Proactive Intrusion Prevention and Response via Attack Graphs

Steven Noel and Sushil Jajodia

Center for Secure Information Systems, George Mason University
{snoel, jajodia}@gmu.edu

Network defense today is largely reactive rather than proactive, and lacks sufficient context for optimal countermeasures. Administrators and security analysts are overwhelmed by constant outside threats, complexity of security measures, and network growth. Today's status quo for network defense is often reduced to mere triage and after-the-fact remediation. This chapter examines proactive methods of attack risk reduction and response through attack graphs. Our attack graphs map potential paths of vulnerability through a network, showing exactly how attackers may penetrate a network. Attack graph analysis identifies critical vulnerabilities and provides strategies for protection of critical network assets. But because of operational realities, vulnerability paths often remain. In such cases attack graphs provide an ideal methodology for planning appropriate attack responses. This includes optimal placement of intrusion detection sensors, correlating intrusion alarms, accounting for missed detections, prioritizing alarms, and predicting next possible attack steps.

1 Introduction

Network security is inherently difficult. Protocols are often insecure, software is frequently vulnerable, and educating end-users is expensive. Security is labor-intensive and requires specialized knowledge, and is error prone because of the complexity, volume, and frequent changes in security data and network configurations. Network administrators and security analysts can easily become overwhelmed, and reduced to simply reacting to security events. A much more proactive stance is needed.

Security concerns in a network are also highly interdependent, i.e., susceptibility to attack can depend on multiple vulnerabilities across the network. Attackers can combine such vulnerabilities to incrementally penetrate a network and compromise critical systems. But traditional security tools are generally point solutions that provide only a small part of the picture. They give few clues as to how attackers might exploit combinations of vulnerabilities to advance an attack on a network. It remains a painful exercise to combine results from multiple tools and data sources to understand one's true vulnerability against sophisticated multi-step attacks. It can be difficult even for experienced analysts to recognize such risks, and is especially challenging for large dynamically evolving networks.

Security is not a one-time single-point fix, but rather a continuous process, as exemplified in the *protect-detect-react* life cycle. To *protect* from attacks, we take steps to prevent them from succeeding. Still we must understand that not all attacks can be averted in advance, and there must usually remain some residual

vulnerability even after reasonable protective measures have been applied. We then rely on the *detect* phase to identify actual attack instances. But the detection process needs to be tied to residual vulnerabilities, especially ones that lie on paths to critical network resources. Once attacks are detected, comprehensive capabilities are needed to *react* to them based on vulnerability paths. We can thus reduce the impact of attacks through advance planning, by knowing the paths of vulnerability through our networks.

To create such a proactive stance, we need to transform raw security data into attack roadmaps that help us prioritize and manage risks, maintain situational awareness, and plan for optimal countermeasures. This chapter describes the latest advances in an innovative proactive approach to network security called *Topological Vulnerability Analysis* (TVA)[1,2]. By analyzing vulnerability interdependencies, TVA builds a complete map showing all possible paths of multi-step penetration into a network, organized as a concise attack graph. The TVA attack graph then supports proactive network defenses across the entire protect-detect-react life cycle. This includes identifying critical vulnerabilities, computing key security metrics, guiding the configuration of intrusion detection systems, correlating and prioritizing intrusion alarms, reducing false alarms, and planning optimal attack responses.

The next section reviews the TVA approach and provides an illustrative example. Section 3 describes the process of capturing network attack models in TVA for simulating multi-step penetrating attacks. Section 4 discusses how to apply attack graphs for optimal network protection, and Section 5 covers the application of attack graphs to intrusion detection and response. Section 6 summarizes our approach, and suggests possible future advances.

2 Topological Vulnerability Analysis (TVA)

Because of vulnerability interdependencies across networks, a topological (attack graph) approach is needed, especially for proactive defense against insidious multi-step attacks. The traditional approach that treats network data and events in isolation, without the context provided by attack graphs, is clearly insufficient. TVA combines vulnerabilities in ways that real attackers might do, discovering all attack paths through a network.

2.1 Overview of Approach

As shown in Figure 1, TVA begins by building an input attack model, based on the network configuration and potential attacker exploits. It then analyzes this

model, matching exploits against the network configuration to trace multi-step attacks through the network. From the resulting attack graph, TVA then generates recommendations for optimal network hardening. The attack graph can also be explored through interactive visualization, for more in-depth risk analysis, including "what-if" scenarios. The TVA attack graph also supports computation of various metrics for measuring overall network security.

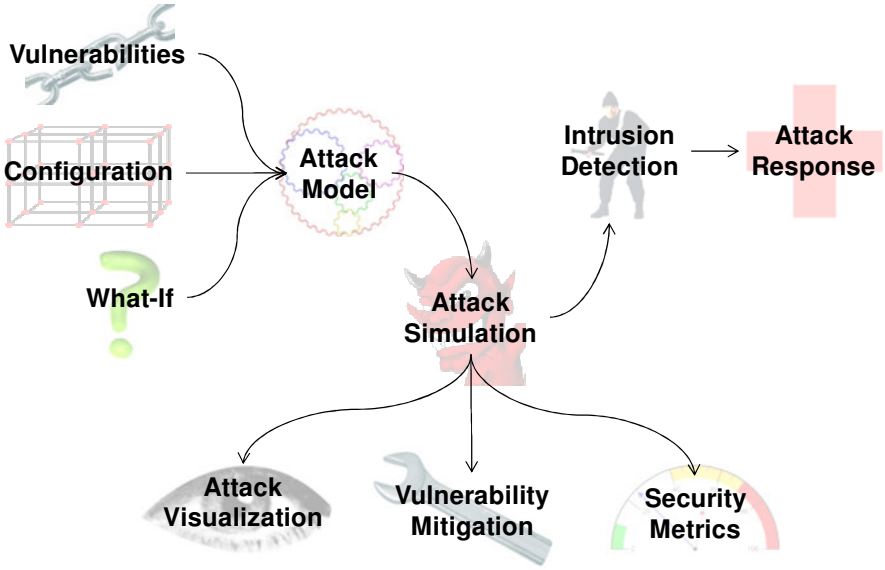

Fig. 1. Topological Vulnerability Analysis (TVA). From an input network attack model, TVA simulates multi-step attacks, forming an attack graph. In advance of attack, the attack graph is visualized, and used to mitigate critical vulnerabilities and compute security metrics. The attack graph guides the intrusion detection process, and helps formulate optimal attack responses.

While the attack graph guides optimal strategies for preventing attacks (e.g., patching critical vulnerabilities), because of realistic operational constrains (availability of patches, mission-critical services, etc.), there usually remain some residual attack paths through a network. At this point, the residual attack graph provides the necessary context for dealing with intrusion attempts. This includes guidance for the deployment and configuration of intrusion detection systems, correlation of intrusion alarms, and prediction of next possible attack steps for appropriate attack response.

For example, the attack graph can guide the placement of intrusion detection sensors to cover all attack paths, while minimizing sensors redundancy. The attack graph can also filter false intrusion alarms, based on known paths of

residual vulnerability. The graph also provides the context for correlating isolated alarms as part of a larger multi-step attack penetration. It also shows the next possible vulnerabilities that could be exploited by an attacker, whether they lie on vulnerability paths to critical network resources, and attack proximity to critical resources. This in turn supports optimal planning and response against attacks, while minimizing effects of false alarms and purposeful misdirection by the attacker.

2.2 Illustrative Example

As a simple illustration of our TVA attack graph approach, consider the small network in Figure 2. Here, a firewall is intended to protect an internal network (web server, mail server, and file server) from an outside attacker. The firewall allows incoming web traffic to the web server, and blocks all other incoming traffic. In this attack scenario, we wish to know if an outside attacker can compromise the mail server, through one or more attack steps.

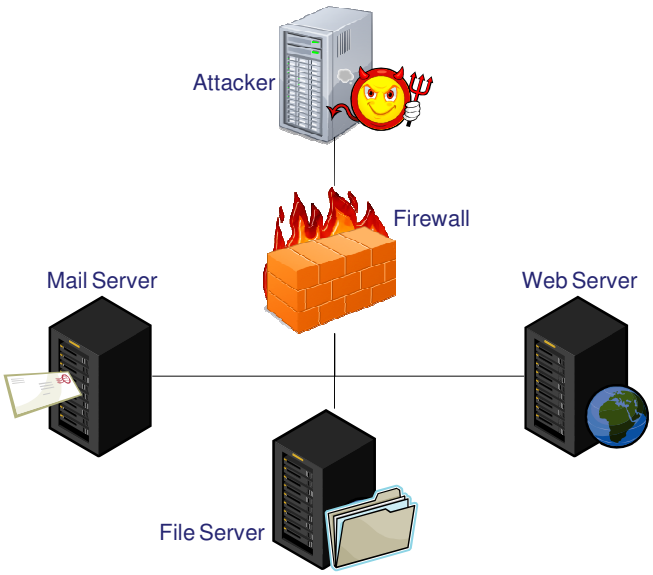

Fig. 2. Small example network. The firewall allows web traffic to the web server, and blocks all other incoming traffic.

To model this scenario, we need to capture elements of the network configuration relevant to attack penetration. This includes the existence of vulnerable software (services) on hosts, as well as connectivity allowed to

vulnerable services. We also need a set of potential attacker exploits that may work against the vulnerable services. For example, we could run a vulnerability scanning tool (e.g., Nessus[3]) against the hosts in the internal network to map their vulnerabilities, and feed this into the TVA model. We could then rely on a database of modeled exploits based on the full set of vulnerabilities detected by Nessus. To incorporate the connectivity-limiting effects of the firewall, we could scan through the firewall (as well as behind it), to implicitly capture firewall effects, or process the firewall rules directly.

Figure 3 shows the resulting attack graph for this scenario. There is indeed a path from the outside to the inside mail server, via a critical vulnerability exposed through the firewall. Figure 3(a) is a high-level view of the attack graph. This shows one vulnerability being exploited (through the firewall) from the outside to the inside.

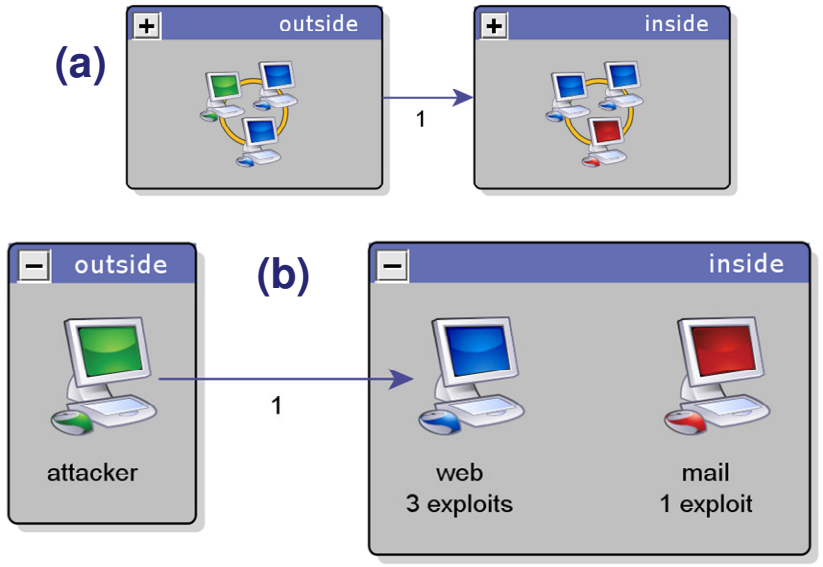

Fig. 3. Attack graph for network in Figure 2. This identifies a critical vulnerability along a path from an outside attacker to the inside mail server. Here, (a) is a high-level overview, while (b) shows additional details.

Figure 3(b) is a more detailed view, showing that the attacker can directly exploit a vulnerability on the web server. From the web server, the attacker can directly attack the mail server (implicitly, anywhere behind the firewall), which has one exploitable vulnerability. The file server host does not appear in the attack graph. It does not play a part in this scenario and can be safely ignored,

i.e., there are no vulnerable paths (from attacker to mail server) through that host. Also, note that Nessus (and other vulnerability scanners) generates many alerts that are merely informational (i.e., irrelevant to network penetration), while TVA carefully excludes these.

This attack graph shows how hosts on a network can be exploited through multiple steps, even when the attacker cannot access them directly. It is not directly possible to compromise the mail server from the outside because of the policy enforced by the firewall. But TVA shows that the attack goal can be reached indirectly, in this case through a sequence of two exploits. Further, it shows that addressing a single critical vulnerability (from among four) would prevent the attack. While such a result may be easy enough for an experienced analyst on a small network, for maintaining a proactive security posture on realistic networks an automated tool is crucial.

3 Attack Modeling and Simulation

TVA decomposes attack graph generation into two phases: (1) capture of an input network attack model, and (2) using the model to simulate multi-step network penetration. The attack model represents the network configuration and potential attacker exploits. In attack simulation, the input model is analyzed to form a graph of causally interdependent exploits (attack graph), according to user-specified constraints.

3.1 Network Attack Modeling

In TVA, the network attack model includes aspects of the network *configuration* relevant to attack penetration, as well as a set of potential attacker *exploits* that match attributes of the configuration. The TVA approach can apply to many different types of attack models (even non-cyber models) as long as a common schema is employed across the model.

Figure 4 is an example of one such schema for TVA network models. This shows that a network is comprised of machines and collections of machines into protection domains. Here, domains capture the idea that the set of machines in a domain have (implicitly) unrestricted access to one another's vulnerable services (e.g., broadcast domains), which is an important abstraction for model scalability. A machine includes sub-elements and attributes relevant for modeling network attack penetration. This includes operating system (an attribute of a machine element in Figure 4), connections to vulnerable services on other machines, sets

of machines that are trusted, application programs on a machine, groups to which the machine belongs, and user-defined generic attributes.

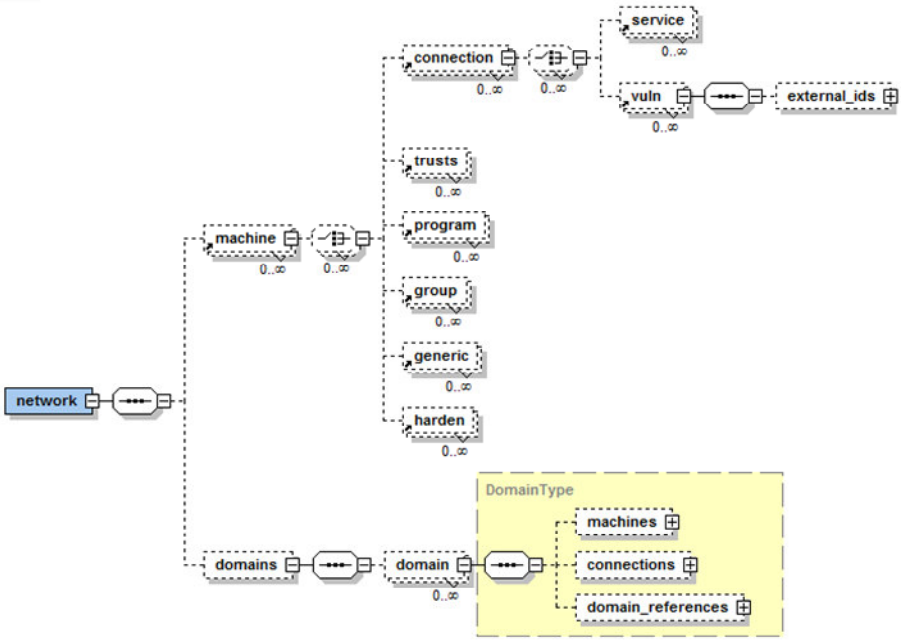

Fig. 4. Example schema for TVA network models. A network is comprised of machines, domains (sets of machines with unrestricted access), network connections, and other kinds of elements relevant to network penetration.

A connection describes how a machine connects to potentially vulnerable services across the network, to ports on other machines or to its own ports. This mirrors the Transmission Control Protocol/Internet Protocol (TCP/IP) reference model, in which a layered connectivity structure represents the various network architectures and protocols[4]. A service connection indicates a running service on a destination machine (to which a source machine can connect). Each connection is composed of a service or application type at the appropriate TCP/IP layer. For example, an HTTP connection specifies the web server name/version at the transport layer. Link-layer connectivity models exploits against the Address Resolution Protocol (ARP), e.g., defining the scope of attacks based on traffic sniffing. Application-layer connectivity models exploits that rely on particular application configurations, trust relationships, or other high-level details.

Alternatively, this schema can represent a detected vulnerability explicitly, using a standard vulnerability identifier, rather than specifying a vulnerable

software component itself. For example, we can identify vulnerable connections through vulnerability identifiers reported by scanning tools such as Nessus, Retina[5], or FoundScan[6]. For convenience, we can also specify a set of any external vulnerability identifiers (synonyms) taken from other vulnerability sources, such as CVE[7] and Bugtraq[8]. In this schema, a harden element defines the hardening of a connection vulnerability, i.e., in the attack simulation, attacks against a given vulnerability on a given machine are omitted.

To keep pace with emerging threats, we must continually monitor sources of reported vulnerabilities, and add those to our database of modeled TVA exploits. We model an attacker exploits in terms of preconditions and postconditions, for generic attacker and victim machines, which are subsequently mapped to the target network.

For populating models automatically, we need to map outputs of network scanning tools to our network schema, which in turn provide preconditions for attack graph exploits. Figure 5 shows example output data for Centennial Discovery[9], a network asset management tool. A Discovery agent deployed on a network host machine reports detailed host configuration data. Each software component, specified by product/manufacturer/version (for simplicity, generalized as a "service"), needs to be checked for vulnerability.

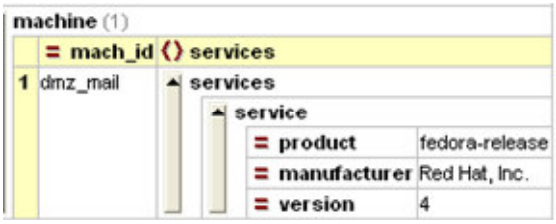

Fig. 5. Software item reported by asset management tool. A network machine has a particular software component deployed.

The host software information needs to be mapped to preconditions for modeled exploits. Figure 6 shows preconditions and postconditions for exploitation of a Bugtraq vulnerability, in terms of generic attacker/victim machines. The preconditions are that (1) the attacker can execute code on the attacking machine, and (2) there is a vulnerable connection from attacker to victim, identified as Bugtraq 13232.

Fig. 6. Example TVA modeled exploit. Connection to vulnerable service on victim machine is specified via Bugtraq vulnerability identifier.

Symantec DeepSight[10] (a web service feed of the Bugtraq database) gives the vulnerable software components (products, manufacturers, and versions) for each vulnerability record. But if host configuration data are gathered from an asset management tool such as Discovery, software descriptions generally differ from those in DeepSight. We therefore map discovered host software components to their corresponding vulnerability records, as in Figure 7. Here we map a Discovery software description (Red Hat Fedora 4) to its Bugtraq vulnerability (13232).

Fig. 7. Software to vulnerability mapping. This indicates that a version of Linux has a particular Bugtraq vulnerability.

Figure 8 illustrates a resulting connection to vulnerable software (Bugtraq 13232) on the host machine. This connection is built into the attack model by mapping the discovered host software to a known vulnerability. Then, since a connection with Bugtraq 13232 is a precondition for a particular exploit, this exploit may be included in an attack graph for this network.

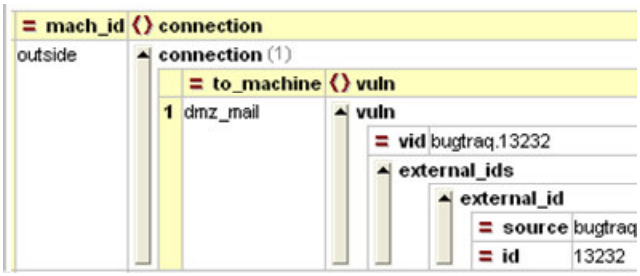

Fig. 8. Network connection to vulnerable software. This specifies that a particular machine connects to another, with a given Bugtraq vulnerability on the destination machine.

The Discovery asset management tool also defines protection domains, i.e., sets of machines with full connectivity to one another's vulnerable services. This is shown in Figure 9. Each protection domain is identified, along with its member machines.

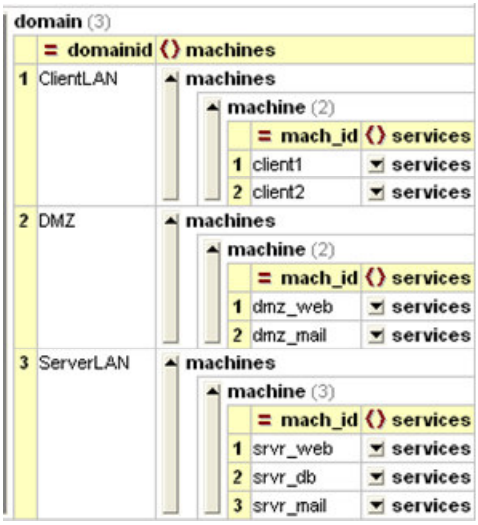

Fig. 9. Protection domains reported by asset management tool. Machines are members of protection domains, and implicitly have unrestricted connectivity to one another's vulnerable services.

Again, the purpose of modeling network configuration in TVA is to support preconditions of modeled attacker exploits. Thus machines are explicitly included in the model only if they offer services over the network that can be exploited. The effects of other network devices such as routers and firewalls is

captured implicitly, in the way that connectivity is modeled at the various network layers, i.e., in the way that they provide/restrict connectivity to vulnerable services on other machines.

For remote (versus host-based) scanners such as Nessus, Retina, and FoundScan, we can capture the implicit effects of devices such as firewalls by scanning from different network vantage points, targeting hosts through the firewall. We can then combine multiple scans from various network locations, building a complete map of connectivity to vulnerable services throughout the network. Alternatively, we can analyze firewall rules directly, adding the resulting vulnerable connections to the model, eliminating the need for scanning through firewalls.

3.2 Attack Simulation

In TVA attack simulation, modeled exploits are matched against the network configuration model, forming an attack graph of causally interdependent exploits, according to user-specified simulation constraints. Because the model is pre-populated through network scans and vulnerability databases, all that remains is to define the attack scenario, e.g., the starting point, the attack goal, and any what-if changes to the network configuration.

In other words, given an input model of network configuration and attacker exploits, the exploits are instantiated for specific attacker/victim machine pairs in the network. Preconditions for instantiated exploits are tested, and resulting postconditions are matched with preconditions of other exploits. Figure 10 shows an exploit that has been instantiated for particular machines in the network model. The attacker and victim machines are no longer generic, i.e., they are defined for actual machines in the network.

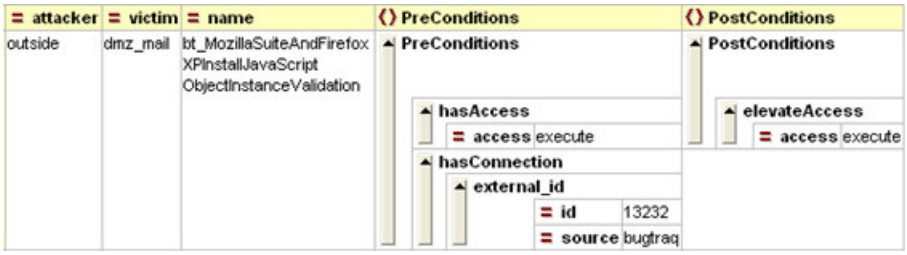

Fig. 10. Exploit instantiated for particular network. Attacker and victim are actual network machines, and preconditions are satisfied from network model.

An attack graph also needs to follow the structure of protection domains defined for the network. Within a protection domain, it is assumed that each machine has unrestricted access (complete connectivity) to vulnerabilities on all other machines in the domain. This implies that the attack graph is completely connected with a domain.

Figure 11 shows example protection domains in attack graph data. Within each domain, the set of all member machines is specified, as well as exploits relevant to each domain. There are two possible types of exploits, within-domain and across-domain. Within-domain exploits are accessible to machines within the protection domain only. Thus it is sufficient to specify the victim machine only, since the attacking machines are implicit. Across-domain exploits are those that attack machines in other domains. Those exploits have both attacker and victim machines specified.

ProtectionDomain (2)

	= name	Machine	Exploit
1	Internet	**Machine** (1)	**Exploit** (3)
2	DMZ	**Machine** (2)	**Exploit** (5)

Machine (1) — Internet

	= name
1	outside

Exploit (3) — Internet

	= attacker	= victim	= name	= withinPdom	PreConditions	PostConditions
1	outside	dmz_mail	bt_MozillaSuiteAndFirefoxXPInstallJavaScriptObjectInstanceValidation	false	PreConditions	PostConditions
2	outside	dmz_mail	bt_MozillaSuiteAndFirefoxDocumentObjectModelNodesCodeExecution	false	PreConditions	PostConditions
3	outside	dmz_web	bt_MicrosoftWindowsMediaPlayer_ASXBufferOverflow	false	PreConditions	PostConditions

Machine (2) — DMZ

	= name
1	dmz_web
2	dmz_mail

Exploit (5) — DMZ

	= attacker	= victim	= name	= withinPdom	PreConditions	PostConditions
1	dmz_mail	srvr_mail	bt_MicrosoftWindowsMediaPlayer_ASXBufferOverflow	false	PreConditions	PostConditions
2	dmz_mail	srvr_mail	bt_WindowsMediaPlayer_ASXBufferOverflow	false	PreConditions	PostConditions
3		dmz_mail	bt_MozillaSuiteAndFirefoxDOMPropertyOverridesCodeExecution	true	PreConditions	PostConditions
4		dmz_mail	bt_MozillaSuiteFirefoxAndThunderbirdMultiple	true	PreConditions	PostConditions
5		dmz_web	bt_MicrosoftInternetExplorerURIDecoding	true	PreConditions	PostConditions

Fig. 11. Protection domains in attack graph data. Within each domain, member machines have unrestricted access to one another's vulnerabilities, and machines can attack into other domains.

In TVA, an attack graph can be completely unconstrained, i.e., all possible attack paths regardless of assumed starting and ending points in the network. In such a scenario, the source of the threat is assumed unknown and no particular

critical network assets are identified as specific attack goals. Figure 12 is an example of such an unconstrained attack graph.

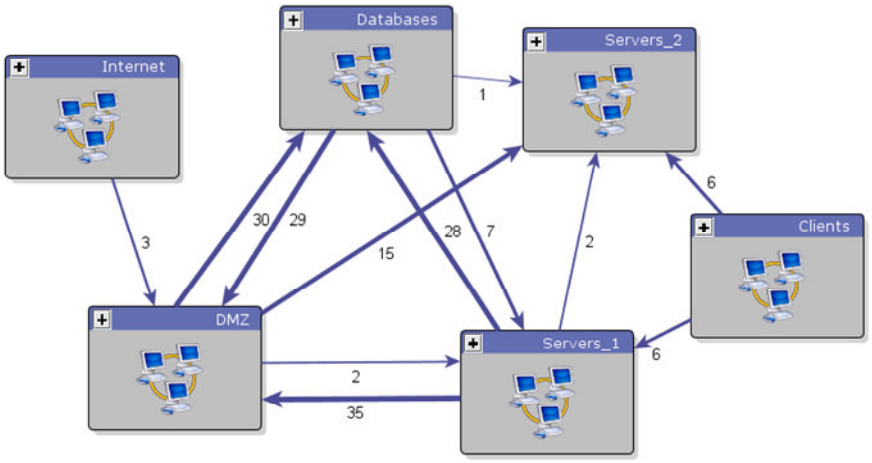

Fig. 12. Unconstrained attack graph. In this scenario, all possible threat sources are considered, and no critical network assets are identified.

Another option is to constrain the attack graph to a given starting point (or points) for the attack. The idea here is that the origin of the attack is assumed, and that only paths that can be reached from the origin are to be included. Figure 13 is an example attack graph in which the attack starting point (Internet) is specified.

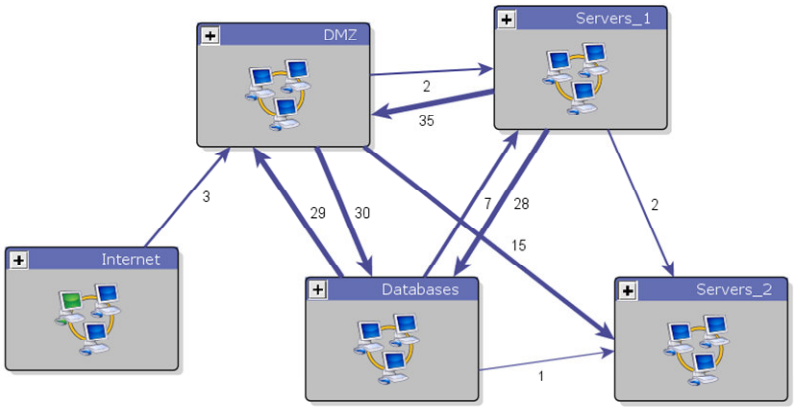

Fig. 13. Attack graph with constrained starting point. Only attack paths emanating from the starting point (Internet) are included in the graph.

Yet another option is to constrain the attack graph so that it ends at a given ending point (or points) serving as the attack goal. Here the idea is that certain critical network assets are to be protected, and only attack paths that reach the critical assets are to be included. This option could be exercised alone (with unconstrained starting point), or combined with a constrained starting point. Figure 14 is an example of the latter, in which the both the attack starting point (Internet) and attack ending point (Databases) are specified.

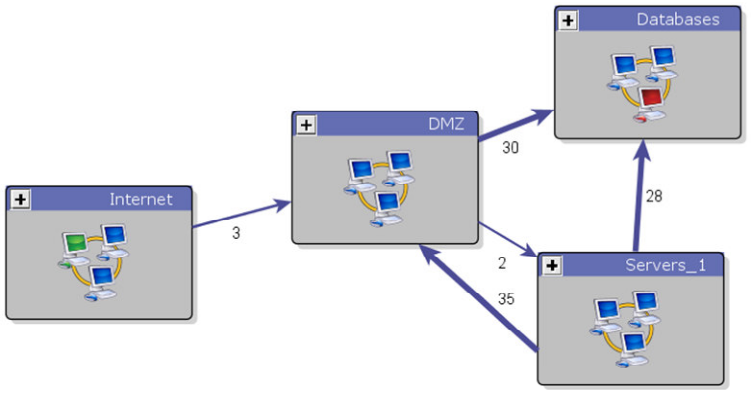

Fig. 14. Attack graph with constrained starting and ending points. Only attack paths emanating from the starting point (Internet) and leading to the ending point (Databases) are in the graph.

The motivation for constraining the attack graph (versus unconstrained) is to reduce the scope of the graph to expected attack scenarios, eliminating unnecessary clutter. For example, in Figure 14, the outgoing edges from the Database protection domain are omitted. If the primary goal is to protect the databases, then attacks *away* from there are less important, i.e., the databases have already been compromised. Similarly, any attacks *into* the starting point could be omitted, since the attacker already has control of it.

One could also argue that the most important attack paths to consider are the most direct ones, i.e., leading from the attack start or leading to the attack goal. This is shown in Figure 15. Here there are two scenarios considered. In the scenario in Figure 15(a), the attack starting point is assumed, and the graph consists of all direct paths from the starting point. In Figure 15(b), both the attack starting point and goal points are assumed, and the graph consists of all direct paths from the starting point to the goal point.

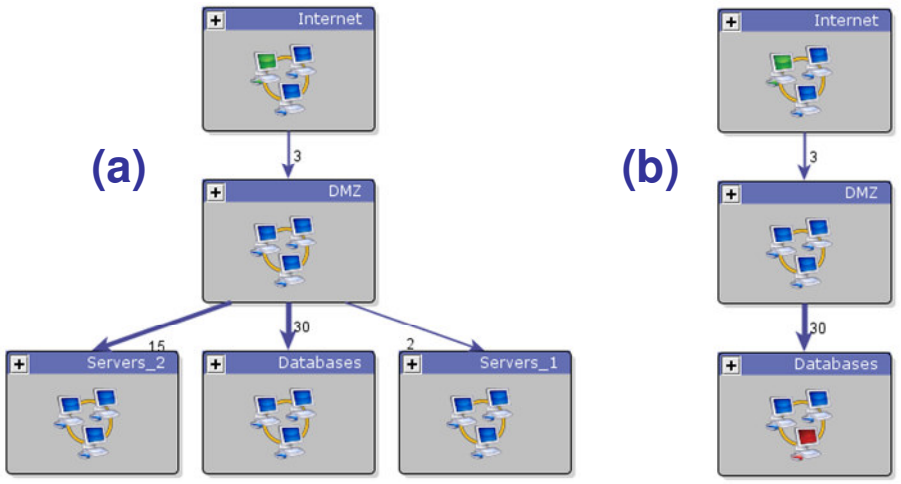

Fig. 15. Attack graph constrained to direct attacks. Direct paths are shown from (a) the given starting point, and (b) the given starting and ending points.

Again, the idea is to identify the most critical paths and vulnerabilities, for pre-attack network hardening as well as real-time alarm correlation, prediction, and response. Thus, given the assumed threat sources, attacker behavior, and critical network resources, we can tailor our analysis and defensive measures accordingly.

4 Optimal Network Protection

Attack graphs provide a powerful framework for proactive network defenses. A variety of analytical techniques are available for attack graphs, providing context for informed risk assessment. Attack graphs pinpoint critical vulnerabilities, and form the basis for optimal network hardening. Through sophisticated visualization techniques (purely graph-based as well as geo-spatial), we can interactively explore attack graphs, while effectively managing graph complexity without getting overwhelmed with details. Attack graphs also support a number of key metrics that concisely quantify the overall state of network security.

4.1 Vulnerability Mitigation

TVA automates the defense of networks against multi-step attacks. TVA attack graphs reveal the true scope of threats by mapping sequences of attacker exploits

that can penetrate a network. We can then use these attack graphs to recommend ways to address the threat. This kind of automated support is critical; finding such solutions manually is tedious and error prone, and infeasible for larger networks.

One kind of recommendation is to harden the network at the attack source, i.e., the first layer of defense. This option prevents all further attack penetration beyond the source.

This is shown in Figure 16. Here, we use the same attack scenario (starting and ending points) as in Figure 14. However, the network configuration model is changed slightly, with a resulting change in the attack graph (numbers of exploits between protection domains). For first-layer defense for this network configuration, the recommendation is to block the 20 exploits from Internet to DMZ.

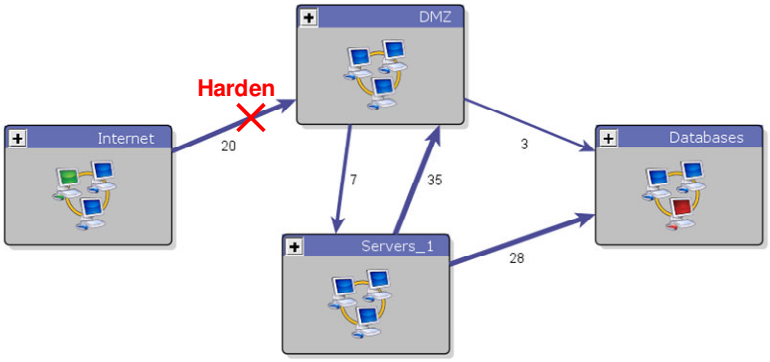

Fig. 16. First-layer network hardening. TVA provides recommendation for hardening the network immediately after the attack starting point.

Figure 17 shows a different kind of recommendation for network hardening, i.e., hardening the network at the attack goal at the last layer of defense. This option protects the attack goal (critical network resource) from all sources of attack, regardless of their origin.

The attack scenario and network configuration are the same as for Figure 16 (the attack graphs are identical). But for last-layer defense in Figure 17, the recommendation is to block the 3 exploits from DMZ to Databases plus the 28 exploits from Servers_1 to Databases, for a total of 31 exploits.

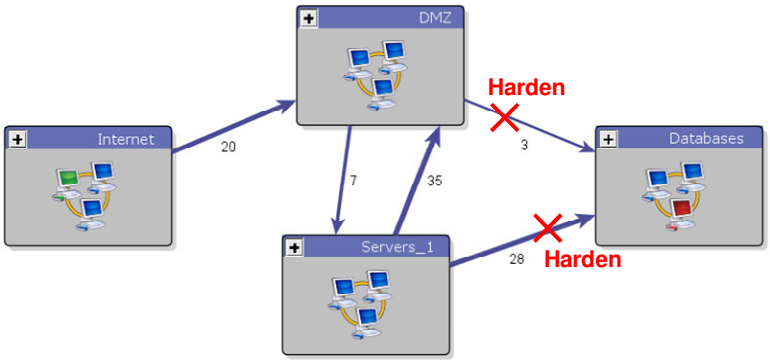

Fig. 17. Last-layer network hardening. TVA provides recommendation for hardening the network immediately before the attack ending point.

Another kind of recommendation is to find the minimum number of blocked exploits that breaks the paths from attack start to attack goal. In other words, we seek to break the graph into two components that separate start from goal, minimizing the total number of blocked exploits[11].

This is shown in Figure 18. The attack graph is the same as in Figure 16 and Figure 17. For the minimum-cost defense, the recommendation is to block the 3 exploits from DMZ to Databases plus the 7 exploits from DMZ to Servers_1, for a total of 10 exploits. This is a savings of 10 blocked exploits in comparison to first-layer hardening, and a savings of 21 blocked exploits in comparison to last-layer hardening.

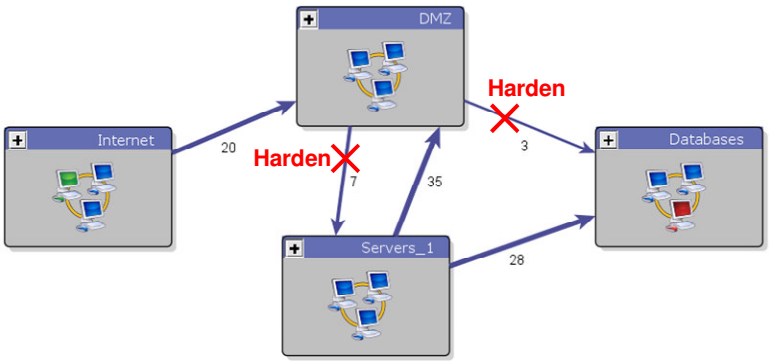

Fig. 18. Minimum-cost network hardening. TVA provides recommendation for hardening the network involving the fewest number of vulnerabilities blocked.

4.2 Attack Graph Visualization

One of the challenges in TVA is managing attack graph complexity. In early formalisms, attack graph complexity is exponential[12,13,14,15], because paths are explicitly enumerated, leading to combinatorial explosion. Under reasonable assumptions attack graph analysis can be formulated as monotonic logic, making it unnecessary to explicitly enumerate states, leading to polynomial (rather than exponential) complexity[16,17,18]. Our protection domain abstraction reduces complexity further, to linear within each domain[19], and complexity can be further reduced based on host configuration regularities[20].

Thus, while it is computationally feasible to generate attack graphs for reasonably large networks, complex graphs can overwhelm an analyst. Rather than presenting attack graph data in their raw form, we present views that aid in rapid understanding of overall attack patterns. Employing a clustered graph framework[21], a clustered portion of the attack graph provides a summarized view, while showing interactions with other clusters. Arbitrarily large and complex attack graphs can be handled in this way, through multiple levels of clustering.

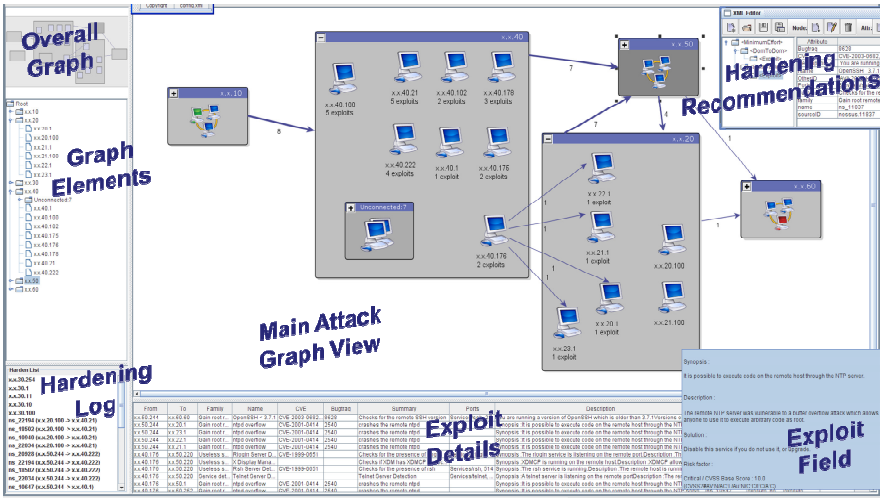

Fig. 19. Attack graph visualization interface. Linked panels support interactive exploration of attack graph, with drilldown for additional details.

Through sophisticated visualization[22], graphs can be rolled up (aggregated) or drilled down (expanded) as the graph is explored. Figure 19 shows a visualization interface for attack graph exploration and analysis. The main view

of the graph shows all possible paths through the network, based on the user-defined attack scenario. In this view, the analyst can expand or collapse graph clusters (protection domains) as desired, rearrange graph elements, and select elements for further details. In the figure, two domains are expanded to show their specific hosts and exploits between them.

When an edge (set of exploits) is selected in the main view, details for the corresponding exploits are provided. Each exploit record contains a number of relevant fields describing the underlying vulnerability. A hierarchical (tree) directory of all attack graph elements is provided, linked to other views. A view of the entire graph is constantly maintained, providing overall context as the main view is rescaled or panned. Automated recommendations for network hardening are provided, and specific hardening actions taken are logged.

The visualization interface in Figure 19 provides an abstract purely cyber-centric view of network attacks. But in some situations, understanding the physical location of attacks may be important, as for assessing mission impact. Given the locality of network elements, we can embed the attack graph into a geo-spatial visualization.

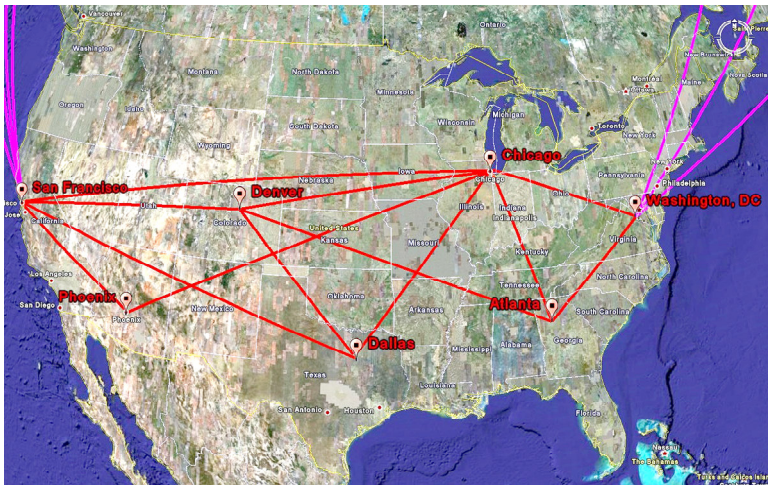

Fig. 20. Geo-spatial attack graph user interface. Provides more direct interpretation of attack elements in terms of mission impact.

This is illustrated in Figure 20. Here, elements of the attack graph are clustered around major centers of the network, and graph edges show exploits between centers. Interactive visualization capabilities can support drilldown for further details at a desired level of resolution.

4.3 Security Metrics

We face sophisticated attackers who may combine multiple vulnerabilities to penetrate networks with devastating impact. Assessment of attack risk must go well beyond simply counting the number of vulnerabilities or vulnerable hosts. Metrics like percentage of patched systems ignore interactions among network vulnerabilities; such metrics are limited, because vulnerabilities in isolation lack context.

Attack graphs show how network vulnerabilities can be combined to stage an attack, providing a framework for much more precise and meaningful security metrics. Attack graph metrics can help quantify risk associated with potential security breaches, guide decisions about responding to attacks, and accurately measure overall network security. Informed risk assessment requires such a quantitative approach. Desirable properties of metrics include being consistently measurable, inexpensive to collect, unambiguous, and having specific context[23]. Metrics based on attack graphs have all these properties.

Some early (non-quantitative) standardization efforts resulted in the System Security Engineering Capability Maturity Model (SSE-CMM)[24]. National Institute of Standards and Technology (NIST) publications outline processes for implementing security metrics[25] and establishing a security baseline[26]. The Common Vulnerability Scoring System (CVSS)[27] provides a way of scoring vulnerabilities based on standard measures. But in all these cases, vulnerabilities are treated in isolation without considering their interdependencies on a target network.

In contrast, attack graph metrics are holistic measures, taking into account patterns of vulnerability paths across the network. These can also be tailored for specific attack scenarios, including assumed threat origins and/or critical resources to protect. They provide consistent measures over time, so that an organization can continually monitor security posture through the course of network operation. They can also be used for evaluating relative security of planned network changes, so that risks can be assessed and alternatives compared well in advance of actual deployment.

One basic metric might be the overall size (vertices and edges) of the attack graph. For example, it may that for a given attack scenario, the attack paths constitute only a small subset of the total network vulnerabilities. This could be for a given attack starting point (with the attack goal unconstrained), thus measuring the total "forward reach" of the attacker. Or it could be for a given attack goal (attack start unconstrained), measuring the "backward susceptibility"

of a critical asset. Alternatively, it could be computed for constrained start and constrained goal, measuring joint attack reachability/susceptibility.

While attack graph size provides a basic indicator, it does not fully quantify levels of effort for defending against attacks. For example, the number of exploits in the first-layer recommendation (Section 4.1) quantifies the effort for blocking initial network penetration. Similarly, the number of exploits in the last-layer recommendation quantifies the effort for blocking final-step critical asset compromise. Then the number of exploits in the minimum-effort recommendation is a measure of the least effort required at a minimum to block attack goal from attack origin. Implicitly, computing the first-layer, last-layer, and minimum-effort metrics require constraining the start, goal, or both (respectively).

Another idea is to normalize metrics by the size of the network, giving a measure that could be compared across networks of different sizes. We could also extend our attack graph models to deal with uncertainties. For example, given that exploits each have individual measures of likelihood, difficulty, etc., we could propagate these through the attack graph, according to the logical implications of exploit interdependencies. This approach could derive an overall measure for the network, e.g., likelihood of catastrophic compromise. Such a measure might then be included in more general assessments of overall business risk. We could then rank risk-mitigation options in terms of maximizing security and minimizing business cost.

The kind of precise measurement provided by attack graphs can also help clarify security requirements and guard against potentially misleading "rule of thumb" assumptions[28]. For example, a network with more total vulnerabilities than another may actually be more secure, e.g., if those vulnerabilities do not lie on paths to critical assets (based on firewall rules, etc.). Or, increased diversity of network host configurations (presumably to make the attacker's job more difficult) may not necessarily increase security. For example, it may provide more options for penetrating into critical assets. Attack graph metrics give precise measures for analyzing these kinds of situations.

5 Intrusion Detection and Response

Attack graph analysis identifies critical vulnerability paths and provides strategies for optimal protection of critical network assets. This enables us to make optimal decisions about hardening the network in advance of attack. But we must also recognize that because of operational constraints such as availability of patches and the need for offering mission-critical services, residual

vulnerability paths usually remain. But the knowledge provided by TVA allows us to plan in advance, and maintain a proactive security posture even in the face of attacks. For example, TVA attack graphs provide the necessary context for deployment and fine tuning of intrusion detection systems, for correlation and prioritization of intrusion alarms, and for attack response.

5.1 Intrusion Detection Guidance

Knowledge of the paths of vulnerability through our network helps us prepare our defenses and plan our responses. TVA attacks graphs can guide the optimal deployment and operation of intrusion detection systems, tailored to our network and its critical assets.

In deploying intrusion detection systems, we must decide where to place detection sensors within the network. Traditionally, intrusion detection sensors are placed at network perimeters, with the idea of detecting attacks from the outside. But with this deployment, traffic in the internal network is not monitored. If an attacker avoids detection at the perimeter, subsequent attack traffic in the internal network is missed.

On the other hand, deploying sensors everywhere may be cost prohibitive, and can overwhelm analysts with floods of alerts. We should strike a balance, in which we cover known residual vulnerability paths, using the fewest sensors necessary. TVA attack graphs provide this balance.

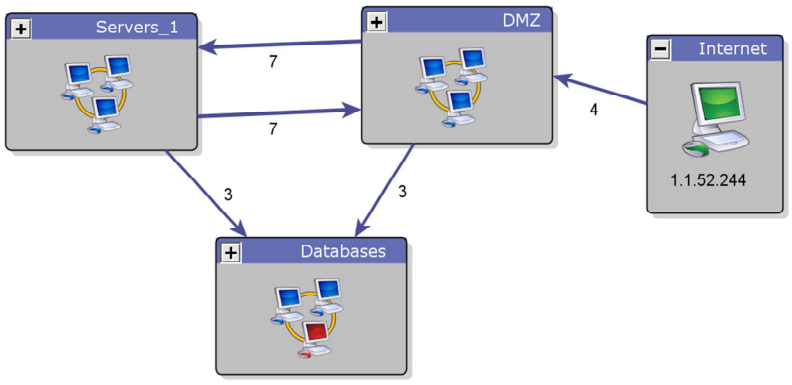

Fig. 21. Residual attack graph. After reasonable measures have been taken to harden the network in advance of attack, some vulnerability paths remain.

Consider the attack graph in Figure 21. Let us assume that this is the residual attack graph after network hardening measures have been applied. So now the

goal is to map this attack graph to the network topology, and embed intrusion detection sensors in the network to cover all the vulnerability paths (with the fewest sensors).

Figure 22 shows the topology for this network, overlaid by the attack paths from Figure 21. Analysis of this joint network representation shows that detection sensors placed at Router A and Router B cover all vulnerability paths, with the fewest sensors. An alternative is to place sensors at Subnet 1, Subnet 4, and Subnet 8, which also covers all paths, but requires three (versus two) sensors.

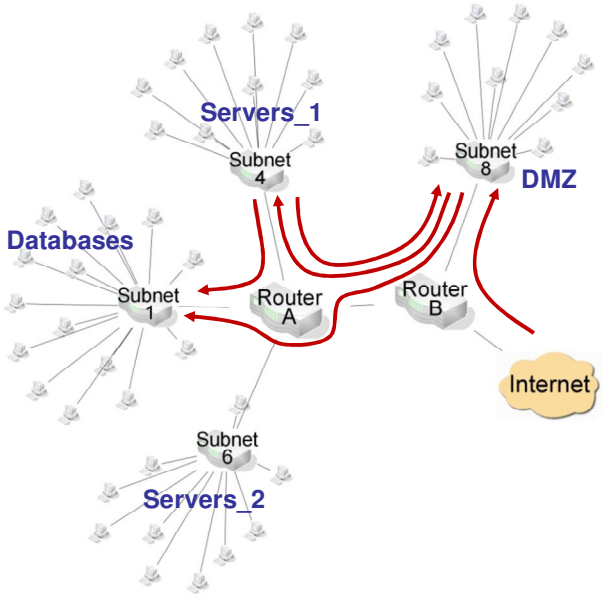

Fig. 22. Intrusion detection sensor deployment. TVA attack graphs guide the placement of sensors to cover all vulnerability paths while minimizing the number of deployed sensors.

In this network, deploying a sensor at the perimeter alone (Router B) will miss attack traffic from Servers_1 to Databases. In the opposite extreme, we might decide to deploy sensors at every subnet (in this case, four subnets) to catch all potential attack traffic. But TVA shows that in fact there are no critical vulnerability paths involving Subnet 6, so deploying a sensor there is wasteful, including continually monitor alerts generated from there. Again, sensors deployed at Router A and Router B are sufficient to cover all vulnerable paths.

For enterprise networks, performing this kind of analysis requires the kind of automated support provided by TVA. Our attack graphs bring together information from a variety of sources over multiple network layers, into a

concise map. While the sensor placement problem itself is NP-hard, there is a heuristic algorithm that scales well and provides near-optimal solutions[29].

Once sensors are deployed and are generating intrusion alarms, we can further leverage TVA for alarm correlation and prioritization. This requires mapping alarms to their corresponding elements (exploits) in the residual attack graph. This in turn requires representing alarms in a common format, using alarm identifiers that match the identifiers (e.g., from vulnerability databases) used in the attack graph model.

In this regard, specifications such as Intrusion Detection Message Exchange Format[30] (IDMEF) or the ArcSight[31] event log format define data formats for information sharing between intrusion detection systems and TVA. For example, one implementation option is the IDMEF plugin[32] for the popular intrusion detection system Snort[33]. This plugin allows Snort to output alerts in the IDMEF message format. Data exchanges in IDMEF are in XML, with the format being enforced through a formal schema.

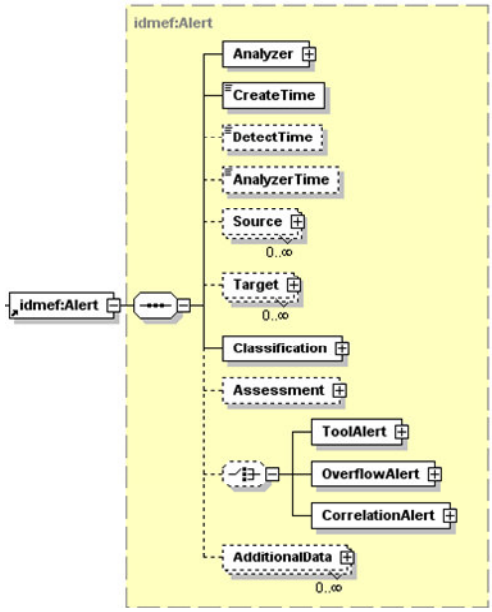

Fig. 23. IDMEF alert schema. This specification provides a standard way of information sharing between intrusion detection systems and TVA.

Figure 23 shows the XML schema for an IDMEF alert. The IDMEF model is intended to represent alerts in an unambiguous fashion, while explicitly assuming that alert information is heterogeneous. Alerts from different tools may

have varying amounts and types of information about an event, which the IDMEF data model is designed to accommodate. For TVA, the critical data are source and target (attacker and victim) network addresses, and an alarm identifier that can be mapped to a vulnerability in the TVA model. In IDMEF, these are supported by the *Source*, *Target*, and *Classification* elements (respectively).

5.2 Attack Prediction and Response

When intrusion alarms are generated, TVA attack graphs provide the necessary context for correlating and prioritizing them. First, we can place a high priority on alarms that lie on vulnerability paths through our network. We can prioritize them even further based on their graph distance to given critical assets. In other words, events that are very close to critical assets (in terms of next attack steps) should be given higher priority.

This kind of attack graph analysis is highly precise, taking all relevant facts into account. We determine not only whether a host is vulnerable to a given attack, but also whether the attacker can traverse through firewalls to reach the host's vulnerable port, and whether that attack could lead to subsequent network compromise. Our prioritization thus also serves as an advanced form of false-alarm reduction, e.g., restriction to alarms along critical paths.

It is important to model network vulnerability as we do. Multi-step alarm correlation that does not take real network vulnerabilities into account is limited[34]. Pre-computing vulnerability-based attack graphs in advance of attack has the additional advantage of rapid correlation, i.e., faster than an intrusion detection system can generate them[35,36].

Further, the predictive capabilities of TVA attack graphs allow us to correlate intrusion alarms based on attack causality. A set of seemingly isolated events may in fact be shown as multiple steps of incremental network penetration. Also, the context provided by these attack graphs allows us to predict missed events (false negatives), helping to mitigate inaccuracies in our intrusion detection systems[35].

To illustrate some of these ideas, consider Figure 24. This is the same residual attack graph as in Figure 21, with relevant protection domains expanded to show additional details. This attack graph provides considerable insight for correlating and prioritizing any alarms generated for this network, and for responding to these potential attacks.

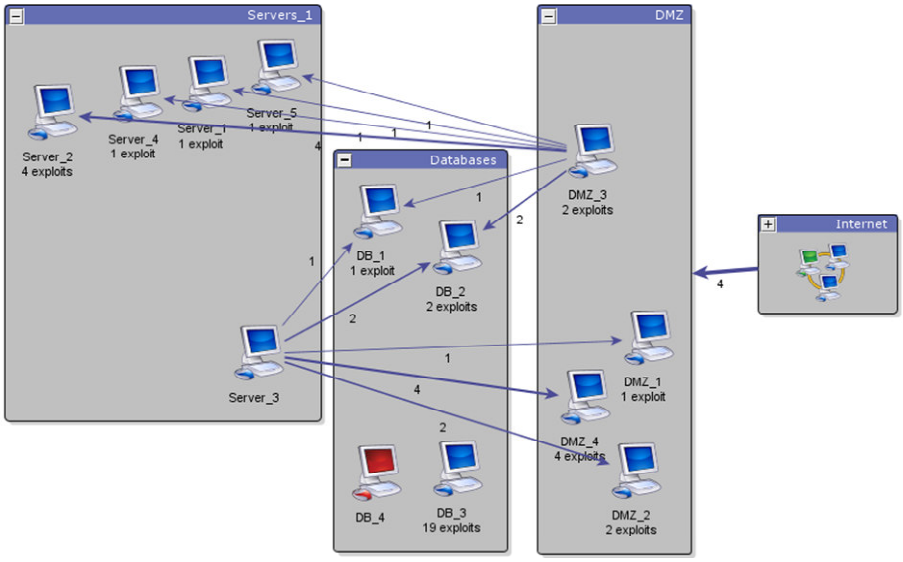

Fig. 24. Attack prediction and response. The attack graph provides the context needed for correlating and prioritizing intrusion alarms, and for predicting next possible attack steps.

For example, suppose an alarm is raised for an attack between two machines in the DMZ, say, from DMZ_1 to DMZ_2. From just a single alarm in the DMZ, we might wait before responding. On the other hand, if an alarm is raised from Internet into DMZ, followed by an alarm within the DMZ, it is a much stronger indicator that the attack may be a real security breach. Remember that false alarms are common with intrusion detection, and erroneously blocking traffic in response to false alarms is a denial of service.

From just a single alarm in the DMZ, another approach might be to block traffic from DMZ_3 to DB_1 and DB_2. We limit this to the vulnerable port(s) on DB_1 and DB_2 only, so that any non-vulnerable services on those machines could remain unblocked. We might then keep traffic from DMZ_3 into Servers_1 machines unblocked, since those machines are one less attack step (i.e., three steps) from critical machine DB_4. In other words, we could still wait to see if an alarm is raised from the DMZ into the Servers_1, at which point we block the vulnerable baths from Servers_1 to Databases.

An even more aggressive response to a single alarm in the DMZ is to block all outgoing traffic from the DMZ to vulnerable services in Servers_1 and Databases. One could also surmise that the alarm in the DMZ is follow-on from a missed intrusion from the Internet into the DMZ. This could guide further

investigation into traffic logs into the DMZ, looking for missed attacks, especially against the four vulnerable paths into the DMZ.

If an attack was detected within Servers_1 (e.g., from Server_1 to Server_2), a similar set of responses is indicated. As a precaution, one could block traffic from Server_3 to vulnerable ports on DB_1 and DB_2. But blocking traffic from Server_3 into the DMZ is less indicated, because it is leading away from the critical Databases domain. Similarly, any alerts from Server_3 into the DMZ are lower priority, especially if they are not against vulnerable DMZ services.

Thus TVA gives a range of reasonable responses, ranked by severity or actual likelihood of attack. Here, severity is in terms of lying on critical vulnerability paths, especially close to critical assets, and likelihood is increased by causal correlation of alerts. Multiple options are available that allow us to fine tune responses as potential attacks unfold, based on proactive response plans.

6 Summary

TVA attack graphs map all the potential paths of vulnerability, showing how attackers can penetrate through a network. TVA identifies critical vulnerabilities and provides strategies for protection of critical network assets. This allows us to take a much more proactive stance, hardening the network before attacks occur, handling intrusion detection more effectively, and responding appropriately to attacks.

TVA models the network configuration, including software, their vulnerabilities, and connectivity to vulnerable services. It then matches the network configuration against a database of modeled attacker exploits for simulating multi-step attack penetration. During simulation, the attack graph can be constrained according to user-defined attack scenarios. From the resulting attack graphs, TVA computes recommendations for optimal network hardening. It also provides sophisticated visualization capabilities for interactive attack graph exploration and what-if analysis. TVA attack graphs support a number of metrics that quantify overall network security, e.g., for trending or comparative analyses.

Further, by mapping TVA attack paths to the network topology, we can deploy intrusion detection sensors to cover all paths using the minimum number of sensors. TVA attack graphs then provide the necessary context for correlating and prioritizing intrusion alerts, based on known paths of vulnerability through the network. Standardization of alert data formats and models facilitates integration between TVA and intrusion detection systems.

By mapping intrusion alarms to the TVA attack graph, we can correlate alarms into multi-step attacks, and prioritize alarms based on distance from critical network assets. Further, through knowledge of network vulnerability paths, we can formulate best options responding to attacks. Overall, TVA offers powerful capabilities for proactive network defense, transforming raw security data into actionable intelligence.

7 Acknowledgments

This material is based upon work supported by Homeland Security Advanced Research Projects Agency under the contract FA8750-05-C-0212 administered by the Air Force Research Laboratory/Rome; by Air Force Research Laboratory/Rome under the contract FA8750-06-C-0246; by Federal Aviation Administration under the contract DTFAWA-04-P-00278/0001; by Air Force Office of Scientific Research under grant FA9550-07-1-0527 and FA9550-08-1-0157; and by the National Science Foundation under grants CT-0716567, CT-0627493, and IIS-0430402. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring organizations.

8 References

1. S. Jajodia, S. Noel, B. O'Berry, "Topological Analysis of Network Attack Vulnerability," in *Managing Cyber Threats: Issues, Approaches and Challenges*, V. Kumar, J. Srivastava, A. Lazarevic (eds.), Kluwer Academic Publisher, 2005.
2. S. Jajodia, S. Noel, "Topological Vulnerability Analysis: A Powerful New Approach for Network Attack Prevention, Detection, and Response," *Indian Statistical Institute Monograph Series*, World Scientific Press, 2007.
3. R. Deraison, *Nessus*, http://www.nessus.org, last retrieved June 2008.
4. R. Ritchey, B. O'Berry, S. Noel, "Representing TCP/IP Connectivity for Topological Analysis of Network Security," in *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC)*, 2002.
5. eEye Digital Security, *Retina Network Security Scanner*, http://www.eeye.com/html/Products/Retina/index.html, last retrieved July 2008.
6. Foundstone, *FoundScan*, http://www.foundstone.com/us/index.asp, last retrieved September 2008.
7. MITRE, *CVE – Common Vulnerabilities and Exposures*, http://cve.mitre.org/, last retrieved October 2008.
8. Security Focus, *Bugtraq Vulnerabilities*, http://www.securityfocus.com/vulnerabilities, last retrieved October 2008.
9. Centennial Software, *Discovery Asset Management*, last retrieved September 2008.

10. Symantec Corporation, *Symantec DeepSight Threat Management System*, https://tms.symantec.com/Default.aspx, last retrieved August 2008.
11. L. Wang, S. Noel, S. Jajodia, "Minimum-Cost Network Hardening Using Attack Graphs," *Computer Communications*, 29(18), 3812-3824, 2006.
12. D. Zerkle, K. Levitt, "Netkuang – A Multi-Host Configuration Vulnerability Checker," in *Proceedings of the 6th USENIX Unix Security Symposium*, 1996.
13. R. Ritchey, P. Ammann, "Using Model Checking to Analyze Network Vulnerabilities," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2000.
14. L. Swiler, C. Phillips, D. Ellis, S. Chakerian, "Computer-Attack Graph Generation Tool," in *Proceedings of the DARPA Information Survivability Conference & Exposition II*, 2001.
15. O. Sheyner, J. Haines, S. Jha, R. Lippmann, J. Wing, "Automated Generation and Analysis of Attack Graphs," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.
16. P. Ammann, D. Wijesekera, S. Kaushik, "Scalable, Graph-Based Network Vulnerability Analysis," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, 2002.
17. S. Noel, J. Jajodia, "Understanding Complex Network Attack Graphs through Clustered Adjacency Matrices," in *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC)*, 2005.
18. R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, R. Cunningham, "Validating and Restoring Defense in Depth Using Attack Graphs," in *Proceedings of the MILCOM Military Communications Conference*, 2006.
19. S. Noel, S. Jajodia, "Managing Attack Graph Complexity through Visual Hierarchical Aggregation," in *Proceedings of the Workshop on Visualization and Data Mining for Computer Security (VizSec)*, 2004.
20. W. Li, *An Approach to Graph-Based Modeling of Network Exploitations*, PhD dissertation, Department of Computer Science, Mississippi State University, 2005.
22. S. O'Hare, S. Noel, K. Prole, "A Graph-Theoretic Visualization Approach to Network Risk Analysis," in *Proceedings of the Workshop on Visualization for Computer Security (VizSec)*, 2008.
21. S. Noel, M. Jacobs, P. Kalapa. S. Jajodia, "Multiple Coordinated Views for Network Attack Graphs," in *Proceedings of the Workshop on Visualization for Computer Security (VizSec)*, 2005.
23. A. Jaquith, *Security Metrics: Replacing Fear, Uncertainty, and Doubt*, Addison Wesley, 2007.
24. *The Systems Security Engineering Capability Maturity Model*, http://www.sse-cmm.org/index.html, last retrieved November 2008.
25. M. Swanson, N. Bartol, J. Sabato, J Hash, L. Graffo, *Security Metrics Guide for Information Technology Systems*, Technical Report 800-55, National Institute of Standards and Technology, 2003.
26. G. Stoneburner, C. Hayden, A Feringa, *Engineering Principles for Information Technology Security*, Technical Report 800-27 (Rev A), National Institute of Standards and Technology, 2004.
27. Forum of Incident Response and Security Teams (FIRST), *Common Vulnerability Scoring System (CVSS)*, http://www.first.org/cvss/, last retrieved June 2008.
28. L. Wang, A. Singhal, S. Jajodia, "Toward Measuring Network Security using Attack Graphs," in *Proceedings of the ACM Workshop on Quality of Protection*, 2007.

29. S. Noel, S. Jajodia, "Optimal IDS Sensor Placement and Alert Prioritization Using Attack Graphs," *Journal of Network and Systems Management*, 2008.
30. Internet Engineering Task Force, *The Intrusion Detection Message Exchange Format (IDMEF)*, http://www.ietf.org/rfc/rfc4765.txt, last retrieved November 2008.
31. ArcSight, *Enterprise Security Management*, http://www.arcsight.com/, last retrieved October 2008.
32. SourceForge, *Snort IDMEF Plugin*, http://sourceforge.net/projects/snort-idmef, last retrieved July 2008.
33. Sourcefire, *Snort – The De Facto Standard for Intrusion Detection/Prevention*, http://www.snort.org/, last retrieved September 2008.
34. P. Ning, Y. Cui, D. Reeves, "Constructing Attack Scenarios through Correlation of Intrusion Alerts," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2002.
35. S. Noel, E. Robertson, S. Jajodia, "Correlating Intrusion Events and Building Attack Scenarios through Attack Graph Distances," in *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC)*, 2004.
36. L. Wang, A. Liu, S. Jajodia, "Using Attack Graphs for Correlating, Hypothesizing, and Predicting Network Intrusion Alerts," *Computer Communications*, 29(15), 2006.