Performance-sensitive Real-time Risk Management is NP-Hard

Ashish Gehani
Department of Computer Science, Duke University

Abstract

This paper introduces a formal model for quantifying host-based risk, and two classes of primitives that can be utilized to manage it. The performance overhead introduced by selecting a set of response primitives to manage the risk is also factored into the framework. The resulting problem, of managing risk while minimizing the impact on performance is shown to be NP-hard. Since the goal is real-time response, a heuristic is described that allows the first primitive to be chosen in constant time (which is frequently sufficient to disrupt an attack).

1 Introduction

If a system is simple, its properties can be completely ascertained, either analytically or empirically. When a system is complex, analytical tools can not address all issues and empirical techniques require more resources than can typically be devoted to the task of verification. To secure an information processing system, it is necessary to ensure that it obeys a set of rules and maintains a set of properties. Modern computing systems are complex. This makes it infeasible to address the task empirically. Analytical techniques can alleviate the issue, but they can not resolve it completely [Harrison76]. In this context, *risk* serves as a measure of the extent to which the security of the system is likely to be violated.

Early approaches to computer security *risk management* employed static strategies, such as the use of passwords, discretionary or mandatory access control, and encrypted network connections [Fletcher95]. These approaches did not provide the flexibility needed to allow risk managers to alter the levels of risk they were willing to tolerate in exchange for commensurate costs. Hence, techniques to dynamically vary the risk were developed. Inherent in the new approach was the need for *risk analysis* to quantify the level of risk present in each configuration of a system.

Large data processing centers started to use the Annual Loss Expectancy (ALE) metric [FIPS31], [FIPS65]. To compute it, the set of all possible *hazards* that could impact the system over the course of a year was enumerated as $H = \{h_1, h_2, \ldots, h_n\}$. The loss associated with each hazard h_α was denoted by $l(h_\alpha)$ and the frequency with which it was likely to occur was denoted by $f(h_\alpha)$. The risk was then calculated using:

$$ALE = \sum_{\alpha=1}^{\alpha=n} f(h_\alpha) \times l(h_\alpha) \tag{1}$$

The utilization of the paradigm by a number of commercial tools [NIST91] coupled with a focused research effort [CSRMMBW88], [CSRMMBW89], [CSRMMBW90], [CSRMMBW91] resulted in a number of improvements. The hazard construct was decomposed into threat and vulnerability components. The likelihood of a threat being present was added as a factor, replacing the hazard frequency. Each vulnerability was coupled with an associated safeguard. The loss from a hazard's occurrence was modeled as a set of consequences that could affect each asset under consideration. Finally, risk management was formulated as the maintenance of a set of requirements framed as constraints on the aforementioned factors [NIST800-12].

This paper contributes specific semantics in the operating system paradigm for the generic risk analysis concepts of threats, likelihoods, exposures, safeguards, assets and consequences. These are then utilized to construct a model for managing a host's risk in real-time.

2 Runtime Risk Management

The primary goal of an intrusion response system is to guard against attacks. Primitives that address specific threats have been developed. However, invoking these arbitrarily may safeguard part of the system but leave other weaker areas exposed. Thus, to effect a rational response, it is necessary to weigh all the possible alternatives. A course of action must then be chosen which will result in the least damage, while simultaneously assuring that cost constraints are respected. This is the very problem that risk management addresses.

2.1 Response Primitives

Two types of response primitives are required. They need modifications of the access control subsystem and the filesystem, respectively. The first type institutes further runtime checks before granting specific permissions, in order to reduce the host exposure. The second type requires data to be stored in a protected state, and allows the rapid disabling of transparent decryption, signing of modifications and remote replication of specific files, in order to curtail the consequences of an attack. An instance of either type can be chosen as a response at any time. Choosing a primitive imposes an overhead on system performance that is proportional to the frequency with which the primitive is used in a typical workload.

2.2 Risk Factors

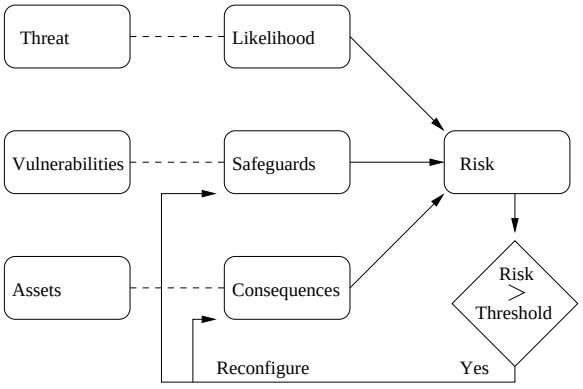

Figure 1: Risk can be analyzed as a function of the threats, their likelihood, the vulnerabilities, the safeguards, the assets and the consequences. Risk can be managed by using the safeguards to control the exposure of vulnerabilities and manipulating the assets to limit the consequences.

Analyzing the risk that a system is faced with requires knowledge of a number of factors. Below we describe each of these factors along with its associated semantics. We define these in the context of the operating system paradigm since our goal is host-based response.

The paradigm assumes the existence of an operating system with a trusted reference monitor that mediates access by subjects to objects in the system. In addition, the file system and auditing subsystem are assumed to be trusted components in the operating system. Finally, a host-based intrusion detection system is assumed to be present and operational.

Threats A *threat* is an agent that can cause harm to an asset in the system. We define a threat to be a specific attack against any of the application or system software that is running on the host. It is characterized by an intrusion detection signature. The set of threats is denoted by $T = \{t_1, t_2, \ldots\}$, where $t_\alpha \in T$ is an intrusion detection signature. Since t_α is a host-based signature, it is comprised of an *ordered set* of events $S(t_\alpha) = \{s_1, s_2, \ldots\}$. If this set occurs in the order recognized by the rules of the intrusion detector, it signifies the presence of an attack.

Likelihood The *likelihood* of a threat is the hypothetical probability of it occurring. If a signature has been partially matched, the extent of the match serves as a predictor of the chance that it will subsequently be completely matched. A function μ is used to compute the likelihood of threat t_α. μ can be threat specific and will depend on the history of system events that are relevant to the intrusion signature. Thus, if $E = \{e_1, e_2, \ldots\}$ denotes the ordered set of all events that have occurred, then:

$$\mathcal{T}(t_\alpha) = \mu(t_\alpha, E \overset{\prec}{\cap} S(t_\alpha)) \tag{2}$$

where $\overset{\prec}{\cap}$ yields the set of all events that occur *in the same order* in each input set.

Assets An *asset* is an item that has value. We define the assets to be the data stored in the system. In particular, each file is considered a separate object $o_\beta \in O$, where $O = \{o_1, o_2, \ldots\}$ is the set of assets. A set of objects $A(t_\alpha) \subseteq O$ is associated with each threat t_α. Only objects $o_\beta \in A(t_\alpha)$ can be harmed if the attack that is characterized by t_α succeeds.

Consequences A *consequence* is a type of harm that an asset may suffer. Three types of consequences can impact the data. These are the loss of confidentiality, integrity and availability. If an object $o_\beta \in A(t_\alpha)$ is affected by the threat t_α, then the resulting costs due to the loss of confidentiality, integrity and availability are denoted by $c(o_\beta)$, $i(o_\beta)$, and $a(o_\beta)$ respectively. Any of these values may be 0 if the attack can not effect the relevant consequence. However, all three values associated with a single object can not be 0 since in that case $o_\beta \in A(t_\alpha)$ would not hold. Thus, the consequence of a threat t_α is:

$$\mathcal{C}(t_\alpha) = \sum_{o_\beta \in A(t_\alpha)} c(o_\beta) + i(o_\beta) + a(o_\beta) \tag{3}$$

By removing an asset from the system, the consequences it faces can be *curtailed* [Gehani03]. In the case of data availability, replication serves this purpose, while in the case of confidentiality and integrity, cryptographic operations can be used. For the purpose of estimating risk, a consequence *curtailment* effectively removes the asset from the analysis.

Vulnerabilities A *vulnerability* is a weakness in the system. It results from an error in the design, implementation or configuration of either the operating system or application software. The set of vulnerabilities present in the system is denoted by $W = \{w_1, w_2, \ldots\}$. $W(t_\alpha) \subseteq W$ is the set of weaknesses exploited by the threat t_α to subvert the security policy.

Safeguards A *safeguard* is a mechanism that controls the exposure of the system's assets. The reference monitor's set of permission checks $P = \{p_1, p_2, \ldots\}$ serve as safeguards in an operating system. Since the reference monitor mediates access to all objects, a vulnerability's exposure can be limited by denying the relevant permissions. The set $P(w_\gamma) \subseteq P$ contains all the permissions that are requested in the process of exploiting vulnerability w_γ. The static configuration of a conventional reference monitor either grants or denies access to a permission p_λ. This *exposure* is denoted by $v(p_\lambda)$, with the value being either 0 or 1. An *active reference monitor* [1] [Gehani03] can reduce the exposure of a statically granted permission to $v'(p_\lambda)$, a value in the range $[0, 1]$. This reflects the nuance that results from evaluating predicates as *auxiliary safeguards*.)

Thus, if all auxiliary safeguards are utilized, the total exposure to a threat t_α is:

$$\mathcal{V}(t_\alpha) = \sum_{p_\lambda \in \hat{P}(t_\alpha)} \frac{v(p_\lambda) \times v'(p_\lambda)}{|\hat{P}(t_\alpha)|} \tag{4}$$

[1] An *active reference monitor* allows each permission to be associated with an independent set of constraints which are verified at runtime prior to granting the permission. By limiting the circumstances under which the permission will be granted, the exposure of the resource being protected is reduced by a pre-determined fraction.

where:

$$\hat{P}(t_\alpha) = \bigcup_{w_\gamma \in W(t_\alpha)} P(w_\gamma) \tag{5}$$

2.3 Risk Analysis

The risk to the host is the sum of the risks that result from each of the threats that it faces. The risk from a single threat is the product of the chance that the attack will occur, the exposure of the system to the attack, and the cost of the consequences of the attack succeeding [NIST800-12]. Thus, the cumulative risk faced by the system is:

$$\mathcal{R} = \sum_{t_\alpha \in T} \mathcal{T}(t_\alpha) \times \mathcal{V}(t_\alpha) \times \mathcal{C}(t_\alpha) \tag{6}$$

2.4 Risk Management

If the risk posed to the system is to be managed, the current level must be continuously monitored. When the risk rises past the threshold that the host can tolerate, the system's security must be tightened. Similarly, when the risk decreases, the restrictions can be relaxed to improve performance and usability. This process is elucidated below.

The system's risk can be reduced either by reducing the exposure of vulnerabilities or limiting the consequences to the data in the event of a successful attack. The former is effected through the use of auxiliary safeguards prior granting a permission. The latter is realized by cryptographically protecting threatened files. Additionally, both approaches may be used simultaneously. Similarly, if the threat reduces, the restrictive permission checks and data protection can be relaxed.

2.4.1 Managed Risk

The set of permissions P is kept partitioned into two disjoint sets, $\Psi(P)$ and $\Omega(P)$, that is $\Psi(P) \cap \Omega(P) = \phi$ and $\Psi(P) \cup \Omega(P) = P$. The set $\Psi(P) \subseteq P$ contains the permissions for which auxiliary safeguards are currently active. The remaining permissions $\Omega(P) \subseteq P$ are handled conventionally by the reference monitor, using only static lookups rather than evaluating associated predicates prior to granting these permissions. Similarly, the set of files O is kept partitioned into two disjoint sets, $\Psi(O)$ and $\Omega(O)$, where $\Psi(O) \cap \Omega(O) = \phi$ and $\Psi(O) \cup \Omega(O) = O$. The set $\Psi(O) \subseteq O$ contains the files that are currently inaccessible and unmodifiable due to their cryptographic encapsulation. The remaining files $\Omega(O) \subseteq O$ are transparently accessible and modifiable.

At any given point, when safeguards $\Psi(P)$ and curtailments $\Psi(O)$ are in use, the current risk $\mathcal{R}'$ is calculated with:

$$\mathcal{R}' = \sum_{t_\alpha \in T} \mathcal{T}(t_\alpha) \times \mathcal{V}'(t_\alpha) \times \mathcal{C}'(t_\alpha) \tag{7}$$

where:

$$\mathcal{V}'(t_\alpha) = \sum_{p_\lambda \in \hat{P}(t_\alpha) \cap \Omega(P)} \frac{v(p_\lambda)}{|\hat{P}(t_\alpha)|} + \sum_{p_\lambda \in \hat{P}(t_\alpha) \cap \Psi(P)} \frac{v(p_\lambda) \times v'(p_\lambda)}{|\hat{P}(t_\alpha)|} \tag{8}$$

and:

$$\mathcal{C}'(t_\alpha) = \sum_{o_\beta \in A(t_\alpha) \cap \Omega(O)} c(o_\beta) + i(o_\beta) + a(o_\beta) \tag{9}$$

2.4.2 Risk Tolerance

While the risk must be monitored continuously, there is a computational cost incurred each time it is recalculated. Therefore, the frequency with which the risk is estimated must be minimized to the extent possible. Instead of calculating the risk synchronously at fixed intervals in time, we exploit the fact that the risk level only changes when the threat to the system is altered.

An intrusion detector is assumed to be monitoring the system's activity. Each time it detects an event that changes the extent to which a signature has been matched, it passes the event e to the intrusion response subsystem. The level of risk $\mathcal{R}_b$ before e occurred is noted, and then the level of risk $\mathcal{R}_a$ after e occurred is calculated. Thus, $\mathcal{R}_a = \mathcal{R}_b + \epsilon$, where ϵ denotes the change in the risk. Since the risk is recalculated only when it actually changes, the computational cost of monitoring it is minimized.

Each time an event e occurs, either the risk decreases, stays the same or increases. Each host is configured to tolerate risk upto a threshold, denoted by $\mathcal{R}_0$. After each event e, the system's response guarantees that the risk will return to a level below this threshold. As a result, $\mathcal{R}_b < \mathcal{R}_0$ always holds. If $\epsilon = 0$, then no further risk management steps are required.

If $\epsilon < 0$, then $\mathcal{R}_a < \mathcal{R}_0$ since $\mathcal{R}_a = \mathcal{R}_b + \epsilon < \mathcal{R}_b < \mathcal{R}_0$. At this point, the system's security configuration is more restrictive than it needs to be. To improve system usability and performance, the response system must deactivate appropriate safeguards and curtailments, while ensuring that the risk level does not rise past the threshold $\mathcal{R}_0$.

If $\epsilon > 0$ and $\mathcal{R}_a \leq \mathcal{R}_0$, then no action needs to be taken. Even though the risk has increased, it is below the threshold that the system can tolerate, so no further safeguards or curtailments need to be introduced. In addition, the system will not be able to find any set of unused safeguards and curtailments whose removal will increase the risk by less than $\mathcal{R}_0 - \mathcal{R}_b - \epsilon$, since the presence of such a combination would also mean that the set existed before e occurred. It is not possible that such a combination of safeguards and curtailments existed before e occurred since they would also have satisfied the condition of being less than $\mathcal{R}_0 - \mathcal{R}_b$ and would have been utilized before e occurred in the process of minimizing the impact on performance in the previous step.

If $\epsilon > 0$ and $\mathcal{R}_a > \mathcal{R}_0$, then action is required to reduce the risk to a level below the threshold of tolerance. The response system must search for and implement a set of safeguards and curtailments to this end.

2.4.3 Recalculating Risk

When the risk is calculated the first time, Equation 6 is used. Therefore, the cost is $O(|T| \times |P| \times |O|)$. Since the change in the risk must be repeatedly evaluated during real-time reconfiguration of the runtime environment, it is imperative the cost is minimized. This is achieved by caching all the values $\mathcal{V}'(t_\alpha) \times \mathcal{C}'(t_\alpha)$ associated with threats $t_\alpha \in T$ during the evaluation of Equation 6. Subsequently, when an event e occurs, the change in the risk $\epsilon = \delta(\mathcal{R}', e)$ can be calculated with cost $O(|T|)$ as described below.

The ordered set E refers to all the events that have occurred in the system prior to the event e. The change in the likelihood of a threat t_α due to e is:

$$\delta(\mathcal{T}(t_\alpha), e) = \mu(t_\alpha, (E \cup e) \overset{\prec}{\cap} S(t_\alpha)) - \mu(t_\alpha, E \overset{\prec}{\cap} S(t_\alpha)) \tag{10}$$

The set of threats affected by e is denoted by $\Delta(T, e)$. A threat $t_\alpha \in \Delta(T, e)$ is considered to be affected by e if $\delta(\mathcal{T}(t_\alpha), e) \neq 0$, that is its likelihood changed due to the event e. The resultant change in the risk level is:

$$\delta(\mathcal{R}', e) = \sum_{t_\alpha \in \Delta(T,e)} \delta(\mathcal{T}(t_\alpha), e) \times \mathcal{V}'(t_\alpha) \times \mathcal{C}'(t_\alpha) \tag{11}$$

2.5 Cost/Benefit Analysis

After an event e occurs, if the risk level $\mathcal{R}_a$ increases past the threshold of risk tolerance $\mathcal{R}_0$, the goal of the response engine is to reduce the risk by $\delta_g \geq \mathcal{R}_a - \mathcal{R}_0$ to a level below the threshold. To do this, it must select a subset of permissions $\rho(\Omega(P)) \subseteq \Omega(P)$ and a subset of objects $\rho(\Omega(O)) \subseteq \Omega(O)$, such that adding safeguards and curtailments respectively to the two sets will reduce the risk to the desired level. By ensuring that the permissions in $\rho(\Omega(P))$ are granted only after relevant predicates are verified and files in $\rho(\Omega(O))$ are cryptographically protected, the resulting risk level is reduced to:

$$\mathcal{R}'' = \sum_{t_\alpha \in T} \mathcal{T}(t_\alpha) \times \mathcal{V}''(t_\alpha) \times \mathcal{C}''(t_\alpha) \tag{12}$$

where the new vulnerability measure, based on Equation 4, is:

$$\mathcal{V}''(t_\alpha) = \sum_{p_\lambda \in (\hat{P}(t_\alpha) \cap \Omega(P) - \rho(\Omega(P)))} \frac{v(p_\lambda)}{|\hat{P}(t_\alpha)|} + \sum_{p_\lambda \in (\hat{P}(t_\alpha) \cap \Psi(P) \cup \rho(\Omega(P)))} \frac{v(p_\lambda) \times v'(p_\lambda)}{|\hat{P}(t_\alpha)|} \tag{13}$$

and the new consequence measure, based on Equation 3, is:

$$\mathcal{C}''(t_\alpha) = \sum_{o_\beta \in (A(t_\alpha) \cap \Omega(O) - \rho(\Omega(O)))} c(o_\beta) + i(o_\beta) + a(o_\beta) \tag{14}$$

Instead, after an event e occurs, if the risk level $\mathcal{R}_a$ decreases, the goal of the response engine is to allow the risk to rise by $\delta_g \leq \mathcal{R}_0 - \mathcal{R}_a$ to a level below the threshold of risk tolerance $\mathcal{R}_0$. To do this, it must select a subset of permissions $\rho(\Psi(P)) \subseteq \Psi(P)$ and a subset of objects $\rho(\Psi(O)) \subseteq \Psi(O)$, such that removing the safeguards and curtailments currently in use for these two sets will yield the maximum improvement to runtime performance. After the safeguards and curtailments are relaxed, the risk level will rise to:

$$\mathcal{R}'' = \sum_{t_\alpha \in T} \mathcal{T}(t_\alpha) \times \mathcal{V}''(t_\alpha) \times \mathcal{C}''(t_\alpha) \tag{15}$$

where the new vulnerability measure, based on Equation 4, is:

$$\mathcal{V}''(t_\alpha) = \sum_{p_\lambda \in \hat{P}(t_\alpha) \cap \Omega(P) \cup \rho(\Psi(P))} \frac{v(p_\lambda)}{|\hat{P}(t_\alpha)|} + \sum_{p_\lambda \in \hat{P}(t_\alpha) \cap \Psi(P) - \rho(\Psi(P))} \frac{v(p_\lambda) \times v'(p_\lambda)}{|\hat{P}(t_\alpha)|} \tag{16}$$

and the new consequence measure, based on Equation 3, is:

$$\mathcal{C}''(t_\alpha) = \sum_{o_\beta \in A(t_\alpha) \cap \Omega(O) \cup \rho(\Psi(O))} c(o_\beta) + i(o_\beta) + a(o_\beta) \tag{17}$$

There are $O(2^{(|P|+|O|)})$ ways of choosing subsets $\rho(\Omega(P)) \subseteq \Omega(P)$ and $\rho(\Omega(O)) \subseteq \Omega(O)$ for risk reduction or subsets $\rho(\Psi(P)) \subseteq \Psi(P)$ and $\rho(\Psi(O)) \subseteq \Psi(O)$ for risk relaxation. When selecting from the possibilities, the primary constraint is the maintenance of the bound $\mathcal{R}'' < \mathcal{R}_0$, where $\mathcal{R}'' = \mathcal{R}_a - \delta_g$ in the case of risk reduction, and $\mathcal{R}'' = \mathcal{R}_a + \delta_g$ in the case of risk relaxation.

The choice of safeguards and curtailments also impacts the performance of the system. Evaluating predicates prior to granting permissions introduces latency in system calls. Cryptographically protecting objects decreases usability. Hence, the choice of subsets $\rho(\Omega(P))$ and $\rho(\Omega(O))$ or subsets $\rho(\Psi(P))$ and $\rho(\Psi(O))$ is subject to the secondary goal of minimizing the overhead introduced.

The adverse impact of a safeguard or curtailment is proportional to the frequency with which it is utilized in the system's workload. Given a typical workload, we can count the frequency $f(p_\lambda)$ with which permission p_λ

is requested in the workload. Similarly, we can count the frequency $f(o_\beta)$ with which file o_β is accessed in the workload. This can be done for all permissions and files. The cost of utilizing subsets $\rho(\Omega(P))$ and $\rho(\Omega(O))$ for risk reduction can then be calculated with:

$$\zeta(\rho(\Omega(P)), \rho(\Omega(O))) = \sum_{p_\lambda \in \rho(\Omega(P))} f(p_\lambda) + \sum_{o_\beta \in \rho(\Omega(O))} f(o_\beta) \tag{18}$$

Similarly, if the safeguards of subset $\rho(\Psi(P))$ and the curtailments of consequences to assets in subset $\rho(\Psi(O))$ are relaxed, the resulting reduction in runtime cost can be calculated with:

$$\zeta(\rho(\Psi(P)), \rho(\Psi(O))) = \sum_{p_\lambda \in \rho(\Psi(P))} f(p_\lambda) + \sum_{o_\beta \in \rho(\Psi(O))} f(o_\beta) \tag{19}$$

The ideal choice of safeguards and curtailments will minimize the safeguards' and curtailments' impact on performance, while simultaneously ensuring that the risk remains below the threshold of tolerance. Thus, for risk reduction we wish to find:

$$\min \zeta(\rho(\Omega(P)), \rho(\Omega(O))), \quad \mathcal{R}'' \leq \mathcal{R}_0 \tag{20}$$

In the context of risk relaxation, we wish to find:

$$\max \zeta(\rho(\Psi(P)), \rho(\Psi(O))), \quad \mathcal{R}'' \leq \mathcal{R}_0 \tag{21}$$

2.6 Complexity

We note that the semantics of risk management require that at each step the risk must be reduced below the threshold of tolerance. This precludes optimization strategies such as minimizing a weighted sum of risk and runtime performance. We conclude that runtime risk management is a $0-1$ integer non-linear programming problem with a linear objective function and quadratic constraint. The decision problem that corresponds to the aforementioned optimization problem is NP-hard [Garey79] as argued below.

2.6.1 Optimization Problem

Risk reduction can viewed as selecting a set of vertices in a vertex-weighted, edge-weighted bipartite graph, such that the sum of the weights of the vertices selected is minimized, subject to the constraint that the sum of the weights of the edges present in the subgraph induced by the selected vertices is greater than a fixed threshold. The vertices in one partition correspond to the set of unsafeguarded permissions $\Omega(P)$, while the vertices in the other partition correspond to the set of objects whose access is uncurtailed $\Omega(O)$. The weight of each vertex $p_\lambda \in \Omega(P)$ is $f(p_\lambda)$, the frequency of the permission in the workload, while the weight of each vertex $o_\beta \in \Omega(O)$ is $f(o_\beta)$, the frequency of the object in the workload.

The weight of an edge (p_λ, o_β) between a permission p_λ and an object o_β is the contribution to the total risk that results from the exposure of the corresponding permission and the cost of the corresponding object's security being subverted. Thus the weight of the edge is:

$$w(p_\lambda, o_\beta) = \sum_{t_\alpha \in T \,:\, p_\lambda \in \hat{P}(t_\alpha) \cap \Omega(P) \;\wedge\; o_\beta \in A(t_\alpha) \cap \Omega(O)} \mathcal{T}(t_\alpha) \times \frac{v(p_\lambda) \times v'(p_\lambda)}{|\hat{P}(t_\alpha)|} \times c(o_\beta) + i(o_\beta) + a(o_\beta) \tag{22}$$

The fixed threshold is the risk tolerance, $\mathcal{R}_0$. Risk relaxation is similar, with the exception that the sum of weights of the chosen vertices (which are from the sets $\Psi(P)$ and $\Psi(O)$ instead of sets $\Omega(P)$ and $\Omega(O)$) must be *maximized*, while the sum of the weights of the edges in the induced subgraph must remain below a fixed threshold. The two optimization problems are equivalent.

2.6.2 Decision Problem

The decision problem for risk reduction takes as input: (i) a bipartite graph of the form described above, (ii) a fixed threshold which is the risk tolerance, and (iii) the sum of the weights of the subset of vertices to be chosen, which corresponds to the runtime cost of the primitives in a proposed response. The output is only `true` if the algorithm is able to find a subset of vertices whose weights add up to the specified total, while the sum of the weights of the edges in the induced subgraph is at least the specified threshold.

2.6.3 NP-Hard

Given an algorithm for the risk reduction optimization problem, the decision problem can be solved by checking if the target sum of vertices' weights is less than, equal or greater than the minimum cost output by the optimization algorithm.

Given a decision algorithm for risk reduction, we can solve the *maximum edge biclique problem* which takes a bipartite graph and a threshold as inputs and outputs whether the graph includes a biclique that is the size of the threshold or larger. The problem is known to be NP-complete [Peeters03].

To solve the maximum edge biclique problem, we repeatedly invoke the risk reduction decision algorithm. The number of invocations is bounded above by the size of the vertex set in the graph. The input is the bipartite graph (with all vertices and edges weighted 1), the threshold and a target number of vertices that ranges during invocations from 1 to the total number of vertices in the bipartite graph. The first time the output is `true`, we stop and output `true` for the maximum edge biclique problem.

Since the risk reduction decision algorithm output `true`, the sum of the weights of the vertices (which is equal to the number of vertices since all the weights were set to 1) is the least possible such that the sum of the weights of the edges (which is equal to the number of edges since the weights of all the edges were set to 1) was at least the threshold specified. The total number of edges in a biclique is the maximum possible for a subset of vertices of the biclique's size. Thus, the smallest subset of vertices that will contain a specified number of edges is a biclique.

If all the invocations of the risk reduction decision problem produced an output of `false`, then the output for the maximum edge biclique problem is also `false`. This completes the reduction. If the risk reduction decision problem were *tractable*, then the maximum edge biclique problem would also be tractable, but it is known to be NP-complete. Therefore, the risk reduction (and risk relaxation since it is analogous) decision problems are NP-hard.

2.7 Response Selection

Determining the optimal choice of safeguards and curtailments for risk management corresponds to an NP-hard problem, as argued in Section 2.6. Additionally, there is evidence that the maximum edge biclique problem is difficult to approximate [Kogan04]. Since the choice is to be made in real-time, we will use a heuristic which guarantees that the risk threshold is maintained. The heuristic uses the greedy strategy of picking the response primitive with the highest benefit-to-cost ratio repeatedly till the constraint is satisfied. By maintaining the choices in a *heap* data structure keyed on the benefit-to-cost ratio, the first primitive in the response set can be chosen in $O(1)$ time. This is significant since implementing a single response primitive is often sufficient for disrupting an attack in progress.

Since the benefit associated with each unutilized safeguard or curtailment is the degree to which the risk will be reduced if it is used, this is a function of other safeguards or curtailments related to the threats that it affects. Similarly, since the loss of benefit associated with each currently utilized safeguard or curtailment is the degree to which the risk will increase if it is used, this is also a function of the other safeguards or curtailments associated with the threats that it affects. As a result, the benefit of adding or removing each safeguard or curtailment must be recalculated each time other safeguards or curtailments are added or removed.

2.7.1 Risk Reduction

We outline the algorithm for the case where the risk needs to be reduced. The first two steps constitute pre-processing and therefore only occur during system initialization.

Step 1 The benefit-to-cost ratio of each candidate safeguard permission $p_\lambda \in \Omega(P)$ can be calculated by:

$$\kappa(p_\lambda) = \frac{\sum\limits_{t_\alpha : p_\lambda \in (\hat{P}(t_\alpha) \cap \Omega(P))} \mathcal{T}(t_\alpha) \times \frac{v(p_\lambda) \times (1 - v'(p_\lambda))}{|\hat{P}(t_\alpha)|} \times \mathcal{C}'(t_\alpha)}{f(p_\lambda)} \tag{23}$$

Step 2 Similarly, the benefit-to-cost ratio of protecting an object $o_\beta \in \Omega(O)$ can be calculated by:

$$\kappa(o_\beta) = \frac{(c(o_\beta) + i(o_\beta) + a(o_\beta)) \quad \times \sum\limits_{t_\alpha : o_\beta \in (A(t_\alpha) \cap \Omega(O))} \mathcal{T}(t_\alpha) \times \mathcal{V}'(t_\alpha)}{f(o_\beta)} \tag{24}$$

Step 3 The response sets are defined as empty, that is $\rho(\Omega(P)) = \rho(\Omega(O)) = \phi$.

Step 4 The single risk reducing measure with the highest benefit-to-cost can be selected, that is:

$$\begin{aligned} &\max\ p_{max}, o_{max} \quad where: \\ &p_{max} = \max\ \kappa(p_\lambda), \quad p_\lambda \in \Omega(P) \\ &o_{max} = \max\ \kappa(o_\beta), \quad o_\beta \in \Omega(O) \end{aligned} \tag{25}$$

If it is a permission it is added to $\rho(\Omega(P))$ and if it is an object it can be added to $\rho(\Omega(O))$.

Step 5 If the choice was a permission p_λ, then the value $\kappa(o_\beta)$ must be recalculated for all objects o_β that are affected by threats which utilize p_λ in the course of their attacks. Thus, each $\kappa(o_\beta)$ must be updated if:

$$o_\beta \in \bigcup_{t_\alpha : p_\lambda \in \hat{P}(t_\alpha)} A(t_\alpha) \tag{26}$$

Instead, if the choice was an object o_β, then the value $\kappa(p_\lambda)$ must be recalculated for all permissions p_λ that are utilized in the course of an attack that affects object o_β. Thus, each $\kappa(p_\lambda)$ must be updated if:

$$p_\lambda \in \bigcup_{t_\alpha : o_\beta \in A(t_\alpha)} \hat{P}(t_\alpha) \tag{27}$$

Step 6 The risk before the candidate responses were utilized is $\mathcal{R}_a$. If the responses were activated the resulting risk $\mathcal{R}''$ is given by:

$$\mathcal{R}'' = \mathcal{R}_a \quad - \sum_{p_\lambda \in \rho(\Omega(P))} \kappa(p_\lambda) \times f(p_\lambda) \quad - \sum_{o_\beta \in \rho(\Omega(O))} \kappa(o_\beta) \times f(o_\beta) \tag{28}$$

This is equivalent to using Equations 12, 13 and 14. While the worst case complexity is the same, when few protective measures are added the cost of the above calculation is significantly lower.

Step 7 If $\mathcal{R}'' > \mathcal{R}_0$ then the system repeats the above from Step 4 onwards. However, if $\mathcal{R}'' \leq \mathcal{R}_0$ then the set of safeguards $\rho(\Omega(P))$ and the set consequence curtailing measures $\rho(\Omega(O))$ must be applied. $\rho(\Omega(P))$ should be transferred from $\Omega(P)$ to $\Psi(P)$ and $\rho(\Omega(O))$ should be transferred from $\Omega(O)$ to $\Psi(O)$. Then the response sets should be reset so $\rho(\Omega(P)) = \rho(\Omega(O)) = \phi$.

The time complexity is:

$$O((\rho(\Omega(P)) + \rho(\Omega(O))) \times (\log|P| + \log|O| + \sum_{t_\alpha \in T}(|\hat{P}(t_\alpha)| + |A(t_\alpha)|))) \tag{29}$$

In the worst case, this is $O(|P| + |O|)^2$. Unless a large variety of attacks are simultaneously launched against the target, the first factor will remain small. Additionally, if there is a strong correlation between the exposures and the consequences, then the second factor will also remain small. Thus, in practice it is likely to achieve acceptable results.

2.7.2 Risk Relaxation

In the case of risk relaxation, the algorithm becomes:

Step 1 For $p_\lambda \in \Psi(P)$ calculate:

$$\kappa(p_\lambda) = \frac{\sum_{t_\alpha : p_\lambda \in (\hat{P}(t_\alpha) \cap \Psi(P))} \mathcal{T}(t_\alpha) \times \frac{v(p_\lambda) \times (1 - v'(p_\lambda))}{|\hat{P}(t_\alpha)|} \times \mathcal{C}'(t_\alpha)}{f(p_\lambda)} \tag{30}$$

Step 2 For $o_\beta \in \Psi(O)$ calculate:

$$\kappa(o_\beta) = \frac{(c(o_\beta) + i(o_\beta) + a(o_\beta)) \quad \times \sum_{t_\alpha : o_\beta \in (A(t_\alpha) \cap \Psi(O))} \mathcal{T}(t_\alpha) \times \mathcal{V}'(t_\alpha)}{f(o_\beta)} \tag{31}$$

Step 3 Set $\rho(\Psi(P)) = \rho(\Psi(O)) = \phi$.

Step 4 Find the safeguard or curtailment which yields the least risk reduction per instance of use:

$$\begin{aligned} &\min\ p_{min}, o_{min} \quad where: \\ &p_{min} = \min\ \kappa(p_\lambda), \quad p_\lambda \in \Psi(P) \\ &o_{min} = \min\ \kappa(o_\beta), \quad o_\beta \in \Psi(O) \end{aligned} \tag{32}$$

Add it to $\rho(\Psi(P))$ if it is a permission. Instead, add it to $\rho(\Psi(O))$ if it is a file.

Step 5 Update $\kappa(o_\beta)$ if:

$$o_\beta \in \bigcup_{t_\alpha : p_\lambda \in \hat{P}(t_\alpha)} A(t_\alpha) \tag{33}$$

or update $\kappa(p_\lambda)$ if:

$$p_\lambda \in \bigcup_{t_\alpha : o_\beta \in A(t_\alpha)} \hat{P}(t_\alpha) \tag{34}$$

depending on whether a permission or a file was chosen in the previous step.

Step 6 Calculate $\mathcal{R}''$:

$$\mathcal{R}'' = \mathcal{R}_a \quad + \sum_{p_\lambda \in \rho(\Psi(P))} \kappa(p_\lambda) \times f(p_\lambda) \quad + \sum_{o_\beta \in \rho(\Psi(O))} \kappa(o_\beta) \times f(o_\beta) \tag{35}$$

Step 7 If $\mathcal{R}'' < \mathcal{R}_0$, repeat from Step 4. If $\mathcal{R}'' = \mathcal{R}_0$, proceed to next step. If $\mathcal{R}'' > \mathcal{R}_0$, undo last iteration of Step 4.

Step 8 Relax all measures in $\rho(\Psi(P))$ and $\rho(\Psi(O))$ and transfer them to $\Omega(P)$ and $\Omega(O)$, respectively. Set $\rho(\Psi(P)) = \rho(\Psi(O)) = \phi$.

3 Related Work

3.1 Intrusion Response

Frameworks have previously been proposed for adding response capabilities. DCA [Fisch96] introduced a taxonomy for response and a tool to demonstrate the utility of the taxonomy. EMERALD's [Porras97] design allows customized responses to be invoked automatically, but does not define them by default. AAIR [Carver01] describes an expert system for response based on an extended taxonomy.

Our approach creates a framework for systematically choosing a response in real-time. This allows an attack to be contained automatically instead of being limited to raising an alarm, and does not require a new response subsystem to be developed for each new class of attack discovered.

3.2 Risk Management

Risk analysis has been utilized to manage the security of systems for several decades [FIPS31]. However, its use has been limited to offline risk computation and manual response. [SooHoo02] proposes a general model using decision analysis to estimate computer security risk and automatically update input estimates. [Bilar03] uses reliability modeling to analyze the risk of a distributed system. Risk is calculated as a function of the probability of faults being present in the system's constituent components. Risk management is framed as an integer linear programming problem, aiming to find an alternate system configuration, subject to constraints such as acceptable risk level and maximum cost for reconfiguration.

In contrast to previous approaches, we use the risk computation to drive changes in the operating system's security mechanisms. This allows risk management to occur in real-time and reduces the window of exposure.

The model described has been realized in a prototype, as described in [Gehani03]. It includes the following: an implementation of a state transition [Ilgun95] based intrusion detector; an extension of the Java access control subsystem to support predicated permissions; the ability to specify the exposure reduction resulting from each predicate's evaluation; a modification of the Java file input/output facilities to allow explicit guarantees about the confidentiality, integrity and availability of the data; the ability to explicitly associate costs for the loss of each of the security characteristics of any file; a risk manager that coordinates the other subsystems. The modified platform was able to respond and contain a suite of synthetic attacks against a web server running on it. This was expected since the configuration had been manually tuned. Further work is needed to automate the process of configuring the system.

4 Conclusion

This paper addresses the issue of modeling the detection and response process by equating the insecurity of a system with the risk it faces. The risk is defined formally in terms of known threats, exposures and consequences. Each threat is characterized using a host-based intrusion detection signature. Its likelihood is estimated by the current extent of the signature's match when a system is running. The system's exposure to a threat is a function of the permissions requested in the course of the attack and the auxiliary checks being performed before they are granted. The consequence of a threat is calculated as a function of the data that can lose confidentiality, integrity or availability when the attack is successful. The products of each threat's likelihood, the system's exposure to it and its consequence, contribute additively to form the total known risk of the host.

Two types of responses are utilized. One type performs further runtime checks before granting specific permissions, while the other allows disabling of transparent decryption, signing of modifications and remote replication of specific protected files. Intrusion response is framed as maintaining the risk below a threshold of tolerance with a simultaneous goal of minimizing the impact of the selected responses on runtime performance. This requires

minimizing a particular linear objective function with a quadratic constraint, which we have shown is NP-hard. Given that the response must be chosen in real-time, an optimal choice is ruled out. Instead, a greedy heuristic is used. It allows the first response primitive to be chosen in constant time, which is frequently sufficient to disrupt an attack. The remaining primitives, required to adequately manage the risk, are selected in quadratic time.

References

[Bilar03] Daniel Bilar, Quantitative Risk Analysis of Computer Networks, PhD thesis, Dartmouth College, 2003.

[Carver01] Curtis Carver, Adaptive, Agent-based Intrusion Response, PhD thesis, Texas A and M University, 2001.

[CSRMMBW88] Proceedings of the 1st Computer Security Risk Management Model Builders Workshop, Martin Marietta, Denver, Colorado, National Bureau of Standards, May 1988.

[CSRMMBW89] Proceedings of the 2nd Computer Security Risk Management Model Builders Workshop, AIT Corporation, Ottawa, Canada, National Institute of Standards and Technology, June 1989.

[CSRMMBW90] Proceedings of the 3rd International Computer Security Risk Management Model Builders Workshop, Los Alamos National Laboratory, Santa Fe, New Mexico, National Institute of Standards and Technology, August 1990.

[CSRMMBW91] Proceedings of the 4th International Computer Security Risk Management Model Builders Workshop, University of Maryland, College Park, Maryland, National Institute of Standards and Technology, August 1991.

[FIPS31] Guidelines for Automatic Data Processing Physical Security and Risk Management, National Bureau of Standards, 1974.

[FIPS65] Guidelines for Automatic Data Processing Risk Analysis, National Bureau of Standards, 1979.

[Fisch96] Eric Fisch, Intrusive Damage Control and Assessment Techniques, PhD thesis, Texas A and M University, 1996.

[Fletcher95] Sharon Fletcher et al, Software System Risk Management and Assurance, Proceedings of the New Security Paradigms Workshop, August 1995.

[Garey79] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.

[Gehani03] Ashish Gehani, Support for Automated Passive Host-based Intrusion Response, PhD thesis, Duke University, 2003.

[Harrison76] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman, Protection in operating systems, Communications of the ACM, 19(8):461 471, August 1976.

[NIST800-12] Guidelines for Automatic Data Processing Physical Security and Risk Management, National Institute of Standards and Technology, 1996.

[NIST91] Description of Automated Risk Management Packages That NIST/NCSC Risk Management Research Laboratory Has Examined, National Institute of Standards and Technology, 1991.

[Peeters03] Rene Peeters, The maximum edge biclique problem is NP-complete, Discrete Applied Mathematics, Volume 131, Issue 3, p651-654, 2003.

[Ilgun95] Koral Ilgun, Richard A. Kemmerer and Phillip A. Porras, State Transition Analysis: A Rule-Based Intrusion Detection Approach, IEEE Transactions on Software Engineering, 21(3), pp. 181-199, March 1995.

[Kogan04] Uriel Feige and Shimon Kogan, Hardness of Approximation of the Balanced Complete Bipartite Subgraph Problem, Technical Report MCS04-04, Department of Computer Science and Applied Math, Weizmann Institute of Science, May 2004.

[Porras97] P.A. Porras and P.G. Neumann, EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances, Proceedings of the Nineteenth National Computer Security Conference, p353-365, Baltimore, Maryland, 22-25 October 1997.

[SooHoo02] Kevin Soo Hoo, Guidelines for Automatic Data Processing Physical Security and Risk Management, PhD Thesis, Stanford University, 2002.