I. J. Computer Network and Information Security, 2013, 11, 58-68
Published Online September 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2013.11.08

Context-Sensitive Access Control Policy Evaluation and Enforcement Using Vulnerability Exploitation Data

Hassan Rasheed
Deanship of Information Technology, Taif, University Taif, Saudi Arabia
hsrasheed@acm.org

Abstract—Conventional approaches for adapting security enforcement in the face of attacks rely on administrators to make policy changes that will limit damage to the system. Paradigm shifts in the capabilities of attack tools demand supplementary strategies that can also adjust policy enforcement dynamically. We extend the current research by proposing an approach for integrating real-time security assessment data into access control systems. Critical application scenarios are tested to examine the impact of using risk data in policy evaluation and enforcement.

Index Terms—Context Awareness, Adaptive Access Control, Vulnerability Assessment

I. INTRODUCTION

Many of the measures used to achieve or maintain system security require the intervention of a human administrator who adjusts the system to respond to changing conditions including intrusions and attacks. While the oversight of a human administrator will likely never be dispensed with completely, ongoing trends regarding the speed and dynamism of attacks have continued to reduce the degree of response and containment that administrators can offer before attacks cause significant damage. The changing nature of attacks was noted in [1] and has since been confirmed in various other reports such as [2]. Amongst the factors noted in the initial report were the following: 1) increasing automation and speed of attack tools, 2) increasing sophistication of attack tools and 3) faster discovery of vulnerabilities.

The first factor implies that each of the four common phases of automated attacks (scanning, compromising, propagating and coordinated management) are being done more quickly and effectively. Attack tools use exploits in the midst of scanning and automatically initiate attack cycles. As a result, the window of response before an attack moves on to the next stage is no longer based on the response time of a human attacker and can, therefore, easily outpace a human administrator's ability to respond.

The second factor indicates that attackers increasingly use techniques to conceal the nature of the tools they use. In addition, the tools themselves are more modular and exhibit more dynamic behavior. This leads us to the notion that security mechanisms must be able to consider multiple factors when assessing the intrusiveness of a given event.

The third factor was supported with the analysis that the number of new vulnerabilities reported more than doubles each year, often due to examination of existing code for newly discovered vulnerability classes. This implies a wider number of available attack vectors at any given point in time. It also implies that the potential for publicizing vulnerabilities will create more occurrences of widespread exploitation of the same vulnerability. Although, in its 2007 annual report IBM's Internet Security Systems (ISS) group reported that vulnerabilities in 2007 were down five percent compared to 2006. The number of those vulnerabilities that were classified as severe (high impact) rose by 28 percent [2]. The report also noted that of all of the vulnerabilities newly discovered in 2007, that only 50 percent of them are correctable with a vendor patch, meaning that the need for detecting vulnerability exploitation as a indicator of attack will remain critical.

The current paradigm for responding to these threats, however, still largely revolves around many of the same techniques: assessing probable threats to systems and preemptively enacting security measures, or monitoring the state of the system through log files or using intrusion detection systems and manually adjusting security policies to mitigate or respond to intrusions based on reports from analysis mechanisms.

The nature of the attacks as discussed earlier, demands more dynamic responses. With attacks progressing more rapidly and in a more sophisticated fashion, responses by human administrators must be augmented by responses that can be triggered based on changing system conditions. It is necessary, therefore, to confront both the need for dynamic responses and the lack of models that facilitate the evaluation of security metrics in real-time. For these two key reasons, we propose a model for dynamically assessing the risk posed by incoming access requests and a framework for triggering responses based on risk data. In addition, two architectures for integrating risk assessment into access control systems are proposed and evaluated. The

remainder of this paper is organized as follows: first will be a discussion of related work. Next will be a detailed discussion of the proposed method including: the model for risk analysis, the architecture, system implementation and various testing results. Finally, we will offer some conclusions based on the results obtained and conclude with future work. For issues of brevity, a detailed performance analysis of the framework will be the subject of a subsequent paper.

II. RELATED WORK

Previous work in four main areas is directly related to the topic under discussion. Work in vulnerability assessment has produced standards for the objective measurement of vulnerability magnitudes which is critical to assessing risk based on evidence of vulnerability exploitation. Efforts in the use of threat assessment have demonstrated concrete approaches for the inclusion of dynamic intrusion-based assessments into the performance of access control. Previous work under the heading of risk metrics has, utilizing the standards for vulnerability assessment, provided structured approaches for attributing risk to system entities based on a historical relationship with published vulnerabilities. Work in intrusion response has explored, categorized and outlined techniques for responding to ongoing attacks; this work provides a library of techniques that can be triggered based on risk data.

A. Vulnerability Assessment

The Common Vulnerability Scoring System [3] is an open standard for describing the impact of system vulnerabilities. It includes three broad categories of metrics: base metrics which are inherent to the vulnerability, temporal metrics which measure aspects of the vulnerability that change over time and environmental metrics that measure aspects of the vulnerability that are specific to a particular environment. The group of base metrics includes the following properties: access vector, access complexity, authentication, confidentiality impact, integrity impact and availability impact. Each property has three possible discrete values and the values for all of the properties are used together to calculate a base score for the vulnerability.

B. Risk Metrics

The authors in [4] develop a metric for assessing the overall security of a network by assessing the accessible services on the network. For each service, its historical record of vulnerabilities and the frequency of those vulnerabilities is used to estimate a probability that new vulnerabilities will be discovered and hence present an opportunity for would-be attacks. The security policies employed on the network are also examined to assess the potential for attack propagation. This approach serves as a significant foundation for the risk model which is subsequently proposed.

In [5] a number of methods for vulnerability assessment are used to improve the accuracy of an intrusion prevention system. A vulnerability scanner is used to monitor system services and keep an updated record of their current vulnerabilities - this information is used to filter out alerts for vulnerability alerts that do not apply to the current version of the service. The alerts themselves are also filtered by only considering alerts linked to vulnerabilities with references in the Common Vulnerability and Exposures (CVE) database [6].

C. Threat Assessment

The approach to integrated security used by Ryutov et al. [7, 8] is based the notion of an advanced security policy that can specify allowed activities, detect abuse and respond to intrusions. Each of these tasks (access control, intrusion detection and intrusion response) is performed by a single, multi-phase policy evaluator. A global 'System Threat Level' is used to integrate information from outside intrusion detection systems.

Teo et al. [9] propose a system to manage network level system access that considers threat information. Each node and service in the system has an associated access threshold. This threshold is checked against the threat level of a part requesting access to determine if access is granted. The threat level of a source is regulated (increased and decreased) when signatures are triggered that specify the type of action to match and the type of threat level adjustment that should be performed.

In [10] the authors use a framework to assess the risk associated with granting a given access request and a corresponding level of trust required by any subject seeking to execute the request. In parallel, the trust level of the actual requesting subject is calculated and compared with the established value for the request to form a decision for the request.

D. Intrusion Response

In [11] a taxonomy of intrusion response systems is offered that classifies systems based on multiple factors. One of the factors is their method for selecting which response is used in a given situation. Methods for response selection are divided into three: those that map attacks statically, those that do so dynamically based on some parameters and those that use a calculation of the relative cost of the intrusion with the cost of the response. Our approach to response selection is roughly within the third category. The author of the access control policy is responsible for deciding which security risk factors (i.e. global system risk, risk from the requesting source or risk to the target) will be used during policy evaluation. A threshold is then set to designate the acceptable limit for the risk parameter before the request is blocked.

In [12] a data mining approach to log file analysis is used in order to maintain a list of IP addresses which are banned from the access control system being protected. This type of data-influenced intrusion response is similar to the approach being presented, with the addition that the current approach aggregates risk data along multiple

dimensions (source, target and action) and thus provides more possibilities for response.

III. ADAPTIVE ASSESSMENT BASED ACCESS CONTROL

A. Overview

The proposed solution for achieving attack-resistant access control is the use of real-time assessment data in access control policy evaluation and enforcement. Specifically, evidences of vulnerability exploitation are collected and analyzed into a higher level risk assessment for the sources and targets of access control requests. This risk assessment is subsequently used as an additional parameter or contextual property in access control policies so that permit and deny decisions for an incoming request are based on an assessment of the risk posed by the requesting source and/or the risk posed to the targeted resource. This approach has been termed the Adaptive Assessment-Based Access Control System (ABACUS). The underlying methodology for this approach is that adaptive security mechanisms must essentially rely on three interrelated processes: data acquisition, data analysis and data application.

Because such a system is heavily dependent on data, the quality of such data is an important issue. This approach assumes the presence of data imperfections and has two strategies for dealing with such imperfections. In anticipation of situations where there is a lack of data (essentially missed detections), the approach relies on a best-effort estimation strategy so that if some intrusive behavior is missed the preponderance of data on a particular entity will still be enough to indicate where the greatest risk lies. The idea is that if we can detect enough of the intrusive activity we can still limit damage to the system, even without detecting all of it. During instances in which there is inaccurate data (essentially false detections) we employ two different filtering techniques to reduce the impact of inaccurate data on the overall risk assessment.

B. Analysis Model

The purpose of the analysis model is to produce a risk assessment for specific system entities based on data from a detection sensor. In this case, the data are descriptions of attempts to exploit software vulnerabilities. There are several difficulties that preclude making access control decisions based on an assessment regarding the intrusiveness of the request itself. The vast majority of the sensors for detecting intrusive behavior do not function with the level of accuracy necessary to dependably base access control decisions on their output. Therefore, we moved to the challenge of simply making 'better' access control decisions using data regarding system state in general and from intrusion detection sensors in particular. The approach is twofold: 1) aggregate incoming data on the sources and targets of suspicious events and 2) instead of attempting to assess new requests directly regarding their intrusiveness, we instead assess the request using previously derived data on the source of the request and the target of the request. Additional details regarding the analysis model are offered in [13].

Estimating Risk for Events/Requests

Risk is associated with a probable intrusion attempt, evidenced by an attempt to exploit a vulnerability the system. The risk posed by a request, therefore, is proportional to the severity of the vulnerabilities it is suspected to be seeking to exploit.

The CVSS standard provides a widely accepted, quantitative measurement scale for the severity of vulnerabilities, and therefore we will leverage that standard for the rating of vulnerabilities. The overall method for providing a single vulnerability estimate based on multiple vulnerabilities spread out over time is derived from the method used in [4]. The method has been adapted, however, to take as input a set of vulnerabilities associated with a request, instead of the set of vulnerabilities that apply to a particular service. The function $R(r_j)$ given as Equation 1 provides an estimation of the total risk for a request r_j by taking the exponential average of all of the vulnerability descriptions associated with that request. The exponential average was chosen, as noted in [4], to provide a risk estimate for the request that is at least as large as the highest severity vulnerability associated with the request. The risk magnitude assigned to a vulnerability exploitation attempt is the exponential average of all of the magnitudes of all of the vulnerabilities referenced in the alert.

$$R(r_j) = \ln\left(\sum_{v_k \in V(r_j)} e^{(w_x * SS(v_k))}\right) \quad (1)$$

The set $V(r_j)$, with members v_k is the set of all vulnerability exploitation signatures triggered by the request r_j. $SS(v_k)$ is the magnitude of the vulnerability v_k and w_x is a weighting function that allows us to optionally amplify the impact of high-severity alerts.

Assigning Risk to Entities

The algorithm for calculating risk based on multiple events is a recursive formulation that allows the framework to efficiently maintain an accurate risk assessment for all of the entities interacting with, or being accessed by the access control system. We define two functions, one for the risk of a targeted resource and the other for the source of requests. The function $TR(t_i, r_{t+1})$, defined as Equation 2 is the risk assessed to the target t_i as a result of intrusive request r_{t+1}. $SR(s_i, r_{t+1})$, defined as Equation 3 is the risk assessed to the source s_i as a result of intrusive request r_{t+1}

$$TR(t_i, r_{t+1}) = \ln\left(\varepsilon * e^{TR(t_i, r_t)} + R(r_{t+1})\right) \quad (2)$$

$$SR(s_i, r_{t+1}) = \ln\left(\varepsilon * e^{SR(t_i, r_t)} + R(r_{t+1})\right) \quad (3)$$

We again take a weighted exponential average of the previous risk assessment for that entity with the risk implied by the most recent event. The ε value, where

$0 < \varepsilon < 1$ serves to weight the previous risk assessment for the entity with respect to the risk assessment for the newest event. This serves the function of decreasing the influence of older data in favor of newer data, but does so as triggered by new events, and not merely a uniform time dependency. This accommodates better assessing risk to entities with vastly different request frequencies.

C. Intrusion Response and Attack Resistance

Strategy Selection

The strategies put forth in the literature for responding to intrusions and attempted system attacks are numerous and varied. Therefore, it is necessary to select only those that most closely match the requirements for achieving the desired goal: namely, attack resistant access control. The first restriction is that the responses applied should serve to manipulate some element in the access control domain. Access control is primarily concerned with a set of subjects, a set of objects and the specific operations that each subject can perform on each object. So our response technique must manipulate these permissions, either at the subject side (by designating which actions a subject can perform) or at the object side (designating what can be done with the object). The second requirement is that the strategy or response can be triggered using risk data.

A number of different intrusion responses are detailed in [14, 15]. Using the criteria just discussed, however, the following three strategies were selected as appropriate for this application: 1) forcing (additional) authentication, 2) restricting subject permissions 3) restricting object permissions.

Response Triggering

The next aspect of the strategy to detail is the activation of the selected response techniques: based on what criteria will they be enacted and how will those criteria be described. Our approach to response selection is roughly within the third category of the intrusion response taxonomy mentioned in - cost-sensitive response selection. The author of the access control policy is responsible for deciding which security risk factors (i.e. global system risk, risk from the requesting source or risk to the target) will be used during the process of evaluating whether or not a request will be permitted. These individual measures are therefore the inputs into the response selection process. Each risk factor is then matched with a threshold that determines when the action associated with the factors should be performed.

Although the cost determination equations for response selection are highly system dependent, the risk progression in Table 1 is provided as an example and has been tested using the model discussed previously. For this specific progression, the attacker executes exploitation attempts of mid-severity every 60 seconds. The risk progression would change if any of the variables such as the risk rating of the individual requests, the inter arrival time between requests, or the weighting of the low, medium and high level risk events were adjusted.

Table 1: Sample Risk Progression

Request Number	Risk Estimation	Number of Previous Requests
1	0	0
2	25.65	1
3	32.19	2
4	36.11	3
5	38.92	4
6	41.11	5
7	42.91	6
8	44.43	7
9	45.75	8
10	46.92	9

Using the example risk progression, the following sample conditions are provided for performing each of the previously mentioned intrusion responses:

1. if Source_Risk >= 36.11 OR System_Risk >= 53.8 THEN Force_Authentication
2. if Target_Risk >= 41.11 THEN Restrict_Permission_X_On_Object
3. if Source_Risk >= 41.11 THEN Restrict_Permission_X_For_Subject

The first condition forces authentication for the subject if the risk generated by the subject exceeds 36.11 (roughly three exploitation attempts of mid-severity) or if the overall system level threat exceeds 53.8 (fifteen exploitation attempts). The second condition denies the subject from performing action X on the object if the target risk has risen at or above 41.11 (meaning it has received 5 or more exploitation attempts). The last condition denies the subject from performing action X on any objects if the source risk is at or above 41.11 (meaning that 5 or more exploitation attempts have been attributed to that subject).

IV. IMPLEMENTATION

A. Overview

A method is proposed for real-time assessment of the risk associated with the source and the target of access requests based on past evidences of vulnerability exploitation. Each request which triggers a vulnerability exploitation is assigned a risk magnitude based on the severity of the threat. By aggregating the risk assessments for all of the requests initiated by a source or directed to a target, we arrive at a risk assessment for that entity. We rely on filtering alerts for exploitation of concrete vulnerabilities in addition to configuration verification, to reduce false positives and increase the accuracy of the risk estimation.

B. Architecture

The primary components of the framework architecture are an alert server, which receives and processes assessment information, an analysis server, which responds to requests for analysis data, and the actual access control mechanisms which performs policy evaluation and enforcement. The access control system integrated with this architecture is the Apache webserver. The webserver is extended to perform the three intrusion responses discussed previously as the means to attack resistance: forcing additional authentication, restricting user permissions and restricting access to a target. Based on the resource and the actions available on that resource, a threshold is determined for the source and target associated risk above which, requests are denied. The intrusion detection system listens on the link for incoming requests and reports alerts for any requests that seem intrusive (in this case specifically, those requests that appear to be an attempt to exploit a known software vulnerability). The raw alerts from the IDS are passed through the alert processing server that performs any required filtering and also updates the risk assessments for the appropriate entities. Finally, the data from the new events is stored in an event database.

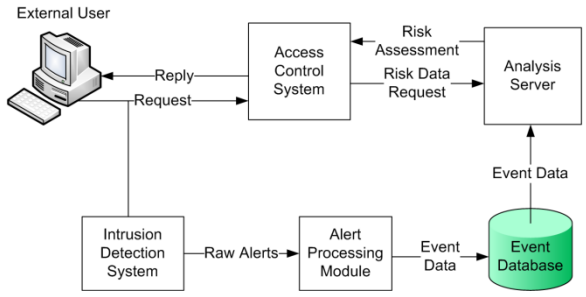

Figure 1: Proposed ABACUS System Architecture

The architecture is shown in Fig. 1.

Alert Processing Server

The alert processing module is responsible for extracting the information for each of the tables mentioned previously from the alerts it receives. In addition it can perform the functions of filtering out alerts that do not reference concrete vulnerabilities, or alerts for which the vulnerability does not match the current system configuration. Because of the nature of the analysis model, many of the most critical analysis functions are actually performed by the alert server. The present analysis model requires that the primary analysis function (updating risk values for entities) occurs as the events are processed (and consequently must be performed by the alert server and not by another entity).

Analysis Server

The analysis server receives client requests for assessment data, extracts the appropriate information from the event database and sends a response to the client (in this case the webserver).

Event Database

The event database is backed by a relational database implementation (in this case MySQL). Some of the structure of this database was derived from the IDMEF schema [16]. Some of the tables contained in the event database are the following:

• CVSS Vulnerabilities - this table stores information regarding current vulnerabilities from the National Vulnerability Database (NVD), which has adopted the CVSS scoring system. Each vulnerability is listed with its CVSS base score, exploit subscore, impact subscore, overall score and vector.

• Network Access Requests - Entries in this table are generated on the receipt of an IDS alert by the alert processing engine. The IP address and port of the source node are listed with the IP address and port of the target node. The time of the request, action being performed and target entity are also included in this table.

• Entity Tables - individual tables for the Nodes, Ports, Files and Users references in requests

• Intrusion Assessments - this table links individual requests to an intrusion assessment. Each assessment provides a classification for the event, its severity (which may be provided by the intrusion detection sensor) and whether or not the attack completed successfully.

• Vulnerability Descriptions - a vulnerability description provides information on a concrete software vulnerability. Each vulnerability description is provided by a vulnerability database (for the purposes of this study we only use cve vulnerabilities because they have an objective scoring system). Each vulnerability description, therefore, only links to one element in the table of cvss vulnerabilities and, consequently, only has one base score.

• Request Risk Cache - this table stores a calculated risk value for each request ID by querying for the cvss score for all of the vulnerability descriptions that are linked to an intrusion assessment (and which provide a CVE ID). As mentioned in the section describing the model, the exponential average of all of the cvss scores for the vulnerability descriptions used in a particular intrusion assessment are taken, and this value is stored in the request risk cache. When a particular risk handler queries the risk cache to produce a risk evaluation for a particular entity, the risk estimate is multiplied by the decay factor to produce a dynamic risk estimate for that particular request.

C. Access Control System

The access control system used with the second approach was the Apache webserver. In order to make as few modifications as possible to its existing access control policy evaluation mechanism, the ability to make and specify custom access control handlers for certain resources was utilized. Rather than returning a value for a specific attribute and querying against the event database within the access control handlers, the querying and analysis functions were abstracted into an external analysis server that provides risk analysis as a service. Requesting access control systems (such as the Apache webserver implementation) submit requests to the

analysis server specifying the type of desired risk analysis (source, target or system) and the attributes of the entity which the analysis should center around (in the case of the source and target analyses). Based on the risk assessment returned and the risk threshold that is assigned to that particular resource or action a permit or deny decision is returned.

Source Restriction Implementation:

An excerpt from the httpd.conf file for the webserver is shown in Fig. 2.

```
<Location /s>
PerlAccessHandler SourcePermissionRestrict
PerlSetVar Action_A1_RiskThreshold 26
PerlSetVar Action_A2_RiskThreshold 32
PerlSetVar Action_A3_RiskThreshold 36
PerlSetVar Action_A4_RiskThreshold 39
    PerlSetVar SourceLockoutThreshold 41
</Location>
```

Figure 2: Example Policy for Source Permission Restriction

This directive establishes the module "SourcePermissionRestrict" as an access control handler. This module implements the attack response of restricting source permission. In this particular example five different levels of granularity are established. Action "A1" is the least tolerant of risk: a threshold of 26 is set for the source risk, above which, requests will be denied. The other actions are progressively more risk-tolerant. The final threshold "SourceLockoutThreshold" establishes that a source will be blocked from all actions on all objects when its source risk level exceeds 41. The corresponding pseudo code for the handler is shown in Fig. 3.

The processing steps for the source restriction and target restriction handlers are relatively the same, summarized in the following steps:

1. The properties of the request (subject and object of the request and the action being performed) are extracted from the URL and the request properties.

```
read threshold_values;
read request_properties;
request source_risk from analysis server;
set response = OK;
if(source_risk > lockout_threshold)
{ response = DENY_REQUEST; }
else if(request_action == A1
AND source_risk > A1_threshold)
{ response = DENY_REQUEST; }
else if(request_action == A2
AND source_risk > A2_threshold)
{ response = DENY_REQUEST; }
else if(request_action == A3
AND source_risk > A3_threshold)
{ response = DENY_REQUEST; }
else if(request_action == A4
AND source_risk > A4_threshold)
{ response = DENY_REQUEST; }
return response;
```

Figure. 3: Pseudocode for Source Restriction Module

2. A request to the risk analysis server is generated specifying a) which type of analysis data is required and b) the identifier for the subject or object of the request

3. Once the risk value is returned, it is compared with the threshold(s) specified in the configuration file to determine if the request should be denied.

4. If none of the thresholds are violated, the request is permitted.

Force Authentication Implementation

The policy configuration for the access control module to force authentication is shown in Fig. 4.

```
<Location /sc3>
SetHandler perl-script PerlHandler AuthChain
PerlSetVar SystemRiskThreshold 55
PerlSetVar SourceRiskThreshold 33
PerlSetVar TargetRiskThreshold 45
PerlSetVar AuthExpiration 300000
</Location>
```

Figure 4: Configuration Directive for a Custom Authentication Handler

```
read threshold_values;
read request_properties;
request source_risk from analysis_server;
request target_risk from analysis_server;
request system_risk from analysis_server;
if(source_risk>source_threshold OR
target_risk > target_threshold OR
system_risk > system_threshold) {
send authentication_request;
if(credentials_incorrect)
{ return AUTHENTICATION_REQUIRED; }
else
{ return AUTHENTICATION_GRANTED; }
}
else
{return NO_AUTHENTICATION_REQUIRED;}
```

Figure 5: Pseudocode for Authentication Module

The authentication module was actually written as a content handler, because the Authentication handlers are somewhat restricted and would not allow for the type of random challenge authentication that was desired in this case. The example shown establishes three independent thresholds, any of which could be used to trigger authentication for the requesting source. The corresponding pseudo code for the authentication module is shown in Fig. 5.

The system threshold is higher to limit the number of authentication requests that are necessary when the risk for a particular source or target is not yet at a suspicious level. It also offers protection for as-yet untouched resources when the majority of intrusive traffic is concentrated elsewhere in the system.

The analysis server receives requests and then loads the appropriate risk analysis module, dynamically generating queries to the event database to select the appropriate events. The risk module then generates the risk measure which is returned to the service requester.


```
<Location /s>
PerlAccessHandler SourcePermissionRestrict
PerlSetVar Action_A1_RiskThreshold 26
PerlSetVar Action_A2_RiskThreshold 32
PerlSetVar Action_A3_RiskThreshold 36
PerlSetVar Action_A4_RiskThreshold 39
PerlSetVar SourceLockoutThreshold 41
</Location>
```

The first set of results pertains to the evaluation of the risk analysis model. The goal of this testing is to demonstrate the following:

1. That the essential assumption of the model - that of escalating risk - is valid for scenarios that involve successive, related intrusion attempts
2. That this assumption can be validated experimentally using real data sets
3. That various techniques exist, and can be used effectively, to deal with some of the problems regarding data quality including: false positives and false negatives

Similar results were presented for an earlier version of the analysis model in [17]. The data for the tests were from the first of the two scenario-specific data sets provided by the Lincoln Laboratory [18]. The data set records a distributed denial of service attack and was divided into the following five phases: 1) an IPsweep of the target network from a remote site 2) a probe of live IP's to look for the sadmind daemon running on Solaris hosts 3) break-ins via the sadmind vulnerability, both successful and unsuccessful on those hosts 4) installation of the trojan mstream DDoS software on three hosts in the target network and 5) launching the denial of service attack. Initial test results showed the intruder (as described in the provided labeling data) with the highest risk rating after a majority of the attack had concluded. Unsatisfactorily, however, due to false positives early in the tests some other nodes were initially given higher risk ratings during the first phases of the attack. In addition, the overall number of nodes that were assessed as potential intruders was high. Two different alert filtering techniques were applied, in an effort to improve the data accuracy and reduce false positives. The first was to use a technique proposed in [] to filter out alerts that do not correspond to the exploitation of a 'concrete vulnerability'. A concrete vulnerability is defined in this case as one which is listed in the CVE [6], a standardized database for software vulnerabilities. In order to compile a working database to check vulnerability signatures, the latest CVE entries were downloaded and stored in a relational database. The results for the second round of testing using the concrete vulnerability filtering are shown in Fig. 6 and Fig. 7.

The latter part of the risk progression is relatively flat because the intrusion detection system being used failed to detect some of the later events involved in the attack sequence. And while the risk model does not make provisions for detecting attacks which are missed by intrusion assessment mechanisms, the use of historical data to assess the threat posed by the source at least ensures that the same risk level based on earlier behavior is maintained. In this way, the model is somewhat tolerant of missed detections.

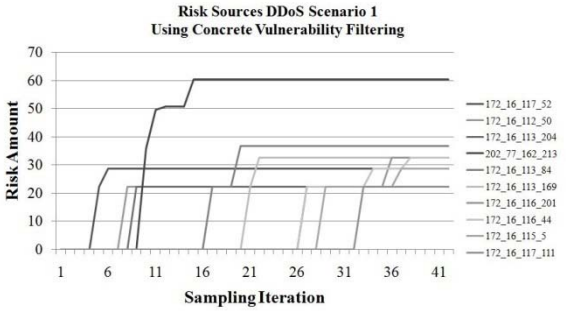

Figure 6: Risk Estimations for Packet Sources Using Concrete Vulnerability Filtering

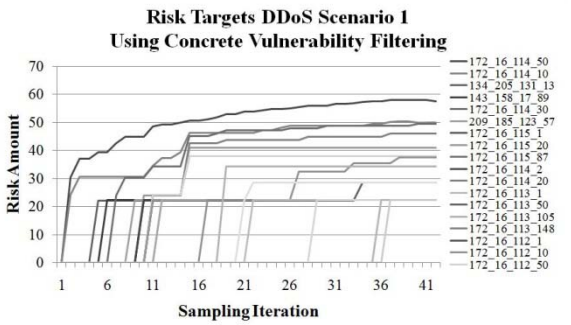

Figure 7: Risk Estimations for Packet Targets Using Concrete Vulnerability Filtering

The risk assessments in the second set of test results were still somewhat inaccurate; a number of nodes on the local network were rated as suspicious and up until approximately the 9th sampling iteration the actual intruder does not have the highest risk rating. A second alert filtering technique was used to further increase the accuracy of the assessment: configuration verification. This is similar to the approach of verify alerts using network knowledge as discussed in [19, 20]. In this case, a database was constructed with all of the known, labeled nodes in the data set, the operating system running on the node and its version of the operating system. Each time an alert was generated this database was consulted to see if the vulnerability being reported actually matched the configuration of the targeted machine. If there was no match, the alert was discarded. Using these two filtering techniques in conjunction the risk assessment reflected the single-intruder nature of the data set, as shown in Fig. 8.

After applying the filtering techniques, the results for target risk estimation were improved. Final results for target risk estimation are shown in Fig. 9: only the nodes actually attacked in the data set are rated, and those nodes for which successful attacks are launched are rated with the highest risk values.

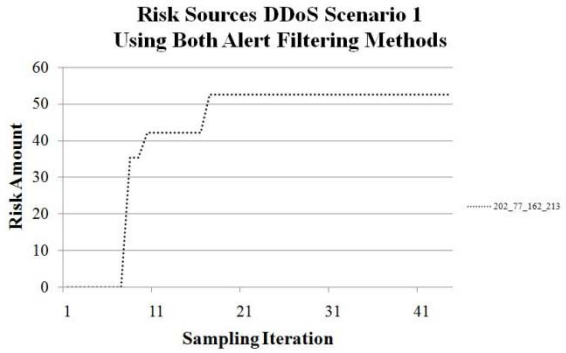

Figure 8: Risk Estimation for Packet Sources Using Both Filtering Methods

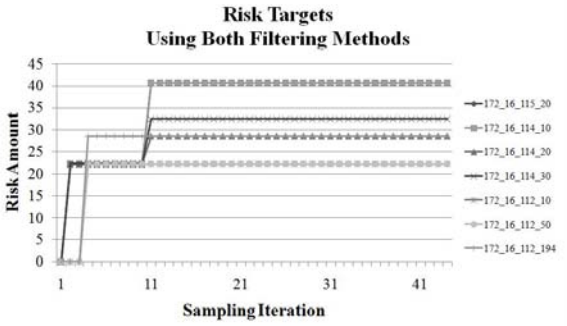

Figure 9: Risk Estimation for Packet Targets Using Both Filtering Methods

VI. EVALUATING THE EFFECT OF RISK DATA ON ACCESS CONTROL POLICY ENFORCEMENT

This second set of testing results is designed to demonstrate results of testing the second of the two architectures (the risk analysis server integrated with Apache) with real time incoming requests. In order to effectively illustrate the effect of the three chosen response techniques, three different scenarios were generated with a webserver traffic simulator and requests were sent to two different webservers: one using the three analysis modules described previously, and another only using the notion of the global system threat to trigger response techniques. Whereas validation of the risk model could be performed with a captured data set being replayed over the network, the use of the response strategies will require active connections to the access control system and hence demands live traffic.

The traffic simulator creates an array of requesting nodes S where s_i is a member of S, each with an intrusiveness rating i_r, an inter-request period p and a total request life *l*. The webserver is arranged as an array of target resources T (where t_i is a member of T). Each t_i has a set of valid actions $\{a_1, a_2, \ldots a_n\}$ and invalid or intrusive actions $\{i_1, i_2, \ldots i_k\}$. Every p seconds (or some randomized derivative of p seconds) request source s_i selects a member of T and then based on its intrusiveness rating, selects either a normal or intrusive action to perform on the resource. Sources with a higher i_r, have a greater probability of selecting an intrusive action for each request. In practice, these intrusiveness or maliciousness ratings range from 0% to 90%.

The risk analysis model was fixed for the simulation of the three scenarios detailed below. Vulnerability weightings were the following: high severity ($w(H) = 3$), medium severity ($w(M) = 2$) and low severity ($w(L) = 1$). The risk multiplier (γ) was set to 10, to provide a more noticeable difference between various assessments.

Scenario 1: Single Intruder, Vulnerability Probing

In this first scenario, a single intruder executes intrusive requests on several system resources - a method indicative of probing for which vulnerabilities have been patched or which configuration holes have been closed. The rest of the sources generating system requests are normal users - executing little or no requests that could be categorized as intrusive. The requests were generated over the course of a three hour simulation. The request trace for the intruder demonstrates that requests for different actions are denied based on his overall risk profile and eventually the intruder is locked out from all system requests. Meanwhile, requests from the other users are still permitted. A summary of the results for a simulation of this scenario are presented in Table 2.

<Location /sc2> SetHandler perl-script AccessHandler SourcePermissionRestrict PerlSetVar TargetRiskThreshold 45 </Location>	<Location /sc2> SetHandler perl-script AccessHandler SystemPermissionRestrict PerlSetVar SystemRiskThreshold 65 </Location>
(a)	(b)

Figure 10: Access Control Policies for Server 1 (a) and Server 2 (b) in scenario 1

Table 2: Simulation Results for Scenario 1

Property	Server 1 (Source Risk)	Server 2 (System Risk)
Total Requests	2472	2472
Total Intrusive Requests	230	230
Intrusive Requests Denied	229	179
Percentage Denied	99.5%	77.8%
Total Normal Requests	2242	2242
Normal Requests Denied	16	1751
Percentage Denied	.7%	78.1%

In this scenario all of the intrusive requests were from the single intruder. Server 1 began to deny requests from the intruder after their source risk passed the threshold of 45. The normal requests blocked by server 1 were also from the intruder. Once the system risk for server 2 passes the threshold, it begins to deny requests from all sources.

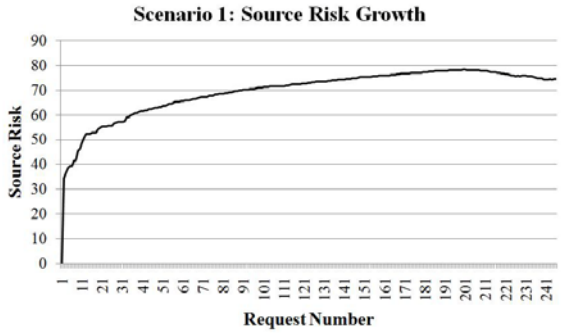

Figure 11: Growth of Risk for the Intruder

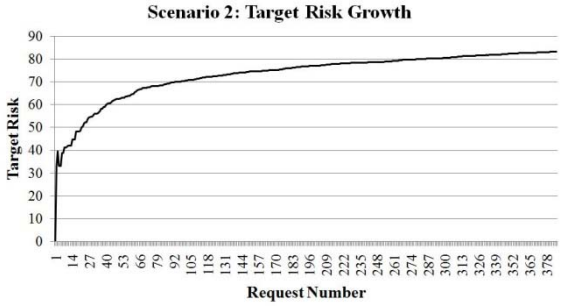

Figure 13: Growth of Risk for the Targeted Resource in Scenario 2

Scenario 2: Multiple Intruders, Single Target, Many-to-One Attack

In the second scenario, multiple intruders target the same resource with two different attacks. This could correspond to the publication of a new vulnerability for an existing service. In the interim period some non-intrusive requests are allowed on the resource, but when the target risk reaches the threshold, all requests to the target are denied. A summary of the results for a simulation of this scenario are presented in Table 3.

<Location /sc2> SetHandler perl-script AccessHandler TargetPermissionRestrict PerlSetVar TargetRiskThreshold 45 </Location>	<Location /sc2> SetHandler perl-script AccessHandler SystemPermissionRestrict PerlSetVar SystemRiskThreshold 65 </Location>
(a)	(b)

Figure 12: Access Control Policies for Server 1 (a) and Server 2 (b) in scenario 2

Table 3: Simulation Results for Scenario 2

Statistic	System 1 (Target Risk)
Total Requests	1023
Total Intrusive Requests	320
Intrusive Requests Blocked	319
Percentage Denied	93.5%
Total Normal Requests	703
Normal Requests Denied	65
Percentage Denied	9.2%

The testing for scenario two demonstrates that using target risk when a particular resource is being targeted can increase the number of intrusive requests that are blocked while maintaining availability for the other system resources. During this simulation, both the system risk and the target risk for the targeted resource peaked at 83. This was due to the fact that all of the intrusive requests in the entire system were directed at the same resource. While the system risk threshold could have been raised to decrease the percentage of normal requests that were denied, it would have also increased the number of intrusive requests that were blocked.

Scenario 3: Multiple Attackers on Various Resources

In the third scenario, multiple intruders attack multiple system resources. This could correspond to a system with high traffic levels that sees exploitation attempts on multiple resources from multiple sources in a given period of time. Using both source and target risk levels, requests at various points in the overall request trace are responded to by a request for authentication. Eventually when the system risk level passes the threshold, all initial requests are responded to by requests for authentication. A summary of the results for a simulation of this scenario are presented in Table 4. The simulation was run for approximately 2.5 hours with nodes generating requests at all levels of maliciousness (and thus there is no clear intruder).

<Location /sc3> SetHandler perl-script PerlHandler AuthChain PerlSetVar SystemRiskThreshold 65 PerlSetVar SourceRiskThreshold 33 PerlSetVar TargetRiskThreshold 45 PerlSetVar AuthExpiration 300000 </Location>	<Location /sc3> SetHandler perl-script AccessHandler SystemPermissionRestrict PerlSetVar SystemRiskThreshold 65 </Location>
(a)	(b)

Figure 14: Access Control Policies for Server 1 (a) and Server 2 (b) in scenario 3

Table 4: Simulation Results for Scenario 3

Statistic	Server 1	Server 2
Total Requests Received	875	875
Total Intrusive Requests	437	437
Intrusive Requests Authenticated	409	252
Percentage Authenticated	93.5%	57.7%
Total Non-Intrusive Requests	438	438
Non-Intrusive Requests Auth.	385	368
Percentage Authenticated	87.9%	84%

Due to the use of source, target and system risk information, the policy for server one was stricter. Despite this, the proportion of non-intrusive requests that were responded to by a request for authentication was only four percent higher than for server two. This number of non-intrusive requests also includes requests from nodes with high maliciousness ratings such as 90%, which would otherwise be deemed intruders but were classified at non-intrusive because the particular request being classified was not intrusive.

VII. CONCLUSION AND FUTURE WORK

Another key challenge was how to design an attack response that was tempered and still effective. We chose to use a strategy of restricting access permissions as the response to likely intrusive behavior by attaching risk thresholds to permissions on the controlled resources. A risk assessment was synthesized from the provided data on vulnerability exploitation attempts in order to provide a quantifiable measurement of the changing state of system entities in relation to their prospect of being attacked. Because the risk assessments were calculated for individual system entities, the assessment data also allowed for more granular responses.

The actual results of the attack simulations showed a marked improvement for the ratio of intrusive requests that were denied using the risk assessments. In the scenario that simulated an attacker performing vulnerability probing against the webserver, 99% of the intrusive requests were denied, while only .7% of the normal requests were denied. In the case of multiple intruders for one target attack, the framework denied 93.5% of the intrusive requests while only denying 9.2% of the non-intrusive requests. Even in the scenario of multiple intruders on multiple resources, where authentication was employed as a response, more intrusive requests were authenticated than non-intrusive ones (93.5% to 87.9%, respectively), leading to a more efficient use of resources over the approach of authenticating all requests in situations of elevated risk.

REFERENCES

[1] CERT Coordination Center. Overview of Attack Trends. Technical report, CERT Coordination Center, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2002.

[2] IBM Global Technology Services. IBM Internet Security Systems X-force 2007 Trend Statistics. Technical report, Internet Security Systems - IBM Global Technology Services, 2007.

[3] Peter Mell, Karen Scarfone, and Sasha Romanosky. A complete guide to the common vulnerability scoring system version 2.0. http://www.first.org/cvss/cvss-guide.pdf, June 2007.

[4] M.S. Ahmed, E. Al-Shaer, and L. Khan. A novel quantitative approach for measuring network security. *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1957–1965, April 2008.

[5] Andreas Hess and Niels Karowski. Automated protection of end-systems against known attacks. In *Proceedings of IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation*, Tuebingen, Germany, 2006.

[6] The MITRE Corporation. Cve - common vulnerabilities and exposures, Retrieved April 18 2012. From http://cve.mitre.org.

[7] Tanya Ryutov, Clifford Neuman, Dongho Kim, and Li Zhou. Integrated access control and intrusion detection for web servers. *Parallel and Distributed Systems, IEEE Transactions on*, 14:841–850, 2003.

[8] Tanya Ryutov, Clifford Neuman, Dongho Kim, and Li Zhou. Integrated access control and intrusion detection for web servers. *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pages 394–401, 2003.

[9] Lawrence Teo, Gail-Joon Ahn, and Yuliang Zheng. Dynamic and risk-aware network access management. *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 217–230, 2003.

[10] Nathan Dimmock, András Belokosztolszki, David Eyers, Jean Bacon, and Ken Moody. Using trust and risk in role-based access control policies. *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 156–162, 2004.

[11] Natalia Stakhanova, Samik Basu, and Johnny Wong. A taxonomy of intrusion response systems. *Int. J. Inf. Comput. Secur.*, 1(1/2):169–184, 2007.

[12] Kazimierz Kowalski and Mohsen Beheshti. Improving security through analysis of log files intersections. *International Journal of Network Security*, 7(1):24–30, July 2008.

[13] Hassan Rasheed and Randy Y.C. Chow. Adaptive risk-aware application-level access control. In *The 2009 Conference on Security and Management (SAM'09)*, pages 10–16, Las Vegas, NV, July 2009.

[14] Curtis Carver, Jr. and Udo Pooch. An intrusion response taxonomy and its role in automatic intrusion response. *IEEE Workshop on Information Assurance and Security*, 2000.

[15] Eric Fisch. *Intrusion Damage Control and Assessment: A Taxonomy and Implementation of Automated Responses to Intrusive Behavior*. PhD thesis, Texas A&M University, 1996.

[16] Herve Debar, David A. Curry, and Benjamin S. Feinstein. The intrusion detection message exchange format (IDMEF), 2007. Request For Comments (Experimental).

[17] Hassan Rasheed and Randy Y. C. Chow. Automated risk assessment for sources and targets of vulnerability exploitation. In *Proceedings of the*

2009 WRI World Congress on Computer Science and Information Engineering - Volume 01, CSIE '09, pages 150–154, Washington, DC, USA, 2009. IEEE Computer Society.

[18] MIT Lincoln Laboratory. 2000 DARPA Intrusion Detection Scenario Specific Data Sets. , http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000data.html, Accessed September 2008.

[19] C. Kruegel and W. Robertson. Alert verification: Determining the success of intrusion attempts. In *1st Workshop on the Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA 2004)*, July 2004.

[20] U. Shankar and Vern Paxson. Active mapping: Resisting nids evasion without altering traffic. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 44, Washington, DC, USA, 2003. IEEE Computer Society.

Hassan Rasheed received his Ph.D. and M.S. degrees in Computer Engineering from the University of Florida. He is currently an Assistant Professor in the Deanship of Information Technology at Taif University in Taif, Saudi Arabia. He also previously served as a postdoctoral research fellow at the Air Force Research Lab in Rome, NY. His broad industry experience includes positions at Ford Motor Company and Honeywell Space Systems and ranges from systems development to management. His current research interests include information security, distributed systems and data mining.

