

Trust Assessment from Observed Behavior: Toward and Essential Service for Trusted Network Computing¹

Partha Pal, Franklin Webber, Michael Atighetchi and Nate Combs
BBN Technologies, 10 Moulton Street, Cambridge, MA 02138
{ppal,fwebber,matighet}@bbn.com, ncombs@roaringshrimp.com²

Abstract

Modern distributed information systems handle increasingly critical data and computation, but there is no systematic way to assess whether a given part of the system can be entrusted with such data and computation on a continuous basis. In a highly interconnected networked environment, components with varying levels of trustworthiness must interact with each other. Occurrence and spread of attack induced failure imply that hosts once trusted cannot remain equally trusted all the time. System components and human operators can benefit from a scheme that assesses the trustworthiness of hosts i.e., the confidence that individual hosts are not corrupt, on a continuous basis by adjusting and adapting their behavior when a host's trustworthiness diminishes. In this work in progress report we present an accusation based trust assessment scheme.

1. Introduction

The notion of a Trusted Computing Base (TCB) [1] was conceived to provide a framework to perform security critical computation—a computing environment that can be trusted to satisfy the predefined security policies all the time. However, experience teaches us that a TCB is hard to build, prove and maintain. Any approximation of a TCB is thus usually found in a small but key part (of a larger system) that is perhaps built with more care and tested better. Information systems are becoming increasingly complex, geographically distributed and interconnected with other systems (and sometimes with public networks like the Internet), making it unlikely that a critical information system can ever be trusted in the TCB sense; and therefore, components with varying levels of trust must, by necessity, interact with each other. Distribution, interoperation and network connectivity further complicates the situation by making current networked information systems more vulnerable to cyber attacks, and more easily accessible to potential adversaries.

This suggests that one must find an alternative means to decide whether components or subsystems can be entrusted with critical computation or data. Furthermore, trustworthiness can neither be black-and-white nor remain constant over time. A small custom-created component, e.g., a Network Intrusion Detection System (NIDS) appliance, may generally be more trusted than other components that are built upon or with products with known vulnerabilities (such as IIS or MySQL server). But, a component should become less trusted when it is under attack. Therefore, a trust management overlay is needed that observes the system, assesses trust values, and makes the assessed values available for system components and operators. A component may then decide to seek the service from a different peer because the current provider cannot be trusted as much, or the system operator may decide to temporarily isolate the untrusted host to investigate it further and to contain the spread of corruption.

In the context of testing and evaluating a highly survivable distributed networked information system for US DoD, we have been developing such a trust management scheme, which we will describe in this paper. One of the innovative aspects of this work is that it presents a framework to use *accusations about a host H* as well as *accusations made by H* to decide whether H is corrupted and hence not to be trusted. The framework is *flexible*, i.e., the relative importance of how accusations from or about H is weighted in assessing the trust value. Our position is that such a service is essential for any trusted network computing environment or infrastructure. Results of preliminary evaluation, lessons learned and future work are also presented.

2. Background

Information superiority is one of the key goals of the US military. Toward that end, a number of efforts have been under way to enhance the reach and capability of combat information management systems. The concept of the Joint Battlespace Infosphere (JBI) [2] is one such effort undertaken by the Air Force, which aims to enable

¹ This research was supported by DARPA under contract No. F30602-02-C-0134

² This work was done by the author when he was at BBN. This is his current email address.

effective interaction among disparate military computer systems that must exchange information in support of various network-centric warfare activities ranging from intelligence gathering to mission planning and tactical operations. The JBI aims to achieve this goal by using a publish-subscribe based framework that will allow diverse computing systems to interact in a decoupled manner as long as they follow a common API for defining Information Objects (IO); and for performing publish, subscribe and query (PSQ) operations, which are facilitated by a set of core services. A JBI instantiation is therefore a set of applications and core services that are needed to execute a specific mission. One such instantiation, simulating the execution of an Air Tasking Order (ATO) and planning of a concurrent airlift through the theater, was used as the demonstration vehicle in the DARPA OASIS Dem/Val program [3].

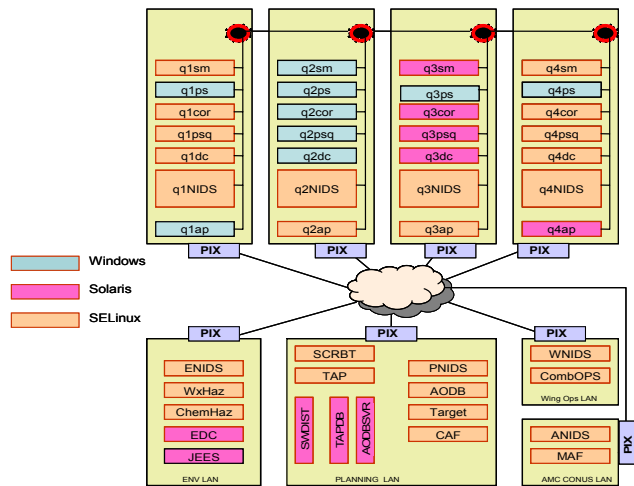


Figure 1: Notional view of the defense-enabled system
 OASIS Dem/Val is a systems program, integrating many of the mechanisms and techniques developed by prior DARPA programs in the areas of Information Security, Intrusion Tolerance, Survivability, and Information Assurance and Information Survivability as well as COTS security products into a new architecture to allow development of the next level of secure and survivable information systems. A BBN led team started with the undefended JBI instantiation (called the “undefended JBI” henceforth) and the prescribed survivability requirements that were meant to establish new high-water mark in survivable information systems. We first designed and validated a survivability architecture (henceforth called the “DPASA³ architecture”), and then used it to create a survivable version (refer to Figure 1) of the undefended JBI (called the “defense-enabled JBI”

³ DPASA is the project acronym, which stands for Designing Protection and Adaptation into a Survivability Architecture.

henceforth), which was subsequently subjected to multiple rounds of Red Team exercises.

One of the key survivability requirements was that the defense-enabled system must survive 12 hours of sustained attacks by a sophisticated adversary. The adversary was considered extremely knowledgeable about the system and fairly unconstrained in terms of rules of engagement (ROE). Attacks of all flavors including breach of integrity and availability were inbound. Therefore, the survivability architecture adopted a stringent view of “trust”, requiring proofs and independent corroboration as much as possible before granting a service or executing a command.

The architecture includes an elaborate configuration of observers ranging from traditional Intrusion Detection Systems (IDSs) to application embedded sensors and policy enforcement engines, to capture information that might facilitate independent corroboration. It uses cryptographic credentials and digital signature to prove identity and integrity; voting and Byzantine Fault Tolerant schemes to detect, convict and tolerate corrupt behavior.

Figure 1 shows the notional representation of the defense-enabled JBI. The clients are organized in 4 different LANS, shown at the bottom. The core services of the publish/subscribe platform are organized in 4 redundant LANS, known as quads, shown at the top. These redundant LANS, known as quads, are numbered 1 through 4. In Figure 1 each host name has a prefix, q1, q2, q3 or q4, to indicate which quad it belongs to. Each of the 8 LANs has a NIDS (Network Intrusion Detection System), and a PIX VPN (Virtual Private Network) router as the perimeter gateway. A managed switch at each quad strictly controls which hosts within a quad can talk to each other, as well as which hosts can talk across the quads. Each host is equipped with an ADF card that restricts communication to virtual private groups of hosts. As shown by the color code in the figure, DPASA makes use of 3 different operating systems. This OS diversity reduces the chance that a successful attack on one OS will work on every host. The communication paths between hosts are designed to ensure that no successful attack on any single OS can give the attacker complete control of any quad or of any service that spans all 4 quads.

Each quad is organized into 3 zones. The outermost zone of a quad X, known as the *crumple zone* consists of a single “Access Proxy” (qXap) that acts as the first point of contact for the clients accessing the core services. The access proxy host runs application-level proxies for functions in the next zone upstream, the *operations zone*, and enforces a number of size and rate checks on traffic between the core functions and the clients. The innermost zone, known as the *executive zone*, consists of a single “System Manager” (qXsm) that is responsible for overall system management and interfacing with human

operators. The operations zone consists of 4 hosts namely qXdc, qXpsq, qXps and qXcorr. The “Downstream Controller” (qXdc) mediates the System Manager’s (qXsm) interaction with the clients, essentially preventing direct contact between the executive zone and the clients. The “Policy Server” (qXps) is the administration and management station for the ADF cards. The “Correlator” (qXcorr) runs the Emerald correlator and a custom Advisor component. The “PSQ server” (qXpsq) runs the main publish-subscribe service and associated repositories, along with a number of custom features, and participate in a protocol able to tolerate 1 Byzantine failure, and can adapt to run in a degraded mode with fewer than 4 servers. Components that play a role in the trust management scheme is explained more detail in Section 3. Further details of the architecture can be found in [4].

3. Trust Management Scheme

We describe the trust management scheme of the defense-enabled JBI terms of the participating components, the information flow contributing to the trust assessment, and the algorithm used for assessing the trust values.

3.1 Participating Components

Sensors: The sensors are responsible for observing and reporting unexpected incidents and anomalies as “alerts”. Each software component in the DPASA architecture is embedded with sensors that report application-level failures or anomalies such as connection failure, presentation of invalid signature or credential, failure to reach an agreement etc. The EMERALD NIDS [5] appliances, deployed on each LAN, report network-level incidents and anomalies. Similarly, the EMERALD Host Intrusion Detection Systems (HIDS), running on Solaris hosts in the architecture, report host level incidents. In addition, each host (irrespective of the OS it runs) makes use of a file system integrity checker (Veracity [6]) and a mechanism that enforces process security policies (SELinux [7], or Cisco Security Agent [8]) which report integrity and policy violation or attempted violation. Finally, a specialized software component called the Local Controller (LC) runs on each host and reports the host’s health status (such as CPU load, memory usage, free disk space).

The Correlator and the Advisor: The Correlator and Advisor components act as the receiver of alerts generated by the sensors. The correlator host in each quad runs these two components (and a “T” processes that logs and splits the incoming alert stream to feed the two functional components in parallel). The Correlator comprehends the alert stream using EMERALD’s

proprietary algorithms [9, 10], aggregates and ranks the critical events for human consumption. The Advisor receives and processes the same alert stream, but instead of ranking and aggregating them, interprets the alerts as accusations and attempts to compute whether a DPASA component is trustable or not (see section 3.3).

System Manager (SM): The system manager in a quad is responsible for receiving the output from the Correlator and Advisor of its quad. The ranked and aggregated alerts produced by the Correlator are displayed for human operators, whereas the trust values are used to recommended defensive responses (such as restarting a component, rebooting or isolating a host). There is one system manager in each quad, and they must reconcile and agree on the trust values they obtain from their respective Advisors. Therefore, the SMs engage in an agreement protocol, where the reported trust values are disseminated to create a composite trust picture of all hosts in the system. The response selection begins when SM reach an agreement that a given host cannot be trusted anymore.

3.2 Information Flow

Alerts from sensors in the client networks are sent to all 4 quads. Alerts from sensors situated in a quad are sent to the “T” process of that quad only. Therefore, alerts originating from within a quad are not seen by the other quads. Alerts originating in the client networks are further subjected to rate-limiting when they pass through the Access Proxy in the crumple zone. While the alerts are directly used by the trust management scheme, heartbeats play a secondary but important role. All software components send periodic heartbeats, which ultimately results in a visual display at the System Manager. Like alerts, heartbeats from client networks go to all quads, while heartbeats from within a quad are visible to the local System Manager only. Absence or flood of heartbeats generates alerts. Absence of heartbeat for a component C provides a visual clue to the human operator to look for the trust value of C’s host, or ranked and correlated alerts about C.

3.3. Trust Assessment

By “trust”, we mean the confidence that the given host can be trusted to carry out critical computation. The goal of the trust assessment mechanism is to estimate the probability that any given host in the information system is not corrupted (and therefore, can be trusted), based on the alerts it receives. Information contained with an alert include the alert type—indicating a broad categorization of alerts, alert observer—the sensor who observed the reported event, alert source—information about the

location of the event being reported, and alert target—information about the end destination (if any) of the reported event etc. In many cases, the observer, source and target may point to the same host indicating a collocated sensor is reporting a local event. The source and target fields point to different hosts if the reported event is about an interaction between remote entities, such as attempts to open a network connection, or presentation of a certificate. The observer and the source (or target) fields point to different hosts in most of the NIDS alerts. Based on these fields, each alert is interpreted as "Host S accusing host X for reason R". Usually, S is the observer host and X is the source host. Accusations cover a wide range of specificity and seriousness. For example, a "security policy violation" alert is considered both very serious and very specific (definitive indication of an attack), whereas a "port scan" alert is considered a less specific and less serious accusation. In this section we will first present an abstract formulation of an accusation-based trust assessment scheme. We will then show how a version of this is implemented in the defense-enabled JBI.

3.3.1. Trust, Accusation, Masking and False-

Accusation. Let us assume that N is the number of hosts in the system. The probability that individual hosts are trusted are maintained in a vector T of size N called the **trust index**, where T(i) denotes the probability that host H_i is trusted. Accusations are aggregated and maintained in an **Accusation matrix** A of size [N×N]. The value A[i,j] denotes the certainty with which host H_i accuses host H_j. Not all individual accusations, implied by individual alerts, are meaningful or equally plausible. The physical architecture constraints whether or how much a given host can observe of other hosts. These constraints are maintained in a **Mask matrix** M, where M[i,j] captures the plausibility of node i accusing node j and reflects an accurate and static description of the topology/architecture of the subject system. A simple example of a mask matrix is a matrix whose elements can either be 0 or 1, where M[i,j] is 1 only when H_i can accuse H_j, and 0 otherwise.

The trust assessment scheme needs to filter out or degrade the value of alerts that are impossible or less plausible based on its understanding of the topology/architecture of the system (such alerts may have been sent by an attacker). This can be performed by computing the **Masked accusation matrix**, which is element-wise product of M and A:

$MA[i, j] = M[i, j] * A[i, j]$ for all i and j
The elements of MA are non-zero whenever i can plausibly accuse j. If M is the simple (0,1) valued matrix described earlier, MA[i,j] becomes 0 when H_i cannot accuse H_j, and retains the value of A[i,j] when they can.

Next, we define the **False Accusation matrix** as $F = A - MA$, where F[i,j] has a minimum value of 0. F is intended to highlight the accusations that were reported, but not plausible in the given system based on topological and architectural constraints—i.e. false accusations. Continuing with the simple example of M being a (0,1) valued matrix, a false accusation from H_i against H_j will show $F[i, j] = A[i, j]$ (the purported score of the false accusation), and all legitimate accusations from H_i against H_j will show $F[i, j] = 0$.

Initially⁴ T(i) is set to 1, indicating that all hosts are trusted when the system starts, and A[i,j] is set to 0 for all i and j, implying that no one has accused anyone yet. If a host H_i is later restarted as a defensive response, T(i) is set to 1, and A[j,i] is reset to 0 for all j.

T(i) declines if host H_i is accused by trusted nodes or accuses other nodes apparently falsely. When any such probability gets below a threshold, the corresponding host is deemed not trustworthy anymore. This threshold can be defined for each host individually, or for classes of hosts, and set statically or learned dynamically. Let us now describe how T and A is updated when a new accusation (in the form of an alert) comes in.

3.3.2. Algorithms for Updating Trust Based on

Accusations. For each alert L where host H_i accuses host H_j, A is updated as follows:

$A[i, j] = \text{Max}\{A[i, j], \text{score}(L)\}$, where the score represents the veracity of the alert—a value between 0 and 1, and is inferred from the information contained in L.

All other elements are unaffected.

Note that a) because of this rule, and because restarting H_j as a defensive response⁵ resets A[i,j] for all i, A[i,j] retains the highest score of "H_i is accusing H_j" alerts seen until the accused host is rebooted, and b) when A is updated, MA and F are also automatically updated because these quantities depend on A, as described in Section 3.3.1.

Our assessment of trust of a host H is based on two trust components:

1. Host trust based on accusation it makes, denoted by T1, and

⁴ Note that this initialization indicating that no host is corrupt when the system starts assumes that there is no life-cycle attack. If that is not the case, i.e., some hosts are to be trusted less, then initialization should be done differently, i.e., T(i) should not be initialized to 1 for the suspected hosts.

⁵ Note that this applies only when a host is rebooted as part of a defensive response. If the attacker reboots a host, the SM and the Advsiar components will not reset the values.

2. Host trust based on accusations made by others, denoted by T2

where both T1 and T2 are vectors of length N, and there are a number of strategies to define these quantities.

T1 is used for reducing the trust of hosts that make false accusations, and there are a number of possibilities. A Draconian approach could decide that false accusers are corrupt, i.e.:

$$T1[i] = 0, \text{ if, for any } j, F[i,j] > 0 \\ = T[i] \text{ otherwise} \text{ ----- (1a)}$$

A kinder, gentler approach could consider the fraction of false accusations made that could have been made, i.e.:

$$T1[i] = (\text{SUM}_j (F[i,j])) / (\text{SUM}_j (1-M[i,j])) \\ \text{----- (1b)}$$

One way to define T2 is to make a host that can be accused in many ways relatively immune to any individual accusation. To compute T2 under this strategy, we first normalize row i of MA by T1[i], down-weighting accusations made by H_i based on T1 trust. This is the normalized “trust-weighted” masked accusation matrix, MA', given by:

$$MA' [i,j] = \text{Min}(T[i], T1[i]) * MA[i,j]$$

Heuristically, MA' (i,j) / MA[i,j] is the credibility of an accusation made by H_i against H_j (if any). Considering all accusers, the average normalized credibility is thus:

$$\text{SUM}_i (MA' [i,j]) / \text{SUM}_i (M[i,j])$$

The T2 trust value of host H_j can be computed as the complement of the normalized average credibility:

$$T2[j] = 1 - (\text{SUM}_i (MA' [i,j]) / \text{SUM}_i (M[i,j])) \\ \text{----- (2a)}$$

Another strategy that can be adopted to compute T2 is to assume that each accuser is always tentatively trustworthy. Under this strategy, T2 is computed as follows:

$$T2[j] = \text{PRODUCT}_i (1 - A[j,i] * .5) \\ \text{for all } i \text{ ----- (2b)}$$

Once T1 and T2 are computed, trust index T can be updated as:

$$T[k] = \text{Min}(T1[k], T2[k]) \text{ ----- (3a)}$$

Or, alternatively:

$$T[k] = 1 - (1 - T1[k]) * (1 - T2[k]) \text{ ----- (3b)}$$

The latter represents the complement of the probability that the host is trusted in both T1 and T2 sense, while the former takes the worst of T1 and T2 trust. Finally, the accusational diagonal, which is a node's self-accusation value, is used to provisionally lower the trust value⁶. This value is always believed and is the complement of trust:

$$T[k] = \text{Min}(1 - A[k,k], T[k])$$

In summary, this section presents a customizable framework for calculating trust based on accusations. As

demonstrated by the various alternatives above, individual formulations can be customized with varying properties or bias on how the accusers and accused are treated.

3.3.3. Implementation in the defense-enabled JBI. The trust assessment scheme implemented in the defense-enabled JBI made several simplifying assumptions. The 34 different alert types it currently handles are scored at three levels: alerts with highest seriousness and specificity get a very high score (1), minimally serious or unspecific alerts get a very low score (0), and all other alerts get a medium score (.5). The threshold that determines when a host cannot be trusted is defined in terms of two classes: “core hosts” and “client hosts”. Threshold values for these classes are set to 0.3 and 0.6 respectively, indicating that the DPASA architecture is designed to be more skeptical (i.e., will take fewer accusations to go below the threshold) of clients (located on the fringes of the architecture and hence more vulnerable) than the core components (that are redundant and better protected). It also assumes that M is a (0,1) matrix, and follows the draconian option for T1 computation. Instead of describing the matrix itself, Table 1 shows the filtering rules that are applicable to the defense-enabled JBI.

Table 1. Examples of filter rules (in English)

Any host can accuse itself
qXps hosts can accuse all ADF-enabled hosts
Access proxy hosts can accuse client hosts
Client hosts can accuse access proxy hosts
Access proxy hosts cannot accuse system manager hosts
A client host cannot accuse another client host
A PSQ host cannot accuse system manager hosts
EMERALD NIDS can accuse access proxy hosts and other client hosts in the same LAN
Quad EMERALD NIDS can accuse access proxy and all client hosts

For each alert L where host H_i accuses host H_j, A is updated as described in section 3.3.2, but in addition it also sets:

$$A[j,i] = \text{Max} \{ A[j,i], \text{score}(L) \}$$

This represents the skeptical view taken in the defense-enabled JBI in answering the question: when H_i accuses H_j, is it an indication of a problem with H_i or H_j? The symmetry implies, that this formulation considers both equally probable, unless other accusations overrules. T is computed using the simplified formula:

$$T(i) = \text{min}(T1(1), T2(1))$$

Finally, self accusations are always trusted, which is accommodated by adjusting T by the accusational diagonal adjustment as explained in Section 3.3.2.

⁶ Self-accusation values can also be used to adjust T1 so that subsequent trust values to which the node contributes are normalized.

4. Preliminary Evaluation, Lessons Learned and Future Work

The trust assessment scheme described above considers alerts from H as well as about H to decide if a host's trust has fallen below the threshold. Typically, multiple accusations about H (unless a very specific and serious alert accuses H) are required to cause H's trust value to fall below the specified threshold. In the defense-enabled JBI, such an event generates a trust advice about H, which is sent to the System Manager⁷. The trust value provides some indication of how severe the remedial response needs to be. For instance, an advice about H with a very low trust value may warrant quarantine of H (it is compromised so severely that it is better to isolate the component and run without that function), whereas an advice with a better trust value may warrant restarting of H (upon reboot the component may function correctly).

Table-2 shows how many alerts of what type are needed to make client and core hosts untrusted in the defense-enabled JBI. The analysis assumes that the accused host did not falsely accuse any other hosts.

In our initial internal test and evaluation of this implementation, we observed few false negatives (i.e., failing to declare a corrupt host untrusted), but did observe a considerable number of false positives (i.e., declaring a good host untrusted). Some of these false positives were due to incorrect characterization of alerts, as well as imperfect policy. To understand the latter, note that if the SELinux policy at a host is imperfect, a high-score "SELinux violation" self accusation may be produced in course of normal operation, which will significantly lower the trust of the host in question. In the 1st Red Team exercise the system was deployed with these flaws, but the exercise saw mainly DoS attacks that were mostly at the outermost boundary of the system. The trust assessment scheme was not exercised much.

After the 1st exercise, we attempted to address the known alert classification problems and tightened the SELinux policies in an attempt to reduce high-score false positive alerts. This updated version was deployed for the 2nd Red Team exercise. In this exercise, the red team was given privilege and access to hosts within the defended system so that they can preposition their attack⁸. This implied that many accusations that would have happened in a real attack situation (when the adversary

⁷ A System Manager in a quad receives trust advice about all client hosts, but only the core hosts that are in its own quad. This is because, as we explained earlier, alerts from qX hosts go only to qXCor.

⁸ Even though it was known that the adversary controls some hosts, it was not known to the Blue Team which hosts were corrupt, making it impossible to initialize T

gains these access and privileges) were eliminated. The trust assessment scheme was therefore mostly exposed to the subsequent events caused by the attack in other hosts. The enormous amount of data collected by the 20 Red Team runs is still being analyzed. So far we are able to capture only a few high level metrics from the raw data that are presented in Figure 2 and 3.

Table 2 – Numerical analysis of trust reduction based on accusations received

Accusations Received about H	T(H)	Comments
One with medium score (.5)	0.75	H is still trusted
Two with medium score (.5)	0.5625	If H is a client, it falls below the threshold of 0.6, and becomes untrusted. If H is a core host, it will still be trusted
One with high score (1)	0.5	If H is a core host it is still trusted. If H is a client host, it is untrusted
One high (1) and one medium (.5)	0.375	Same as above
One high (1) and two medium (.5)	0.28125	Below threshold for core hosts (0.3), if H is a core host it will now become untrusted
Two high (1)	0.25	Irrespective of whether H is a client or a core host, H is untrusted

Figure 2 shows the number of alerts received at each quad over the 20 runs as boxplots. The average median is 525 alerts, and the biggest outlier is 60681 alerts received at Quad3. The outliers occurred in three of the 20 runs in which the Red Team ran application-level flooding attacks from inside the core targeted at the System Managers.

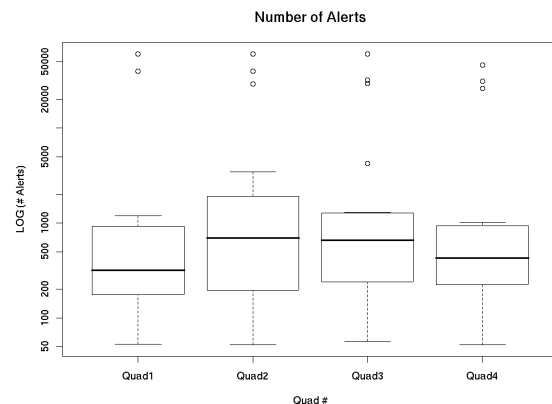


Figure 2 – Boxplot distribution of Alerts received at each quad

Figure 3 shows the number of instances where core and client hosts went below their respective predefined thresholds over the 20 runs. Even though the number of hosts declared untrusted was small compared to the number of alerts, we expect that false positives were prevalent. Side by side analysis of the Red Team logs and the advice are needed to identify which of these were false positives or negatives. A similar analysis with the blue team log is needed to identify which of these were deliberately caused by the blue team member’s manually scrambling to recover from a fatal attack effect. More detailed analysis of the generated advice is needed to identify how many of these cases involved more than one System Managers declaring an individual client and whether that is a false positive. We expect to report some of these results of such analysis in a follow up paper, but currently we are able to verify that in 5 of these 20 runs, the system reduced the trust of one or more hosts that were actually corrupt.

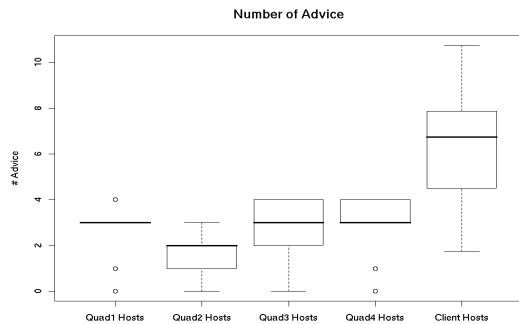


Figure 3 – Boxplot distributions of hosts that went below trust threshold

5. Conclusion

The concept of using accusation is somewhat complementary to using “reputation” (see [11] for a discussion). Reputation systems can be p2p, but centralized management of p2p based reputation is not uncommon (consider the list of how sellers are rated in eBay). Accusation or reputation-based systems both can be considered instances of a more generalized notion of behavior-based trust, where trust assignment is based on observed behavior.

The work presented here is still ongoing. The initial experience with the accusation-based trust assessment scheme showed good promise that it can help other system components and operators by generating trust advice about hosts in the system, but it also clearly exposed the difficulty of alert characterization, tuning of parameters, and eliminating false positives. The tuning of parameters involved in scoring and thresholds are currently set empirically, and are not likely to be optimal. Ongoing analysis of the data collected from past Red

Team exercises is expected to provide more insight, and continued experimentation with live attacks is needed to converge on more optimal tuning parameters. One other avenue for potential improvement is adding a “learning” capability—which will enable the assessment scheme to adjust its tunable parameters automatically based on feedbacks (from users) and computation of an utility function.

Trust assessment is part of our ongoing investigation of the bigger problem of managing survivable systems. Managing a survivable system not only involves managing the functional parts, but also the mechanisms that were added for survivability—a major part of which relies on redundancy and defensive response by adaptation. We indicated in this paper how trustworthiness is used in management decisions—both automated, by the system, and also by the human operators—in terms of the defense-enabled JBI. The discussion in this paper is presented in terms of trustworthiness of individual hosts. But, the scheme could be enhanced for assessing trustworthiness of sub-host entities (such as objects, processes etc) as well as multi-host entities (such as an entire LAN). Accurate estimation of trustworthiness therefore could play a role in a number of decision-making control loops, and it remains to be seen if and how much improvement such incorporation of trustworthiness consideration in decision-making loops can achieve.

6. Acknowledgement

The authors would like to thank DARPA and AFRL for supporting the work, the rest of the DPASA team for their help in developing the defense-enabled JBI within which the trust assessment scheme was implemented, and especially to Martin Fong of SRI for helping us formulating the trust assessment scheme.

7. References

- [1] National Computer Security Center, Department of Defense Trusted Computer System Evaluation Criteria. National Computer Security Center, 1985
- [2] USAF Scientific Advisory Board, Report on Building the Joint Battlespace Infosphere, Vol. 1. The US Air Force SAB, 1999
- [3] OASIS Dem/Val Program Page, http://www.darpa.mil/ipto/Programs/oasis_demval/index.htm. Defense Advance Research Projects Agency

[4] P. Pal et al, Designing Protection and Adaptation into a Survivability Architecture. Final Report of the DPASA Project, submitted to DARPA. BBN Technologies, 2006

[5] P. A. Porras and P. G. Neumann, "Emerald: Event Monitoring Enabling Response to Anomalous Live Disturbances", *Proc. of the 20th National Information Information Systems Security Conference*, National Institute of Standards and Technology (1997)

[6] RockSoft Inc., Veracity Product Page at:
<http://www.rocksoft.com/rocksoft>. RockSoft Inc

[7] NSA, SELinux Project Home Page at:
<http://www.nsa.gov/selinux/>

[8] CISCO, CSA Information Page at:
<http://www.cisco.com/en/US/products/sw/secursw/ps505>

[9] A. Valdes and K. Skinner, "Probabilistic Alert Correlation", *Proc. 2001 International Workshop on Recent Advances in Intrusion Detection (RAID)*, Springer Verlag, LNCS 2212. October 2001

[10] P. Porras, M. Fong and A. Valdes, "A Mission-Impact Based to INFOSEC Alarm Correlation", *Proc. 2002 International Workshop on Recent Advances in Intrusion Detection (RAID)*, Springer-Verlag LNCS 2516. October 2002

[11] S. Marti and H. G. Molina, "Taxonomy of Trust: Categorizing P2P Reputation Systems". Invited paper, *COMNET Special Issue on Trust and Reputation in Peer-to-Peer Systems*. 2005.