

# Applying the Layered Decision Model to the Design of Language-Based Security Systems

Huaqiang Wei Jim Alves-Foss  
Department of Computer Science  
University of Idaho  
[wei3004,jimaf]@uidaho.edu

## Abstract

*Safeguarding practices for networked systems involves decisions in at least three areas: identification of well-defined security policies, selection of cost-effective defense strategies, and implementation of real-time defense tactics. These practices also apply to the language-based defense mechanism for a software system, which is a subset of a networked security system. Although much research has been conducted to develop language-based defense mechanisms to improve the security of software systems, the most comprehensive requirement is still the enforcement of security policies through the end-to-end control mechanism. However, the security enforcement cannot be easily achieved without a comprehensive decision model that integrates decisions about security policies, cost-effective defense strategies, and real-time defense tactics into a single, efficient framework to guide security experts in designing, developing and deploying language-based defense mechanisms in a software system. To address these problems this paper first reviews progress in language-based security defense and the layered decision modeling (LDM) technique. It then explores how to apply the LDM in the design of cost-effective language-based defense mechanisms for software systems through a sample analysis.*

**Keywords:** Language-based security; layered decision model; cost-benefit analysis; case study.

## 1. Introduction

To safeguard a networked system we need a set of well-defined security policies for regulating how this system protects sensitive and critical system resources, design and deploy cost-effective defense strategies for making use of different defense techniques, and design and invoke cost-effective real-time defense tactics for countering immediate threats [15].

As a subset of a networked system, a software system must support the above principles in building a language-based security defense. Researchers of language-based security system have developed many

defense mechanisms for improving system security [2, 3, 7, 9, 10, 11]. These defense mechanisms include reference monitors and in-line reference monitors [3], type safe [9, 10, 11, 12], certifying compilation [7], stack inspection [3, 4], end-to-end security policy enforcement [2, 9, 10], and language-based termination and transactional rollback [8].

End-to-end security policy enforcement is considered the most important mechanism in building a secure language-based system [9, 10]. Its basic principle is to ensure that security policies are enforced at each step during the information flow when executing a software system, which includes data reading, writing, processing, and propagating [9, 10]. However, traditional end-to-end security enforcement mechanisms lack explicit description of how security policies, defense strategies, and defense tactics are integrated. In addition, there is no clear way to determine which particular combinations of defense decisions will result in the most cost-effective solution.

Therefore, security enforcement cannot be easily achieved without a comprehensive decision model that integrates decisions about security policies, cost-effective defense strategies, and real-time defense tactics into a single, efficient framework that guides security experts in designing, developing and deploying language-based defense mechanisms.

To address similar problems, our earlier work proposed a layered decision model (LDM) for the design of cost-effective network defense for general network safeguarding [15], which integrates interrelated decisions of security policies, defense strategies, and defense tactics across the following three decision layers in a consistent uniform framework:

Layer Zero for determining security policies

Layer One for defense strategies

Layer Two for real-time defense tactics.

The LDM also allows tracking of costs and security policy enforcement information through the layers so that security managers can balance the costs of security investment and monitor the enforcement of security policies at each decision layer. The LDM was developed by combining risk assessment, business cost modeling, and cost-benefit analysis techniques.

As a subset of a networked system, a software system would also be a possible avenue for applying the LDM. To explore such a possibility, this paper provides a case study of applying the LDM to the design of cost-effective language-based security mechanism for a governmental software system.

## 2. Language-based security

Software systems play crucial roles in managing system resources, processing jobs, and handling requests for information. They should also play a big role in computer security, such as managing access control, memory protection, and ensuring data confidentiality and integrity. However, traditional safeguarding mechanisms treat software systems as “black boxes” and focus more on standard defense mechanisms, such as firewalls, intrusion detection systems (IDS), and encryption. Many security breaches have successfully occurred through exploitations of security holes in software systems, causing violations of information confidentiality, integrity, and availability (CIA).

Therefore, a software system may be one of the weakest security links in a networked system, and it is necessary to explore the potential vulnerabilities of software systems, design and deploy cost-effective defense mechanisms to remedy these security holes through language-based approaches. The major threats are still those that violate data confidentiality, integrity, and availability. For example, untrusted mobile codes may overrun the buffer and leak into the memory, which causes denial of service (DOS); or a computation system may intentionally or accidentally calculate incorrect data for customer accounts in a bank. To counter these threats, many defense mechanisms have been developed and deployed. Typical language-based defense mechanisms include in-line reference monitors [3], type system [9, 10, 11, 12], certifying compilation [7], stack inspection [3, 4], end-to-end informational flow analysis for security policy enforcement [2, 9, 10], and language-based termination and transactional rollback [8].

Though many language-based defense mechanisms have been applied, we are still not clear about how cost-effective these defense mechanisms are when combined as a defense strategy and whether these deployed defense mechanisms fulfill security policies and business goals of an organization. To address these problems, we apply the LDM to the design of cost-effective language-based defense system.

## 3. Applying the LDM to a software system

The LDM, as describe in detail in [15], includes three decision layers used to define the decision parameters, establish explicit relationships between decision types,

and support and record decisions made. Security policies are defined at Layer Zero, defense strategies at Layer One, and defense tactics at Layer Two.

Other model components – used variously as inputs or as “internal states” needed to make a decision – include risk assessment for known threat profiles and estimating the cost associated with each threat and related defense mechanisms. The cost-benefit analysis is based on the return-on-investment (ROI) analysis, which selects the most cost-effective defense.

We applied the LDM to the design of language-based security for a sample Web-based management software system for a local government agency. The software system was developed in VB.Net, Java Scripts, Applets, and SQL Server database. The main business of this agency is managing and regulating health care providers. This includes maintaining patients’ confidential medical records, establishing policies for disease prevention and control, issuing birth and death certificates, regulating environmental health protection, and coordinating with other government agencies.

The business goals of this system include collecting, processing and storing the above information through networks. Since the agency handles the administration of medical and health services, the security policies and defense strategies must comply with the federal Health Insurance Portability and Accountability Act (HIPAA) of 1996. HIPAA dictates strict guidelines for protecting data confidentiality and integrity. Critical data included records of birth and death, Social Security numbers (SSNs), and other protected health information (PHI) of patients as stipulated by the law.

Data privacy is at risk when data is in storage, transit, or use. Therefore, system security plans must contain protection protocols to cover these three areas of exposure. In addition to PHI protection, the agency must also protect information that is critical for business continuity, since users need to query these data for statistic analysis on a daily basis. Therefore, system availability is also very critical.

### 3.1 Risk assessment and threat identification

The major threats fall into the following categories:

**Denial of service (DoS) attack** DoS attacks try to take down the system by overwhelming system resources, such as memory, stack and system files, which cause services and resources to become unavailable, or cause memory leakage.

**Divulging confidential information (DCI)** Confidential information includes personal ID, financial account numbers, SSNs, medical records, etc. A malicious code may divulge confidential information through implicit information leaking during the execution of the program and expose data to lower trust-level agents during or after execution.

**Root access attack (RA)** Through exploiting software system security holes, attackers can obtain root access to the operating system (OS) and install tools to view, track, modify, and steal customers' information and change system files or modify account data.

**Unauthorized access attack (UA)** Malicious codes can bypass authentication and obtain unauthorized access to data, which may cause data from high trust-level agents to be disclosed to low trust-level agents.

To quantitatively assess these threats, security managers employed risk assessment and cost modeling to estimate the business impact of each type of threat. This assessment helped rank the severity of each kind of threat. The security managers estimated the SLE of each threat, then extrapolated each SLE to estimate the ALE of each threat. We treated ALE as the threat index (TI) indicating how critical the threat was (Table 1).

The challenging tasks for risk assessment and ROI analysis include estimating cost values, effectiveness of specific defense techniques when countering specific threats, and annual attacking frequency of all threats. Butler's research [1] adopted a range of effectiveness, instead of a fixed value, when assessing a defense technique, which can also be applied to the estimates of other variables. Another possible approach is to estimate costs, effectiveness, and annual attacking frequency with a confidence level, which indicates how likely each estimate is based on security managers' experience and knowledge. To simplify the decision making process in this paper, we use fixed values for these estimates. Future research work will be conducted to investigate how a simulation approach can be used to generate input scenarios to test the model performance.

### 3.2 Design of security policies (Layer Zero)

Security policies focus on protecting data privacy, restricting online access to data, monitoring and auditing resource access to prevent DoS attacks and unauthorized access attacks. Table 2 lists the security policies that can be fulfilled by the software system. When defining security policies and design defense mechanisms, we must observe two basic principles [10]:

1. Principle of Least Privilege: each participant should be assigned the minimum access necessary to accomplish its task
2. Principle of Minimal Trusted Computing Base: the security mechanism be small and simple.

### 3.3 Design of defense strategies (Layer One)

At Layer One we designed two defense strategies,  $S_{t,b}^1$  and  $S_{t,b}^2$ , and compared them based on their cost-effectiveness. Table 3 lists the defense techniques of  $S_{t,b}^1$ . Table 4 lists the defense techniques of  $S_{t,b}^2$ .

### 1) Defense Strategy One ( $S_{t,b}^1$ )

$S_{t,b}^1$  includes an in-line reference monitor (IRM), type system (TS) and certified compilation (CP) as the major defense techniques. The cost of each defense technique includes the labor cost for developing and deploying this mechanism, material cost (hardware, software and facility cost), training and maintenance costs. An additional cost is resource overhead which reflects the extra resources and time needed to execute and invoke the security mechanism.

IRM works through program rewriting, which can prevent reads, writes or branches to memory locations outside predefined memory regions [10]. When a security event (normal event or attack) happens, the IRM will check the security state, which stores information about earlier security events that can be used to determine which security event is allowed to proceed. Then, the program fragments will be executed to respond to security events, update the security state, and take other necessary responsive actions, such as signaling security violations and notifying the security managers, blocking/terminating execution of a program, rolling back to previous system points, etc. [10].

TS can check the typing and security properties statically at compilation-time or dynamically at runtime. We need to invest time in the development stage to ensure the program is developed with good type-safe features and that program operations are only applied to appropriate values to ensure memory safety (the program can only access appropriate memory locations) and control safety (the program can only transfer control to appropriate program points) [10]. For example, this can enforce the security policy by reducing DoS attacks due to memory overrun.

In the meantime, the CP guarantees that the code satisfying a security policy has a certificate – machine checkable evidence. This can speed up the checking of security enforcement and reduce the trusted computing base (TCB) of the program.

$S_{t,b}^1$  can monitor system resources and program execution statically and dynamically to protect memory spaces. The disadvantage is that the overhead cost of IRM is big, though the CP can reduce the cost to some extent. However, the integration of IRM, CP and TS is very effective in countering these threats.

### 2) Defense Strategy Two ( $S_{t,b}^2$ )

$S_{t,b}^2$  includes strong typing (ST), bytecode verification (BV) and stack inspection (SI). SI is the generic feature of .NET and Java. SI is implemented in run-time systems such as the Java Virtual Machine (JVM) and .NET environment to accommodate components with diverse levels of trust [3, 4]. SI supports dynamic authorization mechanisms, but it has rather complex and subtle semantics.

**Table 1** Threat and rankings

Threats	Estimated Annual Frequency	Single Loss Expectancy (SLE)	Annual Loss Expectancy (ALE)	Ranking
Divulging confidential information (DCI) ( $t_1$ )	40	\$10,000	\$400,000	1
Root access attack (RA) attack ( $t_2$ )	5	\$10,000	\$50,000	4
Denial of service (DoS) attack ( $t_3$ )	5	\$20,000	\$100,000	3
Unauthorized access (UA) attack ( $t_4$ )	15	\$20,000	\$300,000	2

**Table 2** Security policies ( $P_{t,b}$ ) and coverage

Security policies	Coverage
$p_1$ : If the Web server is substantially slower than normal, the security manager may need to restart it or switch the service to a backup server.	$t_3$
$p_2$ : Improper communication between servers must be recognized and blocked.	$t_2, t_4$
$p_3$ : Access to system resources must be actively monitored and audited.	$t_1 - t_4$
$p_4$ : No unapproved software may be installed on any workstation without authorization from the security managers.	$t_1 - t_4$
$p_5$ : No e-mail or Internet access is allowed on critical database servers.	$t_1, t_4$
$p_6$ : Online access to data must be controlled.	$t_1, t_4$
$p_7$ : Noninterference policy: a variation of high confidential input does not cause a variation of low confidential public output.	$t_1$

**Table 3** Defense Strategy One ( $S_{t,b}^1$ )

Defense Techniques	Purpose
In-line Reference Monitor (IRM) ( $s_1$ )	Access control, security policy enforcement (major threats: $t_1-t_4$ )
Certified compilation (CP) ( $s_2$ )	Check if the code is legitimate (major threats: $t_1 - t_4$ )
Type system (TS) ( $s_3$ )	Security policy enforcement (major threats: $t_1 - t_4$ )

**Table 4** Defense Strategy Two ( $S_{t,b}^2$ )

Defense Techniques	Purpose
Stack inspection (SI) ( $s_1$ )	Security policy enforcement (major threats: $t_1 - t_4$ )
Strong typing (ST) ( $s_2$ )	Security policy enforcement (major threats: $t_1 - t_4$ )
Bytecode verification (BV) ( $s_3$ )	Security policy enforcement (major threats: $t_1 - t_4$ )

**Table 5** Cost-benefit analysis for defense strategies ( $k = 1,000$ )

Threat	$S_{t,b}^1$		$S_{t,b}^2$	
	Effectiveness	Benefit	Effectiveness	Benefit
Divulging confidential information (DCI) ( $t_1$ )	0.85	Benefit(1,DCI) = \$170k	0.6	Benefit(2,DCI) = \$120k
Root access attack (RA) attack ( $t_2$ )	0.75	Benefit(1, RA) = \$38k	0.5	Benefit(2, RA) = \$25k
Denial of service (DoS) attack ( $t_3$ )	0.75	Benefit(1,DoS) = \$30k	0.7	Benefit(2,DoS) = \$28k
Unauthorized access (UA) attack ( $t_4$ )	0.70	Benefit(1,UA) = \$14k	0.6	Benefit(2,UA) = \$12k
Investment cost of each defense strategy		\$200k		\$120k
Expected benefit (cost saving)		\$470k		\$320k
ROI		2.35		2.66

**Table 6** Cost-benefit analysis for defense tactics (k = 1,000)

Threat \ Cost	Damage cost (SLE)	Defense tactics	Response cost	Operational cost	Effectiveness	ROI
Divulging confidential information (DCI) ( $t_1$ )	\$10k	S+TE	\$2k	\$1k	0.6	2.0
		D+TE	\$3k	\$2k	0.7	1.4
		S+D+TE	\$2k	\$3k	0.8	1.6
		S+RB	\$5k	\$1k	0.5	0.9
		D+RB	\$3k	\$2k	0.6	1.2
		S+D+RB	\$2k	\$3k	0.7	1.4
		S+NF	\$5k	\$1k	0.7	1.2
		D+NF	\$5k	\$2k	0.8	1.2
		S+D+NF	\$5k	\$3k	0.9	1.2
		Root access (RA) attack ( $t_2$ )	\$10K	S+TE	\$2k	\$1k
D+TE	\$3k			\$2k	0.7	1.4
S+D+TE	\$2k			\$3k	0.9	1.8
S+RB	\$5k			\$1k	0.5	0.9
D+RB	\$3k			\$2k	0.6	1.2
S+D+RB	\$2k			\$3k	0.7	0.6
S+NF	\$5k			\$1k	0.4	0.8
D+NF	\$5k			\$2k	0.5	1.0
S+D+NF	\$5k			\$3k	0.6	0.8
Denial of service (DoS) attack ( $t_3$ )	\$20k			S+TE	\$2k	\$1k
		D+TE	\$3k	\$2k	0.8	3.2
		S+D+TE	\$2k	\$3k	0.9	3.6
		S+RB	\$5k	\$1k	0.6	2.0
		D+RB	\$3k	\$2k	0.7	2.8
		S+D+RB	\$2k	\$3k	0.8	3.2
		S+NF	\$5k	\$1k	0.5	1.8
		D+NF	\$5k	\$2k	0.6	1.8
		S+D+NF	\$5k	\$3k	0.7	1.8
		Unauthorized access (UA) attack ( $t_5$ )	\$20k	S+TE	\$2k	\$1k
D+TE	\$3k			\$2k	0.7	2.8
S+D+TE	\$2k			\$3k	0.8	3.2
S+RB	\$5k			\$1k	0.5	1.8
D+RB	\$3k			\$2k	0.6	2.4
S+D+RB	\$2k			\$3k	0.7	2.8
S+NF	\$5k			\$1k	0.5	1.8
D+NF	\$5k			\$2k	0.6	1.8
S+D+NF	\$5k			\$3k	0.7	1.8

the code is type safe before running. Strong typing (ST) is a feature of programming languages that enforces basic correctness of properties on how the program manipulates data. It can be enforced dynamically or statically. Dynamic enforcement can halt program execution if it is ill-typed; static enforcement can reject the ill-typed program at compile time [3, 4].

Although Defense Strategy One might have higher effectiveness when countering overall threats, its cost is also large. On the other hand, Defense Strategy Two has a low cost. Table 5 lists the cost data and cost-benefit analysis results of the two defense strategies. ROI is the ratio of cost saving to security investment [15]. From the results we can see that Strategy Two ( $S_{t,b}^2$ ) has slightly higher cost-effectiveness than Strategy One ( $S_{t,b}^1$ ).

### 3.4 Design of defense tactics (Layer Two)

If an attack is going on in the system, we must detect it and respond to it through specific defense tactics ( $R_{t,b}$ ). In this paper, we define a defense tactic as a course of operational actions for detecting attacks and a course of responsive actions for mitigating attacks.

Therefore, the cost of a defense tactic includes the operational cost and response cost.

The operational actions include either static check, dynamic check, or their combination. Combining both static check and dynamic check may cost more, but the effectiveness may also be higher.

To respond to an attack, we must invoke specific responsive actions, which include “terminate execution of the program,” “rollback to previous program points,” “notify security managers,” etc. The cost associated with responsive actions is called response cost, which also depends on the responsive action. To closely estimate each cost is impossible, but we can use estimates of relative costs. In addition we estimate the effectiveness of each defense tactic.

The benefit is saving a damage cost if an attack is unsuccessful. Table 6 lists some combinations of operational actions and responsive actions of Defense Strategy Two, their costs, benefits, and ROIs. We use D to represent “dynamic check”, S to represent “static check”, and S+D to represent “combination of static check and dynamic check”. We also use TE to represent “terminate the execution of program”, RB to represent

“rollback to previous program points”, and NF to represent “notify security managers”. The result shows that combining “static check”, “dynamic check” and “terminate execution of program” (S+D+TE) is the best defense tactic against a root access attack. Combining “static check” and “terminate execution of program” (S+TE) is the best defense tactic for the rest of the threats. In summary, this sample case study provided concrete examples of how to apply the LDM in guiding security managers in designing cost-effective defense for language-based systems. The most challenging job is estimating item costs and annual frequencies of all threats, as well as the effectiveness of defense techniques. Overall, we received fairly good responses from those who participated in this case study, although they had questions about the overall utility of the model when imprecise data is all that is available.

## 4. Conclusions

To safeguard software systems through language-based approach, our research applied a uniform layered decision model (LDM) to the design of cost-effective mechanisms for software systems. The LDM supports consistently connected decisions at three decision layers: Layer Zero for determining security policies, Layer One for defense strategies, and Layer Two for real-time defense tactics. In addition, the LDM supports an analytical framework that allows traceability of costs and security policy enforcement information between all decision layers, conducts cost-benefit analyses for selecting the cost-effective defense plans, and monitors security enforcement. Furthermore, the LDM encourages an iterative, traversing decision process between and among decision layers. To demonstrate how the LDM can be used for software system security, we conducted a sample case study that applied the LDM to the design of a cost-effective language-based defense mechanism for a government agency’s software system. This case study provided a concrete example of how to use the model.

## 5. References

- [1] S. Butler, “Security attribute evaluation method: A cost-benefit approach”, *International Conference on Software Engineering*, 2002, pp. 232–241.
- [2] U. Erlingsson and F. Schneider, “SASI enforcement of security policies: A retrospective”, In *New Security Paradigms Workshop*, 1999, pp. 87–95.
- [3] U. Erlingsson and F. Schneider, “IRM enforcement of Java stack inspection”, In *IEEE Symposium on Security and Privacy*, 2000, pp. 246–255.
- [4] C. Fournet and A. Gordon, “Stack inspection: Theory and variants”, In *ACM Symposium on Principles of Programming Languages (POPL)*, 2002, pp. 307–318.
- [5] G. Klein and M. Strecker, “Verified Bytecode Verification and Type-certifying Compilation”, *Journal of Logic Programming*, 2004, 58 (1-2): pp. 27–60.
- [6] W. Lee, W. Fan, M. Miller, S. Stolfo and E. Zadok, “Toward cost-sensitive modeling for intrusion detection and response”, *Computer Security*, 2002, 10(1-2): pp.5–22.
- [7] G. Necula and P. Lee, “The design and implementation of a certifying compiler”, In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 1998, pp. 333–344.
- [8] A. Rudys and D. Wallach, “Termination in language-based systems”, *ACM Transactions on Information and System Security*, 2002, 5(2), pp. 138–168.
- [9] A. Sabelfeld and A. Myers, “Language-based information-flow security”, *IEEE Journal on Selected Areas in Communication, Special Issue on Formal Methods for Security*, 2003, 21(1): pp. 5–19.
- [10] F. Schneider, G. Morrisett and R. Harper, “A language-based approach to security”, *Informatics: 10 Years Back, 10 Years Ahead*, 2000, pp. 86–101.
- [11] S. Tse and S. Zdancewic, “Run-time principals in information-flow type systems”, In *IEEE Symposium on Security and Privacy*, 2004, pp. 179–193.
- [12] D. Walker, “A type system for expressive security policies”, In *ACM SIGPLAN Symposium on Principles of Programming Languages*, 2000, pp. 254–267.
- [13] H. Wei, D. Frincke, O. Carter and C. Ritter, “Cost-benefit analysis for network intrusion detection systems”, *CSI 28th Annual Computer Security Conference*, 2001.
- [14] H. Wei and D. Frincke, “Risk assessment and cost-effective business modeling for network security”, *The 7th World Multi-Conference on Systemics, Cybernetics and Informatics SCI 2003*, 2003.
- [15] H. Wei, D. Frincke, J. Alves-Foss, T. Soule and H. Pforsich, “A layered decision model for cost-effective network defense”, *IEEE International Conference on Information Reuse and Integration*, 2005, pp. 506–511.