Building a Multidimensional Pattern Language for Insider Threats

DAVID MUNDIE, CERT® Program, Software Engineering Institute
ANDREW P. MOORE, CERT Program, Software Engineering Institute
DAVID MCINTIRE, CERT Program, Software Engineering Institute

As part of ongoing insider threat research at the CERT Program, we have developed 26 patterns for insider threat. This paper describes our attempts at organizing those patterns into a pattern language. After discarding several hierarchical, unidimensional models, we adopted a multidimensional organization that allows searching and browsing along five dimensions simultaneously, using faceted classification. We illustrate the resulting pattern language with a sample pattern, including a discussion of our use of Business Process Modeling Notation (BPMN) and brief descriptions of all the patterns.

1. INTRODUCTION

Developing a moderately large number of patterns will inevitably present the developer with a choice: leave the patterns as a flat, unorganized list—a *pattern catalog* or *pattern collection*—or tease out the connections among the patterns and organize them into a true *pattern language*. As a recent paper [Hafiz 2011] on a security pattern language points out, pattern languages, because they organize their constituent patterns into meaningful hierarchies and networks, are inherently easier to learn, easier to navigate, and easier to apply than simple pattern catalogs.

However, as Hafiz, Adamczyk, and Johnson [2011] point out, building a pattern language out of of a pattern collection is not an easy task. It involves understanding the complex connections and interactions among patterns, as well as the intended audiences and their anticipated use cases. Christopher Alexander [Alexander1977] set the bar high, and achieving anything approaching the lucidity and grace of his language is daunting.

These issues confronted us during our insider threat pattern work at the CERT Program. By pattern mining our database of insider threat cases, our models of insider behavior, and related materials, we developed 25 insider threat patterns. But we would not have been satisfied with publishing them as a flat catalog of unrelated patterns. Insider threat must be dealt with from an enterprise architecture perspective, using a synergistic set of interdependent strategies, so it seemed obvious that we needed to build an integrated pattern language.

The challenge was to determine the best way to integrate our patterns into an organic whole. We considered and rejected six possible organizational principles for our pattern language before deciding that not one of them alone could adequately express the richness of our language. We finally realized that having to choose a single organization as the best organization was a false

---

Authors' addresses: David Mundie, email: dmundie@cert.org. A.P. Moore, email: apm@cert.org. David McIntire, email: dmmcintire@cert.org. All authors are from the Software Engineering Institute, Carnegie Mellon University, 4500 Fifth Avenue, Pittsburgh, PA 15213.
A preliminary version of this paper was presented in a writers' workshop at the 19th Conference on Pattern Languages of Programs(PLoP). PLoP'12, October 19-21, Tucson, Arizona, USA. ACM 978-1-4503-2786-2

dilemma. Our patterns existed in multiple dimensions simultaneously, and we needed a pattern language that captured all those dimensions.

In this paper, we first describe the development of our multidimensional pattern language structure. We then present an example pattern to illustrate the use of Business Process Modeling Notation (BPMN), a feature of the language that we believe is unusual, and conclude with a catalog of the constituent patterns in our language.

2. THE REJECTED APPROACHES

To illustrate the development of our pattern language, we briefly discuss each of the organizational principles we considered and rejected as the single taxonomy for our patterns.

Business Units. Many of our patterns are related to questions in the CERT Program's Insider Threat Assessment tool, which is used to assess an organization's preparedness for addressing insider threats. The questions of the Insider Threat Assessment are arranged by business units to facilitate interview schedules. Classification by business unit also facilitates implementation of the patterns; like the swim-lane diagrams of the process improvement community, arranging assessment questions by business units makes it easy for stakeholders to quickly understand what their roles are and what they must do to address insider threat.

However, organizing the patterns this way seemed contrary to what we know about the insider threat problem. Analysis of many insider threat cases has taught us that insider threats cut across business units and only enterprise-wide strategies can effectively combat them, so arranging the patterns by business units did not seem like the right approach.

Insider Threat Lifecycle Phase. Insider threat controls and capabilities are commonly divided into "prevention," "detection," and "response" activities. These three phases of the lifecycle of insider threat mitigation became our top-level categories during the pattern mining effort. Although they are widely used and immediately understood, these categories do not seem to get at the heart of insider threat.

The Zachman Framework. Throughout our pattern mining, we had assumed that our insider threat patterns would be complementary to existing information security pattern languages. We turned for inspiration to Chapter 4 of *Security Patterns*, by Schumacher et al. [Schumacher 2006], which maps the security patterns discussed in the body of the book to the Zachman framework [Zachman1987]. At this point we became aware that the difference between insider threat patterns and traditional security patterns might be greater than we had realized. In our view organizational security is not just a technology issue but also a people issue, and even more so when dealing with insider threat. Our mapping of insider threat patterns to the Zachman framework bore this out. Of our 25 patterns, 16 mapped to just 2 of the 10 cells in the Zachman diagram ("Risk management" and "Approaches"). The reason is that our patterns were all at the enterprise security strategy and policy level (Z1-2). Very few of the patterns were at the mechanisms and implementations level. Interestingly, our lifecycle classification found a home in the "Approaches" cell.

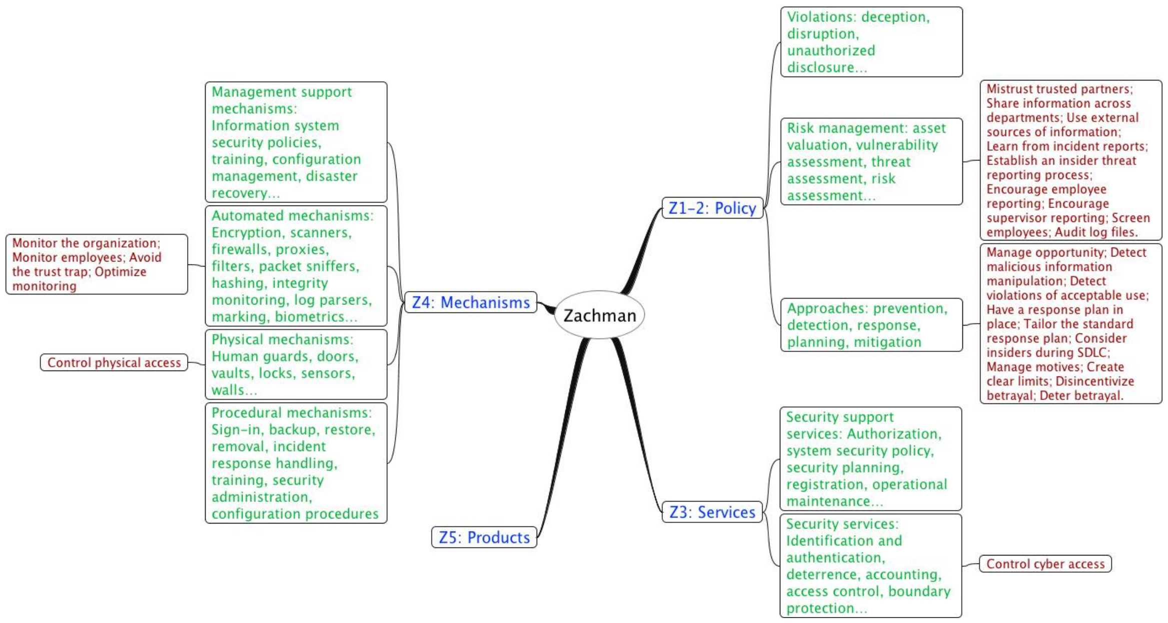

Zachman
Z1-2: Policy
Violations: deception, disruption, unauthorized disclosure...
Risk management: asset valuation, vulnerability assessment, threat assessment, risk assessment...
Mistrust trusted partners; Share information across departments; Use external sources of information; Learn from incident reports; Establish an insider threat reporting process; Encourage employee reporting; Encourage supervisor reporting; Screen employees; Audit log files.
Approaches: prevention, detection, response, planning, mitigation
Manage opportunity; Detect malicious information manipulation; Detect violations of acceptable use; Have a response plan in place; Tailor the standard response plan; Consider insiders during SDLC; Manage motives; Create clear limits; Disincentivize betrayal; Deter betrayal.
Z3: Services
Security support services: Authorization, system security policy, security planning, registration, operational maintenance...
Security services: Identification and authentication, deterrence, accounting, access control, boundary protection...
Control cyber access
Z4: Mechanisms
Management support mechanisms: Information system security policies, training, configuration management, disaster recovery...
Automated mechanisms: Encryption, scanners, firewalls, proxies, filters, packet sniffers, hashing, integrity monitoring, log parsers, marking, biometrics...
Monitor the organization; Monitor employees; Avoid the trust trap; Optimize monitoring
Physical mechanisms: Human guards, doors, vaults, locks, sensors, walls...
Control physical access
Procedural mechanisms: Sign-in, backup, restore, removal, incident response handling, training, security administration, configuration procedures
Z5: Products

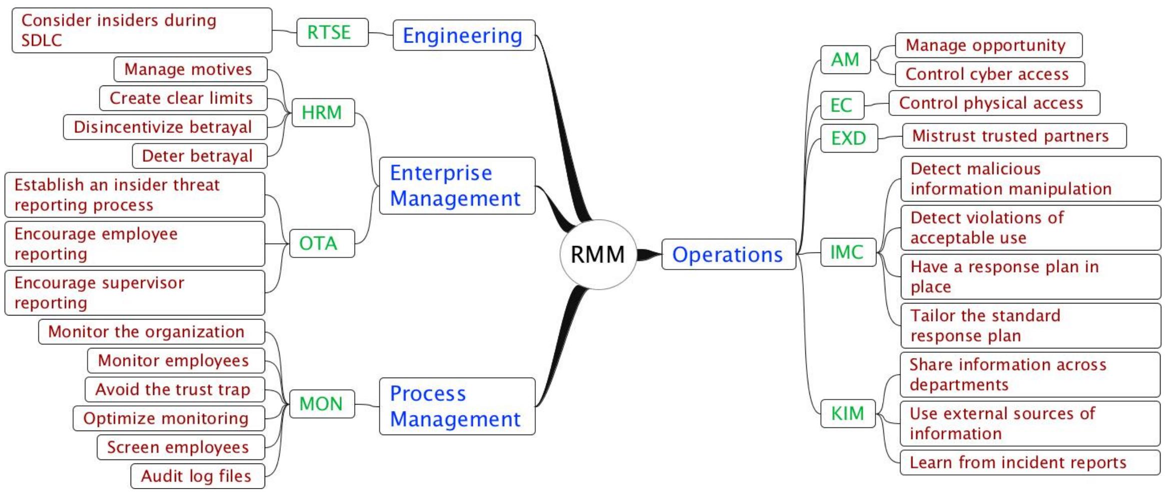
Figure 1. The insider threat pattern language mapped to the Zachman framework.

The Schumacher Landscape. Chapter 5 of *Security Patterns* [Schumacher 2006], "The Security Pattern Landscape," describes the categories that organize the body of the book. They correspond roughly to the categories in the Zachman diagram in Chapter 4. As with those categories, our insider threat patterns were most at home among the "Enterprise Security and Risk Management" patterns, and much less so among the "Identification and Authentication" patterns, the "Operating System Access Control" patterns, and the "Firewall Architecture" patterns. In retrospect the reason is obvious: the crux of the insider threat problem is that to do their jobs efficiently, insiders are trusted by their organizations and are left largely unfettered by authentication controls, access controls, and firewalls. Those technical controls need to be designed with insider threats in mind, but that is just one high-level pattern.

The Hafiz Metapattern. Next we turned to the paper "Growing a Pattern Language (for Security)" [Hafiz 2011]. This ambitious project defines a pattern language for many known security patterns. The paper has a valuable discussion of how to classify security patterns as a preliminary step to organizing them into a pattern language. The authors first considered a classification using the domain concepts of confidentiality, integrity, and availability, but they found those categories too general. Then they examined the possibility of using a Zachman-style enterprise architecture classification, but they noted that most of their security patterns fell into a small number of the cells in the table. In the end Hafiz et al. settled on a hybrid metapattern based on both the enterprise architecture scheme and a scheme based on threat models.

In theory this organization might have worked for us, but in reality it was difficult to reconcile our patterns with those in the metapattern. As with the Schumacher landscape, most of our patterns fit into just one of the seven categories in the Hafiz schema, namely the "Higher-Level Patterns." The threat model focuses on external threats, and key processes for insider threat (human resources and legal, for instance) have no place in the model.

Resilience Management Model. Finally we turned to the CERT® Resilience Management Model (CERT®-RMM). Perhaps unsurprisingly, this is a much better fit for our insider threat patterns because RMM is a broad-based model of the organizational process areas needed for resilience. Our insider threat patterns were spread fairly evenly across 9 of CERT-RMM's 26 process areas. CERT-RMM has processes that cover traditional security concerns such as asset management, monitoring, and incident management, but it also encompasses important insider threat areas that are less important for security patterns, such as human resource management, knowledge and information management, external dependencies, and organizational training and awareness. However, despite the impedance match between RMM and our patterns, we felt there were other aspects such as business units and insider threat lifecycle phases that we wanted to capture.



RMM
Engineering
RTSE
Consider insiders during SDLC
Enterprise Management
HRM
Manage motives
Create clear limits
Disincentivize betrayal
Deter betrayal
OTA
Establish an insider threat reporting process
Encourage employee reporting
Encourage supervisor reporting
Process Management
MON
Monitor the organization
Monitor employees
Avoid the trust trap
Optimize monitoring
Screen employees
Audit log files
Operations
AM
Manage opportunity
Control cyber access
EC
Control physical access
EXD
Mistrust trusted partners
IMC
Detect malicious information manipulation
Detect violations of acceptable use
Have a response plan in place
Tailor the standard response plan
KIM
Share information across departments
Use external sources of information
Learn from incident reports

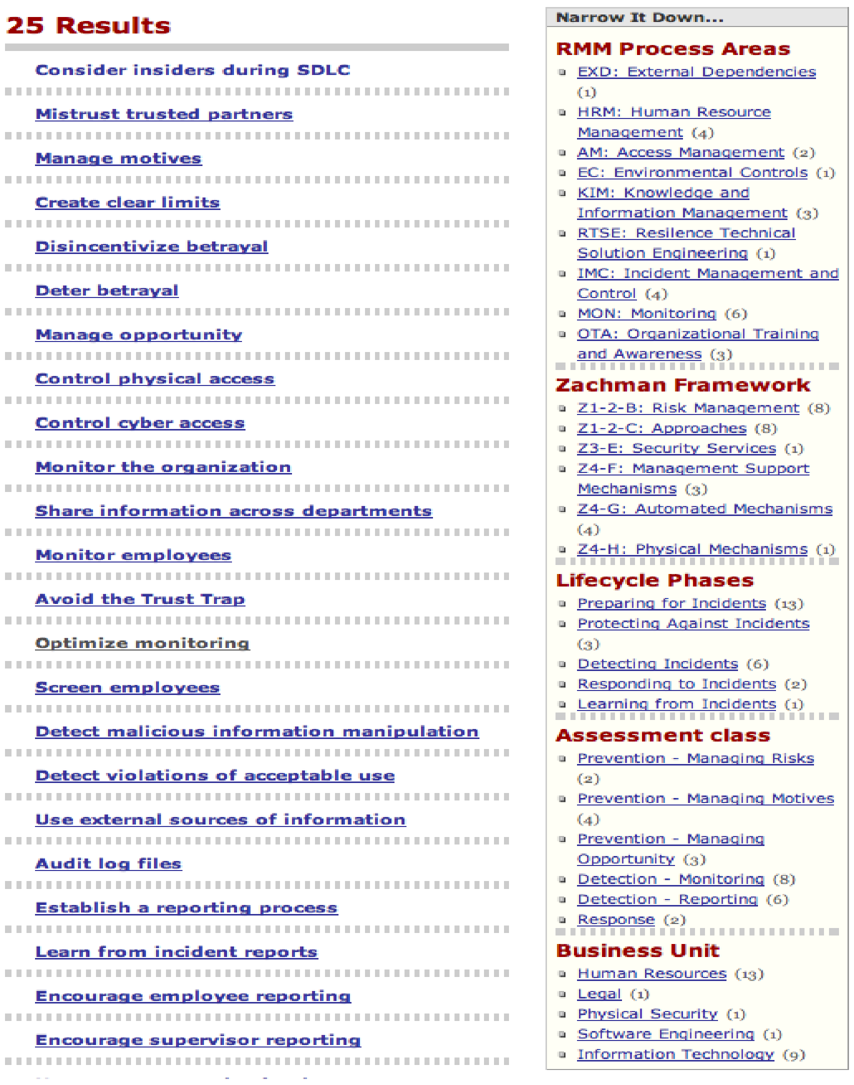
Figure 2. The insider threat pattern language mapped to CERT-RMM. Key to abbreviations: RTSE: Resilience Technology Solutions Engineering, HRM: Human Resource Management, OTA: Organizational Training and Awareness, MON: Monitoring, AM: Asset Management, EC: Environmental Controls, EXD: External Dependencies, IMC: Incident Management and Controls, and KIM: Knowledge and Information Management.

3. THE SOLUTION: A MULTIDIMENSIONAL PATTERN LANGUAGE

By the end of our investigation, we understood that we had run up against a very old problem in classification: trying to squeeze a multidimensional classification space into a one-dimensional hierarchy. It seems clear to us now that no single classification will serve all users and all use cases equally well. In the pre-PC era, there were legitimate reasons to stick with one-dimensional taxonomies, and they are still useful for many purposes, but computers offer more flexible alternatives. One alternative that has achieved some popularity for data retrieval is tagging. However, we believe that tagging is a poor choice for organizing a pattern language because it is too loosely structured and unpredictable for navigation and education.

Instead, we chose a less well-known solution to the problem of multidimensionality. Faceted classification is a library categorization technique which dates back to Condorcet's 1794 *Esquisse d'un Tableau Historique des Progrès de L'Esprit Humain* [Baker 1962]. S. R. Ranganathan's colon classification [Ranganathan 2007] is widely used in Indian libraries and is arguably the fullest expression of faceted classification, comprising 42 main classes and five colon-separated facets for finer categorization. Recently faceted classification has come to be widely used on the internet in search engines. For example, Amazon.com uses faceted classification when a user searches for a product: the website allows users to narrow and extend their search in orthogonal dimensions such as price, color, or manufacturer.

The specific instantiation of faceted classification we used is the facet map, as implemented by Facetmap software.[1] This tool allows users to browse their data in a multidimensional space. We found it straightforward to build an insider threat facet map that categorizes each of the patterns in our pattern language in a five-dimensional space defined by the six classifications described above minus the Hafiz metapattern.



25 Results

- Consider insiders during SDLC
- Mistrust trusted partners
- Manage motives
- Create clear limits
- Disincentivize betrayal
- Deter betrayal
- Manage opportunity
- Control physical access
- Control cyber access
- Monitor the organization
- Share information across departments
- Monitor employees
- Avoid the Trust Trap
- Optimize monitoring
- Screen employees
- Detect malicious information manipulation
- Detect violations of acceptable use
- Use external sources of information
- Audit log files
- Establish a reporting process
- Learn from incident reports
- Encourage employee reporting
- Encourage supervisor reporting

Narrow It Down...

RMM Process Areas
- EXD: External Dependencies (1)
- HRM: Human Resource Management (4)
- AM: Access Management (2)
- EC: Environmental Controls (1)
- KIM: Knowledge and Information Management (3)
- RTSE: Resilence Technical Solution Engineering (1)
- IMC: Incident Management and Control (4)
- MON: Monitoring (6)
- OTA: Organizational Training and Awareness (3)

Zachman Framework
- Z1-2-B: Risk Management (8)
- Z1-2-C: Approaches (8)
- Z3-E: Security Services (1)
- Z4-F: Management Support Mechanisms (3)
- Z4-G: Automated Mechanisms (4)
- Z4-H: Physical Mechanisms (1)

Lifecycle Phases
- Preparing for Incidents (13)
- Protecting Against Incidents (3)
- Detecting Incidents (6)
- Responding to Incidents (2)
- Learning from Incidents (1)

Assessment class
- Prevention - Managing Risks (2)
- Prevention - Managing Motives (4)
- Prevention - Managing Opportunity (3)
- Detection - Monitoring (8)
- Detection - Reporting (6)
- Response (2)

Business Unit
- Human Resources (13)
- Legal (1)
- Physical Security (1)
- Software Engineering (1)
- Information Technology (9)

Figure 3. The Facetmap interface.

[1] www.facetmap.com

We found two principal benefits of organizing the pattern language as a facet map. First, it enables a very usable browsing interface that allows users to drill down to the exact aspects of the patterns that are of interest to their immediate use cases. Second, the faceted classification's formal description of the pattern language organization makes it easy to generate alternate representations, such as a hierarchy with RMM process areas as the top-level categories.

Using classifications that transcend three dimensions and giving up our paper-based hierarchical structures takes some getting used to. Still, as the distinction between print and electronic documents continues to disappear, and as web searches using facet maps become ever more common, we think that multidimensional organization may make it to the mainstream.

After this work was completed, it came to our attention that there is a precedent for using faceted classification for pattern languages. In [VanHilst 2009] Michael VanHilst and his colleagues describe a six-dimensional faceted classification that they devised for their security pattern language. Their approach is similar in spirit to ours, but differs in the details because we are dealing with architectural patterns rather than design patterns. For example, their lifecycle dimension is similar to ours, but we have nothing equivalent to the "code source" dimension, which is specific to software development.

4. OVERVIEW OF THE PATTERN LANGUAGE

To illustrate the patterns from which the pattern language was derived, this section presents a single pattern in some detail and briefly describes the other patterns.

4.1 A Sample Pattern

A particularly insidious form of insider threat involves modifying the source code of software being developed within the insider's organization. Such modifications can be very difficult to detect, especially if the modifications are made to compilers or other tools for writing software.

Considering insider threats during the software development lifecycle (SDLC)

Context

The organization develops software for use externally or, more critically, internally.

Problem

At any point in the software development lifecycle (SDLC), malicious insiders can either inject malicious code or create vulnerabilities in the software that can be exploited later. Such modifications of the software can be extremely difficult to detect.

Resistance to an insider maliciously modifying code is in principle just another design consideration (such as survivability, resilience, fault-tolerance, maintainability, and security) that must be taken into account throughout the lifecycle. However, because the insider software developers cannot be trusted, insider modification of code is particularly insidious and difficult to detect.

Solution

The basics of code quality assurance (reviews, walkthroughs, etc.) are essential to preventing the corruption of the software produced, and organizations should review them to ensure they account for insider threat, especially insider collusion. Other techniques include the following:

1. *Rigorous change management and configuration management.* It is much easier to detect any malicious changes incrementally, as they are being checked into the repository, than it is to detect them after the entire project is complete.
2. *Careful selection of compilers and development tools.* Ken Thompson's 1984 Turing Award lecture pointed out the near-impossibility of detecting vulnerabilities injected by malicious compilation tools. Diverse double-compiling is a partial countermeasure.
3. *Coding standards.* A good style guide will increase the probability that code review will detect abnormal code injections.
4. *Pair programming.* The agile practice of pair programming will make code injection much more difficult because it would require the collusion of both programmers.

4.2 Overview of the Pattern Language

Our insider threat patterns were mined using materials that have been developed over a long period time within the CERT insider threat center. Elisa Bertino has recently published a useful survey [Bertino 2012] of the insider threat landscape.

- *Consider insider threats during the SDLC.* How can organizations that develop software protect against malicious modifications by insiders? Enforce configuration management, use code reviews, carefully select development tools, use coding standards, and practice pair programming.
- *Mistrust trusted partners.* How can organizations that have given external organizations access to important assets protect against malicious actions by those external organizations? Create clear and detailed service level agreements (SLAs), use visibility mechanisms, and check the external organization's credentials of attention to insider threats.
- *Manage motives.* How can organizations deter insiders' decision to attack? Create loyalty to the organization, and establish a system of sanctions.
- *Create clear limits.* How can organizations minimize the probability that insiders will maliciously break the rules of the organization? Institutionalize clear, unambiguous, enforceable policies; and establish a sustained program of security awareness training.
- *Disincentivize betrayal.* How can organizations minimize the disgruntlement that has been shown to precede most insider attacks? Play fair, intervene positively, plan ahead for reorganizations and other disruptive events.
- *Deter betrayal.* How can organizations tip the balance of the insider's scale of potential benefit versus potential harm in favor of not taking malicious action? Skillfully handle sanctions, demotions, and terminations; define a policy on punishment for insider attacks; establish procedures for evaluating employees prior to management intervention.
- *Manage opportunity.* How can organizations remove the opportunity for insiders to attack the organization's assets? Re-engineer physical and cyber access controls, taking insider threats into account.
- *Control physical access.* How can organizations guard their physical assets against unauthorized access by insiders? Tailor existing physical security policies to ensure employees cannot bypass them by virtue of their insider status; use fine-grained mechanisms to detect

unauthorized access to off-limits portions of the facility, tampering with physical security systems, and physical theft of organizational property.

- *Control cyber access.* How can organizations guard their cyber assets against unauthorized access by insiders? Use dual control for accessing critical assets, exercise meticulous account management, ensure accountability through effective change management.
- *Monitor the organization*. How can organizations understand their risk of malicious insider activity? Institute a monitoring program that collects information on the status of insider vulnerabilities and incidents within the organization.
- *Share information across departments.* How can organizations improve their insider threat monitoring by combining behavioral and technical alerts? Address privacy concerns by using abstract SIEM interfaces and weight inputs from various sensors appropriately.
- *Monitor employees.* How can organizations monitor their employees in a way that is legal, acceptable to all stakeholders, effective, and affordable? Maintain morale through fairness and communication, optimize resources, secure the results of monitoring.
- *Avoid the trust trap.* How can organizations avoid falling into a false sense of security through relaxed monitoring of trusted insiders? Use a formal language for fine-grained policies, educate managers about the trust trap, use modern access control mechanisms such as role-based access control (RBAC).
- *Use optimized monitoring for early detection.* How can organizations configure their infrastructure so that a harmful insider attack will be detected as soon as possible? Optimize sensor deployment, improve analysis algorithms, use more sensitive sensors, deploy more sensors, use sensors that react to different inputs.
- *Combine technical and behavioral monitoring.* How can organizations combine technical and behavioral monitoring to increase the effectiveness of their insider attack detection? Share alerts instead of data; create a small, ultra-trusted team that is allowed to access all data within the organization.
- *Create accurate rule sets.* How can organizations improve the effectiveness of their insider attack detection by improving the accuracy of their rule sets? Start with a large number of rules gathered from empirical data, focus on disgruntlement, use machine learning algorithms, use expert systems.
- *Screen employees and potential employees.* How can organizations minimize their exposure to candidates with past experiences that predispose them to insider attacks? Conduct background checks to decrease the time at which a malicious employee's insider threat score reaches the threshold for further investigation.
- *Detect malicious manipulation of information.* How can organizations quickly detect malicious modifications to its information assets? Keep detailed logs of accesses to critical assets, keep backups and hashcodes of the assets for comparison, instrument the products that are used to manipulate the data.
- *Detect violations of acceptable use.* How can organizations detect violations fairly and efficiently, especially given the widely varying definitions of acceptable use? Use an implementation strategy based on a cost-benefit analysis of classes of unacceptable uses.
- *Use external sources of information.* What external sources of information should organizations consider when expanding employee monitoring beyond themselves? Use information from customers and information from social networks.
- *Audit effectively.* How can organizations best leverage their investment in monitoring and logging and avoid the pitfalls in this domain? [Under development.]
- *Learn from reports.* How can organizations make good use of reports of insider activity? [Under development.]
- *Encourage reporting by coworkers.* How can organizations create an atmosphere in which employees will be more willing and able to report suspicious behavior? Create well-enforced

policies that require reporting; encourage a climate of trust, fairness, and ethical behavior through awareness training; use anonymous reporting mechanisms.

- *Encourage reporting by managers.* How can organizations ensure that managers report suspicious behavior of their direct subordinates? Maintain clear and fair policies; handle insider events tactfully and equitably; train managers to deal with problem employees; ensure that there is a smooth, well-documented mechanism for reporting insider incidents up the management chain.
- *Have an insider incident response plan in place.* How can incident handlers respond to insider attacks efficiently and in a reproducible manner? Ensure that the incident management team has a clear, institutionalized response plan for dealing with insider incidents.
- *Tailor incident response processes.* How do incident responders know how to deal with insider attacks? Tailor the standard incident response processes to deal with insider attacks.

CONCLUSION

In recent years, Clay Shirky has been a vocal critic of hierarchical organization. In *Here Comes Everybody: The Power of Organizing without Organizations* [Shirky 2008] he argues that the new technologies of social networks are doing away with hierarchical structures in our society, and in "Ontology is Overrated" [Shirky 2005] he tells the amusing story of Yahoo's efforts to maintain a hierarchical classification in face of the overwhelming mass of manifestly non-hierarchical data on the internet. In organizing our insider threat pattern language, we decided to embrace the advantages of these changes and use a multi-dimensional structure.

This material is based upon work funded and supported by the United States Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

ACKNOWLEDGEMENTS

The authors would like to thank Paul Ruggiero of the Software Engineering Institute for technical editing of this paper.

REFERENCES

[Alexander 1977] Alexander, C. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, 1977.

[Allweyer 2010] Allweyer, T. *BPMN 2.0: Introduction to the Standard for Business Process Modeling.* Books on Demand, 2010.

[Baker 1962] Baker, K.M. "An unpublished essay of Condorcet on technical methods of classification." *Annals of Science*, vol. 18, No 2, 1962, 99-123.

[Bertino 2012] Bertino, E. "Data Protection from Insider Threats." *Synthesis Lectures on Data Management.* Morgan & Claypool Publishers, 2012.

[Caralli 2011] Caralli, R.; Allen, J.; & White, D. *CERT® Resilience Management Model: A Maturity Model for Managing Operational Resilience.* Addison-Wesley, 2010.

[Hafiz 2011] Hafiz, M.; Adamczyk, P.; & Johnson, R. "Growing a Pattern Language (for Security)." *18th Conference on Pattern Languages of Programs (PLoP '11)*, October 21-23, 2011.

[Ranganathan 2007] Ranganathan, S. *Colon Classification*. Ess Ess Publications, New Delhi, 2007.

[Schumacher 2006] Schumacher, M.; Fernandez-Buglioni, E.; Hybertson, D.; Buschmann, F.; & Sommerlad, P. *Security Patterns: Integrating Security and Systems Engineering.* John Wiley & Sons, Ltd., 2006.

[Shirky 2008] Shirky, C. *Here Comes Everybody: The Power of Organizing without Organizations.* Punguin Press, New York, 2008.

This material is based upon work funded and supported by the United States Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

[Shirky 2005] “Ontology is Overrated.” Verified 2012-08-15 at shirky.com/writings.

[VanHilst 2009] M. VanHilst, E.B.Fernandez, and F. Braz, "A multidimensional classification for users of security patterns", Journal of Research and Practice in Information Technology, vol. 41, No 2, May 2009, 87-97.

[Zachman1987] Zachman, J. “A Framework for Information Systems Architecture.” *IBM Systems Journal, 26*(3), IBM Publication G321-5298, 1987.