et al.	265
6.5	Discussion	265
7	Related work	268
8	Conclusion and future work	269
	References	270
	Epilogue	273
	Chapter 9: A comprehensive security methodology for distributed systems	274
	Prologue	274
	1 Introduction	276
	2 Background	277
	2.1 Security methodologies	277
	2.2 Construction and overview of ASE.....	277
	3 ASE: conceptual security framework aspect.....	279
	4 ASE: security process aspect.....	281
	4.1 Requirements analysis (ReqAn).....	281
	4.2 Design (Des).....	281
	4.2.1 Security Requirements Determination (SecReq) phase	281
	4.2.2 Countermeasure Introduction (CounterIntro) phase	287
	4.3 Implementation (Impl)	289
	4.3.1 Security Implementation (SecImpl) phase	289
	5 Applying ASE in practice	290

5.1 Requirements analysis (ReqAn).....	290
5.1.1 Outline of system functionality.....	290
5.1.2 ASE's Security Requirements Determination (SecReq at Req) and Countermeasure Introduction (CounterIntro at Analysis) phases.....	291
5.2 Design (Des)	293
5.2.1 Outline of SHARED's design.....	293
5.2.2 ASE Security Requirements Determination (SecReq at Des) phase	294
5.2.3 ASE Countermeasure Introduction (CounterIntro at Des) phase.....	294
5.3 Additional details	294
5.3.1 Alternative design considerations	294
5.3.2 Security Implementation (SecImpl) phase	294
6 Conclusion and future work	294
6.1 Flexibility and range of solutions.....	294
6.2 Extending and assessing the effectiveness of ASE.....	294
6.3 Practical details: tool support.....	294
6.4 Development methodology integration.....	294
6.5 Re-engineering ASE and transferring features to other methodologies	294
References	294
Epilogue	305
Conclusion and Future Directions.....	306
Achievement of Research Objectives and Contributions	306
Future Research Directions	307
Bibliography (for Introduction and Conclusion and Future Directions)	309

Abstract

Over the last decade, researchers and practitioners have increasingly come to acknowledge that the introduction of security into software systems – especially complex, distributed systems – should proceed by means of a structured, systematic approach, combining principles from both software and security engineering. Such systematic approaches, particularly those implying some sort of process aligned with the development life-cycle, are termed *security methodologies*. While there are numerous methodologies in the literature, each with its own peculiar advantages and disadvantages, making it more or less suitable for a given set of project situations, none can lay claim to being universal, i.e. able to take into account all system-specific attributes, all technologies, all skill levels, and – in general – to be applicable to all project situations. In other words, the literature does not currently present developers with an “ideal” methodology (in an absolute sense); and, indeed, such a requirement would be infeasible, since “ideal” must necessarily be interpreted with respect to a given situation – encompassing system types, technologies, skillsets and whatever other qualities are seen as desirable. The problem facing the area is thus not so much the construction of “bigger and better” methodologies with novel or interesting features – i.e. (unattainably) ideal methodologies in an absolute sense – but the construction of (attainably) ideal methodologies for particular project situations.

This thesis proposes a comprehensive solution to the latter problem by developing a conceptual “toolkit” for engineering security methodologies, with an emphasis on security methodologies for distributed systems. The “toolkit” consists of a number of inter-related parts: a framework of process patterns, a domain-specific meta-model and a unifying meta-methodology for constructing and tailoring security methodologies (for any system type); a set of generic conceptual security artefacts – usable across different methodologies – for addressing various networked and distributed systems security features and for supporting threat modeling in a networked/distributed systems context; and a framework for assessing and improving security methodology quality, which, when combined with the meta-methodology, helps to ensure that all construction/tailoring efforts are sensible – i.e. of measurably good quality. Besides being part of an overall approach, each of these inter-related parts makes its own set of unique, self-contained contributions to the area of secure software engineering. Some of the parts are complete in themselves, while others require elaboration or specialization for different situations. In all cases the frameworks, artefacts and other “tools in the toolkit” can be customized and extended in various ways, providing developers, architects, methodology engineers and other team members with a high degree of freedom and flexibility. The latter point in particular, as well as the whole approach, is demonstrated incrementally throughout the thesis via the engineering of a specific pattern-based security methodology for distributed systems – a case study which in itself is another, final contribution. Of course, the case study methodology is also bigger and better (with respect to its predecessor) and contains novel features, but is only ideal with respect to its project situation. Through the presentation of the parts of the “toolkit” and the illustration of its use, the thesis accomplishes its task of both proposing and demonstrating the value of a comprehensive, holistic approach to engineering security methodologies, thereby offering a solution to the initial problem.

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give consent to this copy of my thesis when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

The author acknowledges that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Anton Victor Uzunov
May 2014

Acknowledgements

First of all I would like to thank my two supervisors: Dr. Katrina Falkner and Prof. Eduardo B. Fernandez. I felt very fortunate throughout my degree to have two pairs of experienced eyes glance at my work and provide feedback from two different, complementary angles. I am particularly grateful to Katrina for promptly reading and always providing valuable feedback on all my manuscripts, even under excruciatingly tight deadlines; for encouraging me and supporting my work; for all the friendly discussions; for inspiring me towards excellence; and for allowing me to collaborate freely and trusting me to develop my ideas in the way I saw fit. I am particularly grateful to Eduardo for all his advice and his deeply insightful comments and suggestions; for always being “on-line”, even for late-evening Skype meetings spanning different continents, and being so generous with his time; for all the discussions on topics in software security and beyond; and in the end for helping me to realize what I myself had done (!). When I first spoke to Eduardo about external co-supervision a number of years ago, I was struck by his friendly unpretentiousness; in good time, I realized I had the privilege of meeting not only an immensely experienced and knowledgeable supervisor, but also a man with broad interests and a genuine friend. Collaborating with both my supervisors was always marked with mutual respect and understanding; in fact, I often felt as though I was working with peers or colleagues – something that I appreciated very much.

I would like to thank the University of Adelaide's School of Computer Science administrative staff – Sharyn Liersch and Julie Mayo – for taking care of my conference travel arrangements; as well as the postgraduate coordinators – Dr. Michael Sheng and Dr. Frank Neumann – for their readiness to help with filling out all the Graduate Centre forms. I would particularly like to thank Michael for helping me to settle in the School of CS at the beginning and also for his friendly encouragement.

Thanks goes to the local IT-admin guru (and fellow PhD student) Yuval Yarom, not only for setting up my University machine (which was used rarely, but to good purpose), user accounts etc., but also for the comradeship. Shalom Yuval!

I would also like to thank my colleagues, supervisors and line managers back at work – especially Richard Appleby, Bradley Hopkins, Chris North and Dr. Olivier De Vel – for supporting me in my decision to undertake PhD studies. My thesis topic is all my own, but it really is in security; after gaining many skills and much experience and expertise, I believe that (besides coming out alive – not a bad feat) I managed to become a bigger and better security expert and, above all, a bigger and better security researcher.

Finally, I would like to thank my family – close and distant – for all their love and support, which, although mentioned here last, was indispensable. To my grandad I would say: no, I didn't worry too much at the end, and, yes, I mostly “hurried slowly” – *Eile mit Weile* (as the Germans say).

Anton V. Uzunov
October 2013