Journal of Engineering Science and Technology
Vol. 12, No. 9 (2017) 2513 - 2527
© School of Engineering, Taylor's University

KNOWLEDGE REPRESENTATION OF SECURITY DESIGN PATTERN LANDSCAPE USING FORMAL CONCEPT ANALYSIS

POONAM S. PONDE*, SHAILAJA C. SHIRWAIKAR, VILAS S. KHARAT

Department of Computer Science, Savitribai Phule Pune University, Pune 411007, India
*Corresponding Author: poonamponde@gmail.com

Abstract

Security design patterns are proven solutions to recurring security problems. They are classified into various categories, each containing a set of attributes. However, the large number of patterns and classification schemes makes it difficult to choose a pattern for a given security problem. To apply patterns effectively, there must be a systematic method of organizing the patterns, so that it is possible to look up a design pattern unambiguously according to its purpose. While a lot of research focuses on developing new patterns and classifications, these issues have not been adequately addressed. In this paper, we present a novel approach of applying Formal Concept Analysis (FCA) on a chosen set of patterns classified according to a common set of attributes. The resulting concept lattice can be used for mining knowledge from the concepts, identifying pattern groups, and their relationships with the goal of applying appropriate patterns to security requirements. We propose the use of FCA over conventional data analysis methods for the simplicity of data preparation, the discovery of hidden knowledge, and cluster interpretation, with a visual representation of the pattern domain.

Keywords: Design patterns, Security, Concept lattice.

1. Introduction

Integrating security into all aspects of Information Systems has become critical. However, it requires skill and expertise to design secure systems. Expert knowledge in the form of design patterns can provide valuable guidance to the designers. The first design patterns for security were developed by Yoder and Barcalow in 1997 [1].

Subsequently, many patterns and pattern catalogues emerged. Due to a large

Nomenclatures

c	A function which calculates the confidence of an association rule
s	A function which calculates the support of an association rule

Special Symbols

$\forall$	For each
$\in$	Set membership
$\subseteq$	Subset
$\cup$	Union of two sets

Greek Symbols

η	Support count indicating number of objects that have a specific set of attributes
σ	A function which gives a common set of attributes for a set of objects
τ	A function which gives a common set of objects for a set of attributes

Abbreviations

FCA	Formal Concept Analysis

number of patterns, searching for a pattern becomes a daunting task, leading to the poor applicability of patterns in practice. This requires patterns be well organized. Researchers have proposed several classification schemes, each with multiple attributes. However, as more security threats and vulnerabilities were discovered, more patterns and newer classification methodologies were identified. Existing patterns need to be reclassified and retrofitted into the new schemes, which is a challenging task. Pattern applicability will improve if all patterns are classified uniformly using the same attributes. In the present work, we have identified a set of attributes which can be applied to all patterns.

Formal Concept Analysis (FCA) is an analytical method on tabular data which captures the relationships in a concept lattice [2]. We propose a novel approach of applying FCA to the classified pattern dataset. This will enable knowledge discovery and effective navigation through the security design pattern landscape. Patterns which can handle one or even multiple security issues at a time can be easily identified. The concept lattice also gives a nice visualization of the pattern clusters.

The paper is organized as follows. Section 2 gives a brief background on security design patterns and their classification. It also discusses the theoretical concepts behind FCA. The steps carried out in applying FCA are discussed in Section 3. The results are discussed in Section 4, followed by a conclusion in Section 5.

2. Background

In this section, we present a brief background on security design patterns and FCA.

2.1. Security Design Patterns

Alexander et al. first introduced the concept of design patterns for use of living spaces, in his book ‘A Pattern Language’, published in 1977 [3]. The concept of a pattern was adopted by the software community in the book ‘Design Patterns:

Elements of Reusable Object-Oriented Software', published in 1994 whose authors Gamma, Helm, Johnson, and Vlissides are known as the 'Gang of Four [4]. The first design patterns related to security were published by Yoder and Barcalow in 1997. Their paper described 7 design patterns. Subsequently, many design patterns and pattern catalogs emerged. Important pattern contributors are Romanosky [5], Kienzle et al. [6], Blakley and members of the Open Group security Forum [7], Hafiz et al. [8, 9] and the Microsoft Patterns and Practices group [10, 11]. In their book published in 2005, Schumacher and a working group of security pattern experts discussed 46 patterns [12]. Steel et al. [13] compiled a catalog of 23 core security patterns for J2EE and web services. In their guide 'Security Design Patterns', Dougherty et al., at the Carnegie Mellon Software Engineering Institute, described 15 design patterns [14]. Fernandez presented 68 design patterns in the book 'Security Patterns in Practice: Designing Secure Architectures Using Software Patterns' [15]. Today the total number of security design patterns is around 400 [16, 17]. To organize the patterns, several classification schemes have been developed [17, 18]. Each classification scheme contains a set of criteria or attributes. Some important classification schemes and their criteria are listed in Table 1.

Table 1. Security design pattern classification schemes and criteria.

Category	Attributes/Criteria
Interrogatives [19]	Purpose (Why), Data (What), Function (How), Timing (When), Network (Where), People (Who) and Scorecard (Test)
Purpose [16]	Structural, Behavioural, Generic, Procedural, Creational
System type [7]	Available Systems, Protected Systems
Lifecycle stages [20]	Architectural, Requirement, Analysis, Design, Implementation, Integration, Deployment, Operation, Maintenance, Disposal
Security principles [21]	Confidentiality, Integrity, Authentication, Authorization, Availability, Confidentiality, Access Control, Non-repudiation, Accountability
Location [21]	Core, Perimeter, Exterior
Logical tiers [16]	Application, Host, Client, Logic, Data, Integration, Platform and OS, Distribution, Transport, Network
Threat Model [22]	Spoofing, Tampering, Information Disclosure, Repudiation, Escalation of Privilege, Denial of Service
Constraint [22]	Regulatory, Organizational, Human, Mechanism
Response [22]	Avoidance, Deterrence, Prevention, Detection, Mitigation, Recovery, Forensics
Code Source [22]	New, Open-source, Runtime, Model transform, Wizard code, Reuse library, Outsourced, Legacy, Off-the-shelf, Remote web service
Enterprise level Security [15, 23]	Enterprise Security and Risk Management, Identification & Authentication, Access Control, System Access Control, Operating System Access Control, Accounting Patterns, Firewall Architecture, Secure Internet Applications, Cryptographic Key Management.
Domain [16]	Web Services, J2EE, Microsoft

2.2. Theoretical basis of formal concept analysis

Formal Concept Analysis is also known as Galois lattice mining and is a branch of lattice theory that allows identification of meaningful groups of objects that have common attributes [24, 25]. According to Bělohlávek [26], the advantage of using FCA is that hidden concepts and dependencies in data can be discovered and reasoned with. The concepts and dependencies can be also be visualized. FCA has been applied to data for knowledge discovery in various fields, such as biology, semantic web, machine learning, medical information systems, decision systems and others [27, 28].

2.2.1. Context

A context is a triple: C = (E,P, I) where E is a finite set of *elements (objects)*, P is a finite set of *properties (attributes)* and I is a binary relation between E and P (such that I $\subseteq$ E X P). I is called the formal context and represented as a table. An example is shown in Fig. 1.

		P			
	I	*a1*	*a2*	*a3*	*a4*
E	*o1*	X		X	
	o2		X		X
	o3	X	X	X	
	o4	X			

Fig. 1. Formal context.

The rows in the formal context are the objects and columns are the attributes. Each table entry is Boolean, which indicates whether the object has that attribute. FCA is a data analysis method where data is the formal context. Each formal context is transformed into a mathematical structure called a concept lattice. According to Wille [24], concepts transform information into knowledge.

2.2.2. Concept

A concept is a pair of sets (X,Y) where X is a set of elements called the extent and Y is a set of properties called the intent; given : X $\subseteq$ E and Y $\subseteq$ P, Y = σ(X) and X = τ(Y) where σ(X) = {p $\in$ P | $\forall$e $\in$ X : (p, e) $\in$ I}, τ(Y) = {e $\in$ E | $\forall$p $\in$ Y : (p, e) $\in$ I} [29].

Therefore a concept is a maximal collection of elements sharing common properties, i.e., any e $\in$ E has all properties in P, and all properties p $\in$ P fit all elements in E. From the formal context shown in Fig. 1, ({o1,o3,o4},{a1}) is a concept. However, ({o1,o3},{a1}) is not a concept since, although σ({o1,o3})= {a1}, τ(a1) = {o1,o3,o4}. Similarly ({o1,o3},{a1,a3}) is a concept since σ({o1,o3})={a1,a3} and τ({a1,a3})={o1,o3}.

FCA produces three kinds of output from the input data:

i. Concept lattice

The formal context is visualized using a concept lattice or Hasse diagram [29]. In a Hasse diagram, the nodes represent concepts. The lattice is a collection of formal concepts, which are hierarchically ordered by a subconcept-superconcept relationship. A concept (A1, B1) is a subconcept of (A2, B2) if (A1, B1) ≤ (A2, B2) iff A1⊆A2 (iff B2 ⊆ B1). This indicates that (A1, B1) is more specific than (A2,B2). The concept lattice is represented graphically to support analysis, mining, visualization and interpretation. The concept lattice for the formal context in Fig. 1 is shown in Fig. 2.

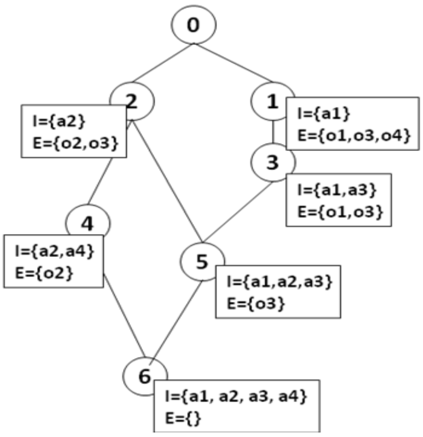

Fig. 2. Concept lattice.

ii. Attribute implications

An attribute implication describes a particular dependency which is valid in the data. It indicates the relationships between attributes. It is an expression of the form: A =>B where A and B represent a set of attributes such that if an object has all attributes from A, it has all attributes from B as well. For example, for the formal context shown in Fig. 1, a3 => a1 is an implication, since an object which has attribute a3, also has attribute a1. The attribute implications of the above formal context are:

- a1 a2 => a3;
- a3 => a1;
- a4 => a2;

iii. Association rules

An association rule is an implication expression of the form X→Y where X and Y are disjoint itemsets. The strength of an association rule is measured in terms of its Support and Confidence. Support determines how often the rule applies to the dataset. Support, s(X=>Y) = η(X∪Y) / N where N is the number of objects in the set. Confidence determines how frequently items in Y appear where X appears in a rule. Confidence, c(X=>Y) = η(X∪Y) / η(X). Association rules that have a higher support are interesting because they reveal important characteristics of the data set. Confidence is a measure of the reliability of the inference. For example, the rule a3=>a1 has a support of 0.5 and confidence of 100%.

When the formal context is very large, the concept lattice is also very large and it is unreasonable to display the complete data in a single lattice diagram. In such a case, the following approaches may be used:

a) Reduce the number of objects or attributes by observation.

b) Split the data, carry out a separate analysis, and merge the results.

c) Calculate the iceberg lattice of the context.

2.2.3. Iceberg concept lattice

The set of all frequent concepts of a context is called the iceberg lattice of the context. It consists of the top-most concepts of the lattice, which provide the most global structuring of the domain [2, 30]. A concept is called a frequent concept if its intent is frequent. These concepts are also called an order filter or upset. Let minsupp be a threshold $\in$ [0,1], then B is said to be frequent itemset if σ(B) >= minsupp [30]. Figure 3 shows the iceberg lattice with the concepts having a minimum support (minsupp) = 0.5.

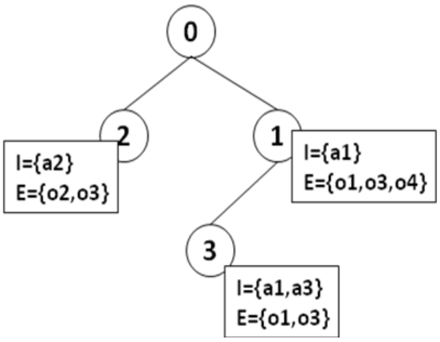

Fig. 3. Iceberg lattice.

3. Using FCA For Pattern Data Mining

The steps that are followed for applying FCA on security pattern data are shown in Fig. 4.

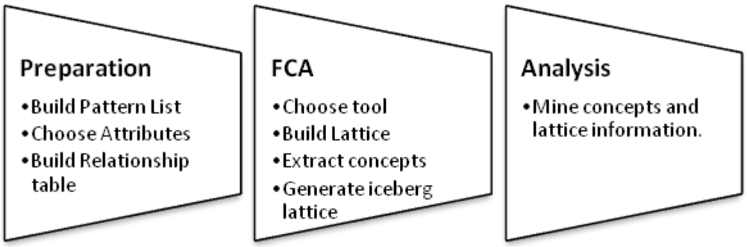

Fig. 4. Steps in applying FCA.

It involves three stages:

a) Data preparation: The activities in this stage are building the pattern list, selecting the attributes for classification and constructing the formal context. The formal context is a matrix of pattern-attribute binary relations.

b) FCA: This stage includes the tasks of choosing the FCA tool, constructing the concept lattice, extracting the concepts, and generating the iceberg lattice based on minimum support.

c) Post-processing: The data that is generated by FCA is used for querying and mining useful information.

3.1. Data preparation

The pattern classification methodologies and their criteria are listed in Table 1. Multiple classification schemes make it difficult to apply them to all patterns. A core set of criteria can aid efficient organization of the pattern landscape. We propose that applying a classification scheme that is based on fundamental core security principles [31] along with the threats [32] that can be mitigated by the pattern, will help designers to apply patterns more effectively. Moreover, these criteria can be applied to most of the security patterns. Table 2 lists the 13 security criteria which are applied to the security patterns.

Table 2. Pattern classification criteria.

No.	Attribute	Description
		Security Principles
1.	Authentication / Identity	Identifies the principals - users, machines, and processes.
2.	Access Control / Authorization	Determines which principal may access which data, i.e., it defines the access privileges.
3.	Integrity	Ensures that data and communications will not be compromised by active attacks.
4.	Confidentiality	Ensure data and communication privacy from unauthorized access.
5.	Availability	Assures that authorized users can use the resources when they are required.
6.	Non-Repudiation	Users' refusal to acknowledge participation in a transaction.
7.	Accountability	Logging certain actions for audits.
8.	Key management	Related to the generation, sharing and storage of keys.
		Threats
9.	Spoofing	Attempt to gain access to a system using a forged identity.
10.	Tampering	Corruption of data during communication through the network.
11.	Information Disclosure	Unwanted exposure and loss of confidentiality of private data.
12.	Denial of Service	Attack on system availability.
13.	Elevation of privilege	Attempt to raise the privilege level by exploiting some vulnerability.

The 13 attributes, a set of 192 security design patterns, and the binary relation between them forms the formal context for FCA. The attributes are represented as columns and patterns as rows. Each table element is assigned a binary value - 1 or 0 depending on whether the pattern satisfies that criterion or not. A representative sample is shown in Table 3.

Table 3. Pattern-attribute relationship table.

No.	Pattern Name	Authentication	Access Control	Integrity	Confidentiality	Non Repudiation	Availability	Accountability	Key Management	Spoofing	Tampering	Information Disclosure	Denial of Service	Elevation of Privilege
1	**A Pattern for WS-Trust**	1	0	0	1	1	0	1	0	1	0	1	0	0
2	**Access Control List**	1	1	0	1	0	0	0	0	1	0	0	0	0
	....													
192	**XML Encryption**	0	0	0	1	0	0	0	0	0	0	1	0	0

3.2. Applying FCA

The pattern-attribute binary relationship table is the input to the FCA tool. There are a variety of software tools available for FCA [33]. The formal context is represented in a format as specified by the FCA tool (.csv is commonly supported). The tools used to carry out the present research work are ConExp and Galicia. The following outputs are obtained as results:

i. Concepts and concept lattice.
ii. Implications and Association Rules.
iii. Iceberg lattice.

4. Results and Discussion

The concept lattice that is generated by FCA for the pattern dataset is shown in Fig. 5. The height of the lattice is 8 and there are 148 concepts in the lattice. Each node in the lattice represents a concept {I,E}, where I represents the intent i.e. the attributes and E represents the extent, i.e., the patterns in the concept. The patterns in a single group share common attributes. Pattern clusters at a higher level in the lattice support fewer attributes than those at lower levels. These patterns are more generic and solve fewer security issues. For example, the pattern group {Trust Partitioning, chroot jail} has only one attribute: {Tampering}. Concepts at lower lattice levels have more attributes and hence, the patterns in these concepts are more useful. These patterns address multiple issues. For example, the lower level group {Multilevel Security, Multilevel Secure Partitions} has the attributes {Confidentiality, Information Disclosure, Access Control, Tampering, Integrity, Elevation of Privilege}. The lattice gives a schematic of the entire security pattern domain and clearly brings out the relationships between the patterns and attributes.

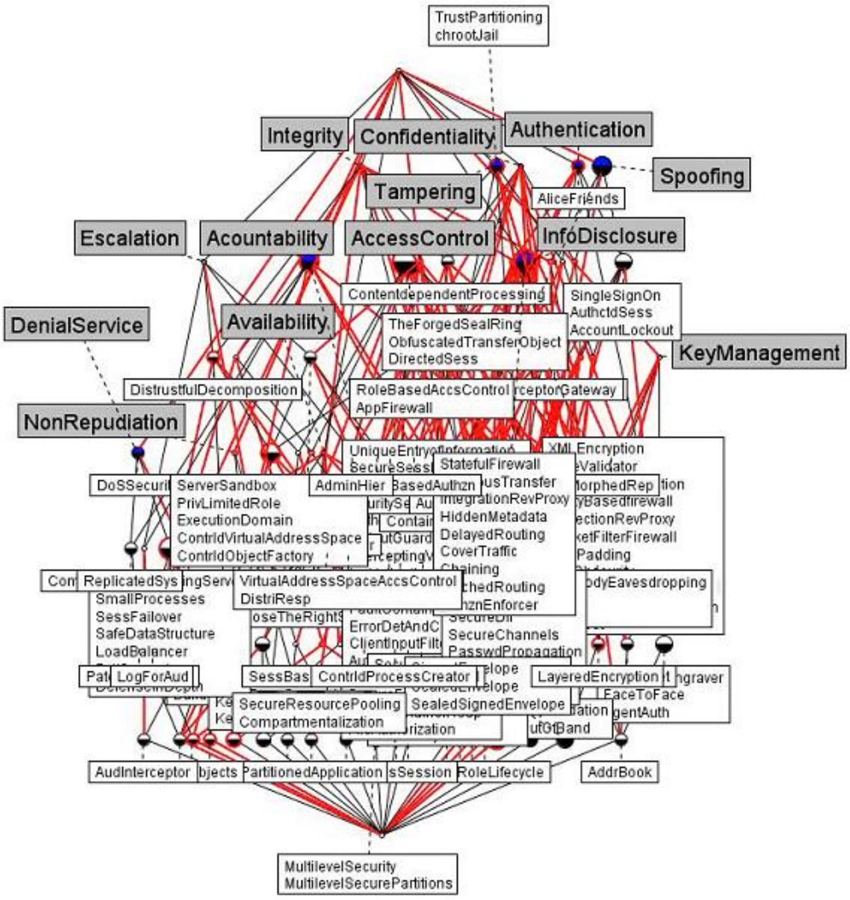

Fig. 5. Concept lattice of patterns and attributes.

4.1. Concepts

Each node of the lattice represents a concept which is connected to other concepts by edges. The node in Fig. 6 corresponds to the concept having the intent {Integrity, Tampering} and an extent of 53 patterns. This concept has 6 sub-concepts, having intents as: {Integrity, Tampering, Access Control}, {Integrity, Tampering, Availability}, {Integrity, Tampering, Escalation of Privilege}, {Integrity, Tampering, Accountability}, {Integrity, Tampering, Authentication} and {Integrity, Tampering, Information Disclosure}. It has two superconcepts which have the attributes of {Integrity} and {Tampering} respectively.

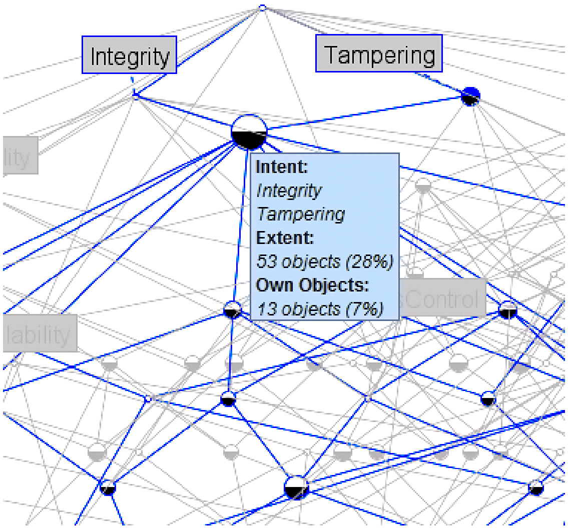

Fig. 6. Concept.

The concepts which have maximum number of attributes in their intent are shown in Table 4. Out of the 148 concepts in the lattice, concepts 1 and 2 have 7 attributes each, while the others have 6 attributes each. The patterns in these concepts are more useful patterns because they are capable of handling more number of security issues. The superconcept-subconcept relationship is clearly visible in the concepts. For example, 1 and 2 are subconcepts of 3.

Table 4. Concepts with maximum attributes.

	Attributes	Patterns
1.	Information Disclosure, Confidentiality, Integrity, Access Control, Tampering, Authentication, Spoofing (Total 7)	Capability, Check Point, Secure Visitor, Secure Process Thread, Limited Access With Errors
2.	Access Control, Tampering, Integrity, Authentication, Confidentiality, Spoofing, Accountability (Total 7)	Full Access With Errors
3.	Confidentiality, Integrity, Access Control, Tampering, Authentication, Spoofing (Total 6)	Capability, Check Point, Secure Visitor, Secure Process Thread, Limited Access With Errors, Full Access With Errors
4.	Confidentiality, Information Disclosure, Access Control, Tampering, Integrity, Escalation of Privilege (Total 6)	Multilevel Secure Partitions, Multilevel Security
5.	Information Disclosure, Confidentiality, Integrity, Authentication, Access Control, Escalation of Privilege (Total 6)	Access Session
6.	Tampering, Integrity, Authentication, Confidentiality, Spoofing, Accountability (Total 6)	Full Access With Errors, Password Authentication
7.	Spoofing, Authentication, Information Disclosure, Confidentiality, Accountability, Non-Repudiation (Total 6)	A pattern for WS Trust
8.	Tampering, Information Disclosure, Confidentiality, Integrity, Accountability, Non Repudiation (Total 6)	Secure Logger

4.2. Iceberg lattice

All the concepts in the lattice cannot be easily viewed at a time. It is difficult to draw any inferences just by observation. Hence, additional tools and algorithms are used to extract lattice information. An iceberg lattice is the set of all frequent concepts of the context, whose support meets a given threshold. It encapsulates the common knowledge, condensed representation or most generic values of the context. The iceberg lattice for minsupp=0.4 is shown in Fig. 7.

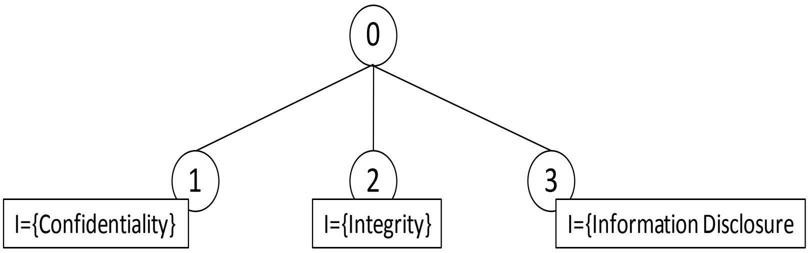

Fig. 7. Iceberg lattice for minsupp=0.4.

As we can see, this lattice has only 3 concepts. This indicates that these three concepts form the topmost concepts of the lattice. These concepts are:

- Confidentiality which has an extent of 81 (42%)
- Integrity which has an extent of 91 (47%)
- Information Disclosure which has an extent of 80 (42%)

The inference from the iceberg lattice is that out of the 192 patterns in the set, more than 40% of the patterns satisfy the criteria of Confidentiality, Integrity, and Information Disclosure. There is no other attribute having a support of more than 40%. Further reduction of minsupp to 0.2 yields the iceberg lattice shown in Fig. 8. It has 11 concepts and the additional concepts are: Access Control (extent: 53, support 28%), Tampering (extent: 60, support 31%) Authentication, Spoofing, Authentication+ Spoofing (extent: 39, support 20%), Confidentiality+ Information Disclosure (extent: 55 objects, support 29%), Confidentiality+Integrity (extent: 42 objects, support 22%), Integrity+Tampering (extent: 53 objects, support 28%).

4.3. Implications

The concepts and their hierarchical structure reveal the relationships between patterns and attributes. The implications of the context give valuable insights into attribute relationships. This can be mined to extract useful information regarding attributes. An attribute implication describes an attribute dependency which holds in the data; for example, concepts which have the attributes of Access control, Tampering, and Information Disclosure also have the attribute of Integrity. A part of the lattice is shown in Fig. 9.

From the diagram, we can detect the implication Access Control, Spoofing => Authentication. It is indicated by the fact that there is no concept having ‘Access Control’ and ‘Spoofing’ in its intent, but not ‘Authentication’. 23 patterns in the data set have this implication and hence, a support of 12%.

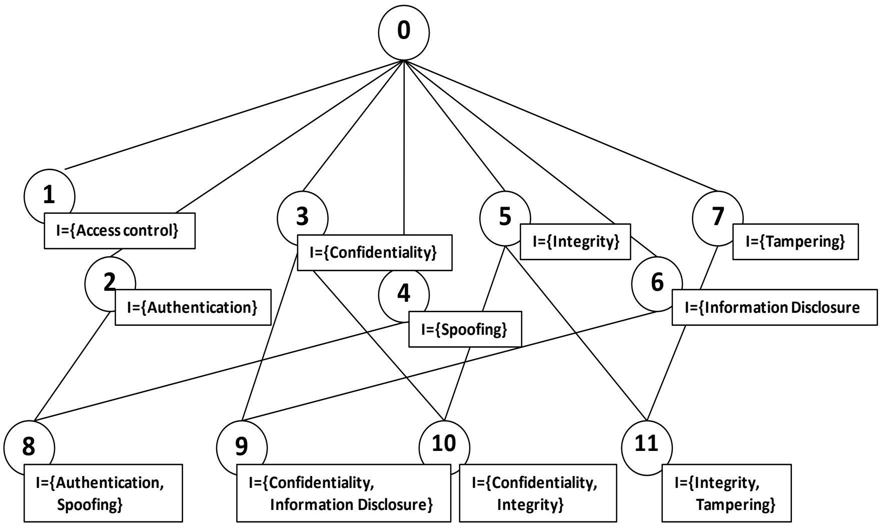

Fig. 8. Iceberg lattice for minsupp=0.2.

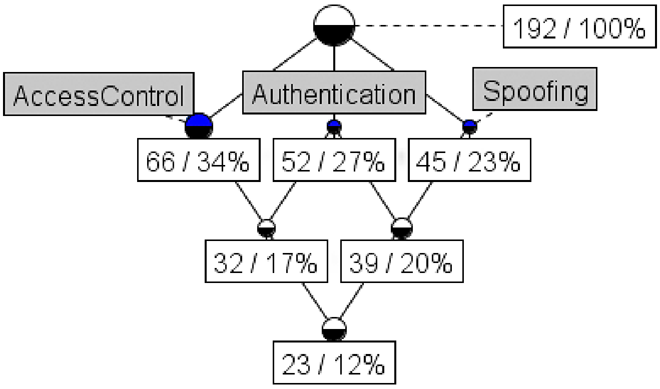

Fig. 9. Deriving implications from the lattice.

5. Conclusions

Formal Concept Analysis is a useful tool to visualize the concepts within data and the relationships between concepts. It is a data analysis method which enables the discovery of hidden knowledge existing in the data. In this paper, we have applied FCA to a set of 192 security design patterns which have been classified according to 13 criteria. The criteria are the fundamental security principles and security threats. The resulting concept lattice gives valuable information about pattern clusters and the attributes of the cluster. Practitioners can easily identify patterns which solve specific security issues. The grouping of patterns indicates the relationships between patterns and can be used to identify redundant patterns. Since the number of concepts is large, it is not possible to draw inferences from the lattice just by observation. Additional inferences are drawn from the iceberg lattice, which gives the topmost concepts in the lattice hierarchy. These concepts form the most generic concepts of the lattice. Implications and association rules reveal relationships and dependencies between the attributes. Support and confidence measures indicate the strength and reliability of the association rules. Future work includes association rule mining and a detailed exploration of the concepts. The concepts and cluster knowledge, if made readily available in the form of a web-based tool, will enable pattern search based on security goals and thereby improve pattern applicability.

References

1. Yoder, J.A.; and Barcalow, J.C. (1997). Architectural patterns for enabling application security. *Proceeding of the Conference on Pattern Languages of Programs*, Monticello/IL, 1-31.
2. Lakhal, L.; and Stumme, G. (2005). Efficient mining of association rules based on formal concept analysis. *Formal Concept Analysis*, Vol. 3626 of LNCS, Springer, 180-195.
3. Alexander, C.; Ishikawa, S.; and Silverstein, M. (1977). *A pattern language: towns, buildings, construction*. Oxford University Press, New York.
4. Gamma, E.; Helm, R.; Johnson, R.; and Vlissides, J. (1994). *Design patterns: Elements of object-oriented software*. Addison Wesley.
5. Romanosky, S. (2003). Enterprise security patterns. *Information Systems Security Association Journal*.
6. Kienzle, D.M.; Elder, M.C.; Tyree, D.; and Edwards-Hewitt, J. (2001). Security patterns repository version 1.0, *DARPA, Washington DC*. Retrieved December 15, 2015, from http://www.scrypt.net/~celer/securitypatterns/repository.pdf
7. Blakley, B.; and Heath, C. (2004). Security design patterns technical guide - version 1. *Open Group, Berkshire, UK*. Retrieved November, 24, 2015, from www.opengroup.org/onlinepubs/9299969899/toc.pdf
8. Hafiz, M.; Johnson, R.; and Afandi, R. (2004). The security architecture of qmail. *Proceedings of the 11th Conference on Patterns Language of Programming* (*PLoP'*04), Monticello, IL, U.S.A.
9. Hafiz, M. (2006). A collection of privacy design patterns. *Proceedings of the 13th Conference on Patterns Language of Programming* (*PLoP*'06). ACM, New York, NY, USA, Article 7.
10. Trowbridge, D.; Cunningham, W.; Evans, M.; Brader, L.; and Slater, P. (2004). *Describing the enterprise architectural space*. Microsoft Press.
11. Microsoft. (2005). *Web service security scenarios, patterns, and implementation guidance for web services enhancements (WSE) 3.0.* Microsoft Press.
12. Schumacher, M.; Fernandez, E.B.; Hybertson, D.; and Buschmann, F. (2005). *Security patterns: Integrating security and* systems *engineering*. John Wiley & Sons.
13. Steel, C.; Nagappan, R.; and Lai, R. (2005). *Core security patterns: Best practices and strategies for J2EE(TM), web services, and identity management*. Prentice Hall International.

14. Dougherty, C.; Sayre, K.; Seacord, R.C.; Svoboda, D.; and Togashi, K. (2009). Secure design patterns. *Technical Report CMU/SEI 2009-TR-010,* Software Engineering Institute, Carnegie Mellon University.
15. Fernandez, E.B. (2013). *Security patterns in practice: Designing secure architectures using software patterns*. Wiley.
16. Bunke, M.; Koschke, R.; and Sohr, K. (2012). Organizing security patterns related to security and pattern recognition requirements. *International Journal on Advances in Security*, 5, 46-67.
17. Ponde, P.; and Shirwaikar, S. (2016). An exploratory study of the security design pattern landscape and their classification. *International Journal of Secure Software Engineering (IJSSE)*, 7(3), 26-43.
18. Yoshioka, N.; Washizaki, H.; and Maruyma, K. (2008). A survey on security patterns. *Progress in Informatics*, 5, 35-47.
19. Fernandez, E.B.; Washizaki, H.; Yoshioka, N.; Kubo, A.; and Fukazawa, Y. (2008). Classifying security patterns. *Proceedings of the 10th Asia-Pacific Web Conference*. Springer Berlin Heidelberg, 342-347.
20. Fernandez, E.B.; Yoshioka, N.; and Washizaki, H. (2007). Using security patterns to build secure systems. *International Workshop on Software Patterns and Quality*. Information Processing Society of Japan, 47-48.
21. Hafiz, M.; Adamczyk, P.; and Johnson, R.E. (2012). Growing a Pattern Language (for security). *ACM International Symposium on New ideas, New Paradigms, and Reflections on Programming and Software*, ACM New York.
22. VanHilst, M.; Fernandez, E.B.; and Braz, F. (2009). A multidimensional classification for users of security patterns. *Journal of Research and Practice in Information Technology*, 41(2), 87.
23. Washizaki, H.; Fernandez, E.B.; Maruyama, K.; Kubo, A.; and Yoshioka, N. (2009). Improving the classification of security patterns. *Database and Expert Systems Applications*, 165-170.
24. Wille, R. (2009). Restructuring lattice theory: An approach based on hierarchies of concepts. *International Conference on Formal Concept Analysis*, Springer Berlin Heidelberg. 314-339.
25. Ganter, B.; and Wille, R. (2012). *Formal concept analysis - Mathematical foundations*. Springer Science & Business Media.
26. Belhlavek, R. (2008). Introduction to formal concept analysis. Palacky University, Department of Computer Science, Olomouc. Retrieved February, 20, 2016, from https://phoenix.inf.upol.cz/esf/ucebni/formal.pdf.
27. Škopljanac-Mačina, F.; and Blašković, B. (2014). Formal concept analysis-overview and applications. *Procedia Engineering*, 69, 1258-1267.
28. Poelmans, J.; Ignatov, D. I.; Kuznetsov, S. O.; and Dedene, G. (2013). Formal concept analysis in knowledge processing: A survey on applications. *Expert Systems with Applications*, 40(16), 6538-6560.
29. Birkhoff, G. (1940). *Lattice theory*. Vol. 25. American Mathematical Society.
30. Jay, N.; Kohler, F.; and Napoli, A. (2008). Analysis of social communities with iceberg and stability-based concept lattices. *Proceedings of International Conference on Formal Concept Analysis*. Springer Berlin Heidelberg. 258-272.

31. Commission of European Communities. (1991). Information technology security evaluation criteria. Retrieved October 3, 2015, from http://www.iwar.org.uk/comsec/resources/standards/itsec.htm.
32. Howard, M.; and LeBlanc, D. (2003). *Writing secure code.* Microsoft Press.
33. Priss, U. (2007). FCA software. Retrieved March 15, 2016, from http://www.upriss.org.uk/fca/fcasoftware.html.