

# An Interactive Attack Graph Cascade and Reachability Display

L. Williams, R. Lippmann, and K. Ingols

**Abstract** Attack graphs for large enterprise networks improve security by revealing critical paths used by adversaries to capture network assets. Even with simplification, current attack graph displays are complex and difficult to relate to the underlying physical networks. We have developed a new interactive tool intended to provide a simplified and more intuitive understanding of key weaknesses discovered by attack graph analysis. Separate treemaps are used to display host groups in each subnet and hosts within each treemap are grouped based on reachability, attacker privilege level, and prerequisites. Users position subnets themselves to reflect their own intuitive grasp of network topology. Users can also single-step the attack graph to successively add edges that cascade to show how attackers progress through a network and learn what vulnerabilities or trust relationships allow critical steps. Finally, an integrated reachability display demonstrates how filtering devices affect host-to-host network reachability and influence attacker actions. This display scales to networks with thousands of hosts and many subnets. Rapid interactivity has been achieved because of an efficient C++ computation engine (a program named NetSPA) that performs attack graph and reachability computations, while a Java application manages the display and user interface.

## 1 Introduction

Attack graphs have been proposed by many researchers as a way to model adversary behavior, identify critical weaknesses, and suggest network changes to improve network security. Researchers and commercial companies have recently developed differing approaches to generating attack graphs (RedSeal, 2007; Skybox, 2007; Ingols et al., 2006; Noel and Jajodia, 2005; Ou et al., 2005). A review of past

---

L. Williams, R. Lippmann, and K. Ingols  
MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02173, USA, e-mail: LCWILL@LL.MIT.EDU, LIPPMANN@LL.MIT.EDU, KWI@LL.MIT.EDU

research is available in (Lippmann and Ingols, 2005a). Attack graphs are constructed by starting an adversary at a given network location and, using information about the network topology and host vulnerabilities, examining how the attacker can progressively compromise vulnerable hosts that are reachable from already compromised hosts. Vulnerability scanners and analyses of filtering performed by firewalls and routers are used to obtain information about host vulnerabilities and to determine host-to-host reachability in a network. Almost all approaches have a method of generating recommendations to patch critical vulnerabilities or make firewalls more restrictive. In addition, most of the existing implementations provide some type of attack graph display. However, the abstract nature of attack graphs has proven to be a serious practical weakness in creating an effective display.

The rest of this paper describes an interactive tool that was designed to overcome the limitations of existing techniques. The following section gives an overview of related work. The next sections present initial design goals, some display types that were considered, and examples of our display for a small test network and a larger field trial network. This is followed by a discussion of future work. The final section concludes.

## 2 Related Work

### *2.1 Limitations of Existing Approaches*

Previous attempts at effectively displaying attack graphs suffer from several fundamental limitations. First, hosts and other network assets are often positioned in a way that is unrelated to normal network diagrams drawn by system administrators. For example, typical network structures such as subnets, virtual local area networks (VLANS), and other fully connected domains are often not easy to delineate and, even when hosts in subnets are collocated, subnets are positioned in locations that are widely divergent from those that are intuitive to system administrators. Second, when an attempt is made to display a complete attack graph at once, this often creates a confusing tangle of edges that is difficult to interpret. Critical steps that allow an attacker to progress between subnets, exploit a new trust relationship, or jump into a network using a client-side attack may be missed in the explosion of edges that occurs after an attacker reaches the network interior. Finally, host-to-host reachability is not usually displayed even though this is a key factor in explaining why critical attack steps are possible and determining which filtering devices allow reachability and could be modified to prevent these attack steps.

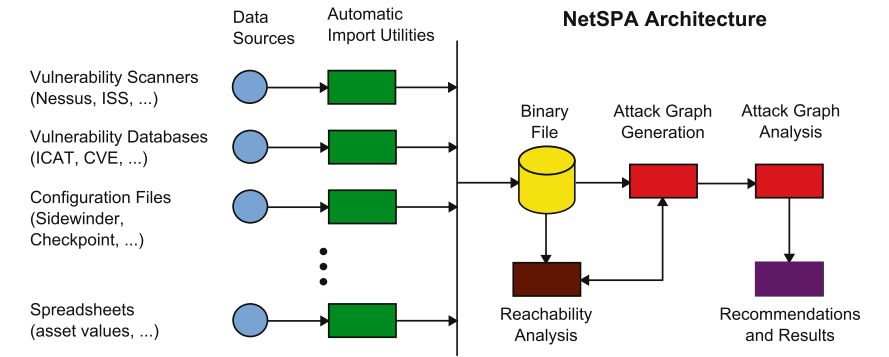
These three limitations initially resulted from the use of automatic graph layout algorithms available in the dot application from the Graphviz software package (Graphviz, 2007; Gansner et al., 1993). Representing hosts as separate nodes and exploitation of vulnerabilities as separate edges often leads to complex graphs that are difficult to display, navigate, and interpret. A number of researchers have suggested methods to simplify these types of attack graph displays (e.g. Ingols et al.,

2006; Noel and Jajodia, 2004, 2005; Swiler et al., 2001) by grouping similar hosts together and representing grouped hosts by single nodes, and by using hierarchical displays. These approaches still result in complex attack graphs that are difficult for untrained system administrators to relate to the underlying analyzed network.

## 2.2 *NetSPA System*

In previous research we described an efficient approach to generating a new type of attack graph called a multiple-prerequisite (MP) graph that scales well to large enterprise networks. Descriptions of the NetSPA tool that generates MP graphs are available in (Ingols et al., 2006; Lippmann et al., 2006; Lippmann and Ingols, 2005b). NetSPA's network model supposes that an individual host possesses one or more interfaces each of which has a listening address. These interfaces also have zero or more open ports which accept connections from other hosts. Interfaces may have rules that dictate how network traffic may flow to, and through, the interface and its host. A port has zero or more vulnerability instances that may be exploitable by an attacker. Each host interface is connected to a link, representing some combination of hubs and switches connecting a set of interfaces together. An attacker is able to obtain one of four access levels on a host using a particular vulnerability: "root" or administrator access, "user" or guest access, "DoS" or denial-of-service, or "other," indicating a loss of confidentiality and/or integrity. The combination of a host and an access level is an attacker state. A state may provide the attacker with zero or more credentials; vulnerability instances may require zero or more of them. Currently, it is assumed that an attacker obtains a host's reachability if "root" or "user" access is achieved. Reachability and credentials serve as prerequisites for exploitation of a vulnerability instance. Our concept of "credential" is any information used for access control such as a password or a private key. Besides providing access to a host, a vulnerability is characterized by whether it can be exploited remotely from a different host or only locally from the vulnerable host.

An MP graph is limited in size because it contains at most one node for each vulnerability instance, host state, reachability group, and credential. Reachability groups represent collections of hosts that are treated identically by all firewalls. These are determined automatically by firewall rule analysis. Edges are contentless and three node types explicitly represent prerequisites of all attacks. State nodes represent an attacker's level of access on a particular host. Outbound edges from state nodes point to the prerequisites they are able to provide to an attacker. Prerequisite nodes represent either a reachability group or a credential. Outbound edges from prerequisite nodes point to the vulnerability instances that require the prerequisite for successful exploitation. Credentials can be used to model many types of trust relationships such as shared passwords used to administer many hosts or automated remote host control via SSH that requires only root access on an administration machine. Vulnerability instance nodes represent a particular vulnerability on a specific port. Outbound edges from vulnerability instance nodes point to the single state



**Fig. 1** System architecture of NetSPA tool

that the attacker can reach by exploiting the vulnerability. These three node types in turn define the sole ordering of paths in the graph: a state provides prerequisites, which allow exploitation of vulnerability instances, which provide more states to the attacker.

The NetSPA system is comprised of several software components. The importer, written in PERL, is responsible for reading in raw data such as Nessus scans, fire-wall rulesets, and NVD database records NVD, 2007, and converting the data into a custom binary file format for later use. A small C program acts as a vulnerability classifier and is designed to identify a vulnerability’s locality (remote or local access) and effect (whether root, user, DoS, or other privilege level is provided). It uses a pattern-matching algorithm that has been trained on a sample vulnerability data set. The classifier was built using the freely available LNKNet tool (Lippmann et al., 1993). The engine, written in C++, is responsible for computing reachability, generating attack graphs, and analyzing the graphs to generate recommendations. It reads the network model from the custom binary file generated by the importer. The block diagram in Fig. 1 gives an overview of the design of the NetSPA system.

Our MP graphs are currently displayed using the Graphviz dot application (Graphviz, 2007; Gansner et al., 1993). These graphs are fast to compute but difficult to interpret for all but the smallest networks. We have developed a simple algorithm to “collapse” many graph nodes together (Ingols et al., 2006), but the simplified graph is still complex and hard for a human to parse. As a result, the current display is of little practical use for nontrivial networks. This motivated our search for a better attack graph display.

### 3 Technical Approach

Attack graphs are usually displayed as node-link graphs. An excellent review of 15 different general approaches to displaying node-link graphs is available in Munzner (2006). We explored many of these using open-source or trial versions of tools and

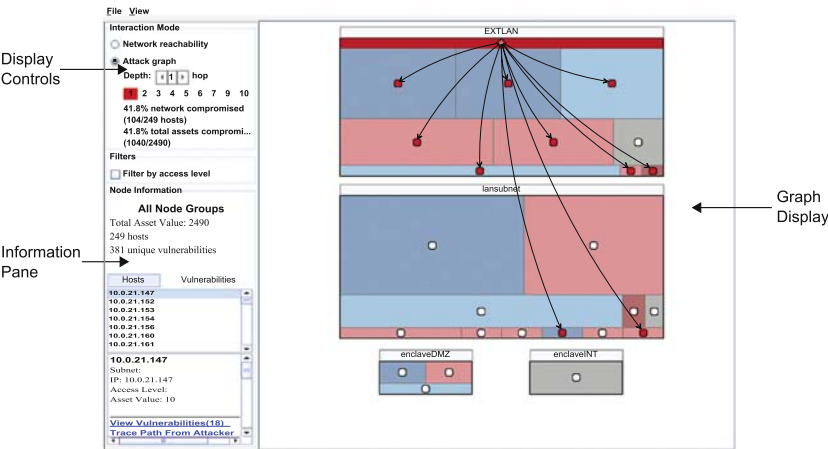
became convinced that displaying attack graphs requires a custom approach. Hand-drawn graphs are too time-consuming. The Graphviz dot tool (Graphviz, 2007) and force-directed approaches lead to excessively complex displays unrelated to the underlying network structure. Techniques to expand and collapse parts of large graphs such as space trees (Plaisant et al., 2002) and hyperbolic trees (Lamping et al., 1995) cause global context to be lost when part of a network is expanded and, as a result, are difficult to follow. Treemaps (Johnson and Shneiderman, 1991) are excellent when summarizing data for a small set of hosts, but they do not represent a network's hierarchical structure well. Finally, multilevel cell matrices (van Ham, 2003; Noel and Jajodia, 2005) are difficult for system administrators to interpret and relate to actual networks.

### ***3.1 Design Goals***

The most important design goal for a custom display that these previous approaches lack is the ability to highlight and explain critical attack steps where attackers progress between subnets, exploit a new trust relationship, or jump into a network using a client-side attack. This capability justifies and explains the automatic recommendations made by the existing NetSPA tool and makes it more likely that these recommendations will be understood and followed by system administrators. In addition, we desired a display where all hosts that a system administrator would expect to be grouped together, such as those in a subnet or VLAN, are also positioned together. We also wanted to allow system administrators to manually position subnets to reflect an intuitive topology and to be able to perform “what-if” experiments and visually see the effect of following recommendations, installing patches, and making firewall rules more restrictive. Finally, we wanted system administrators to be able to explore reachability so they could understand which hosts are exposed through firewalls and other filtering devices. Reachability is too complex to analyze by hand, especially when firewalls include hundreds of filtering and network address translation (NAT) rules and a visual reachability display greatly simplifies a system administrator's task when trying to understand the security of large networks.

### ***3.2 Initial System Design***

Our current display presents an MP attack graph in a readable and concise fashion while preserving much of the essential information. Important features of the nodes are conveyed by grouping, size, and color, while other attributes and edge information are initially hidden and can be displayed on demand. This approach is inspired by the semantic substrate displays described in Shneiderman and Aris (2006). The interface also includes a set of display controls and an information pane which shows additional data about hosts and vulnerabilities. A screenshot of the



**Fig. 2** Interface for attack graph display, with relevant components labeled

entire interface is shown in Fig. 2. The key components are indicated by the text labels.

Our attack graph cascade display attempts to simplify the layout of the MP graph. It shows only the collapsed state nodes from the simplified MP graph, omitting the prerequisite and vulnerability instance nodes. All state nodes attached to the same NetSPA link are grouped together into a rectangular area that typically represents all hosts in a subnet. Initially, subnet groups are successively placed across the display from top to bottom, but they can easily be repositioned and resized to form a more intuitive layout that reflects the conceptual network topology. Each subnet group is labeled by name, and its state nodes are further grouped into collapsed subgroups corresponding to the collapsed state nodes from the MP graph. The inner subgroups are colored according to the level of access of their representative states, and the relative size of each is proportional to the number of hosts it comprises. The subgroups are laid out inside of their respective subnet groups according to the space-filling strip treemap algorithm (Johnson and Shneiderman, 1991). This layout method generally produces reasonable dimensions for the rectangular areas, thus making the subgroups easily discernible.

The user interface is based upon two separate modes of interaction, which determine the meaning of the edges that are drawn between nodes. The edges in the graph display are hidden by default, and each mode enables the user to interactively show or hide a subset of edges. The first mode allows exploration of general reachability within the network. For any particular state node, there is the option to display either incoming or outgoing reachability and the corresponding edges are drawn to all nodes that can reach or be reached from the selected node. Direct interaction with the displayed nodes is supported, and each node provides a context menu containing options for hiding and revealing reachability links. The second mode offers a view of the attack graph, and the edges indicate the shortest attack paths between

sets of nodes. In this attack graph mode, the edges can be incrementally displayed in sets, where each set corresponds to one attacker hop or step. Each hop represents adding to the attack graph links to all hosts that can be compromised from currently compromised hosts. On the first hop, only vulnerable hosts that are reachable from the attacker starting location(s) can be compromised. On successive hops, links to vulnerable hosts reachable from all previously compromised hosts are added to the attack graph. A set of controls is exposed in the side panel that allows the depth of the attack graph to be incremented (and decremented) in this way. We will explore a few simple example networks in the next section to illustrate the features of this new display.

In addition to providing controls for interaction with the display, the accompanying side panel contains an information pane for viewing data that is not graphically represented. This set of information includes hosts represented by each collapsed state group, vulnerabilities present on each host, host IP addresses and asset values, and vulnerability types and descriptions. The informational display is updated whenever a particular state node is selected.

The new display was written in Java and it needed to interface with the NetSPA C++ computation engine to extract information about the network model. We developed a set of bindings between the Java and C++ code using the SWIG toolkit (SWIG, 2007). The Java visualizer is able to load the NetSPA engine as a shared library and drive it programmatically. Because performance in interfacing between Java and C++ is somewhat sluggish, we have taken pains to minimize data transfer requirements, performing analysis on the C++ side or Java side as needed.

The engine and importer required very few changes to support the new Java visualization tool. The importer carries forward some extra data about vulnerabilities which had previously been discarded, and the engine has a few additional methods to aggregate data on the C++ side, minimizing the quantity of calls necessary between C++ and Java. The visualization tool queries the NetSPA library for data on the network model, and it can also request and retrieve reachability computations, collapsed multiple-prerequisite graphs, and recommendations.

3.3 Example Network Results

The first example network shown in Fig. 3 consists of three hosts on an internal network and an external host on the other side of the firewall. The attacker starts on the external network at Host A. The firewall only allows communication between

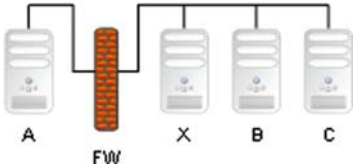


Fig. 3 Example network 1

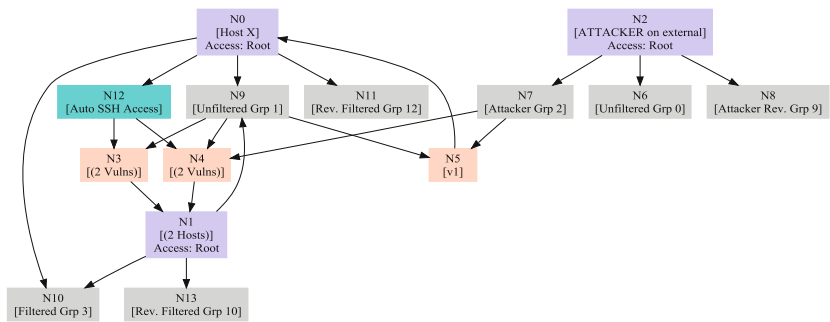


Fig. 4 Original dot display of MP attack graph for network 1

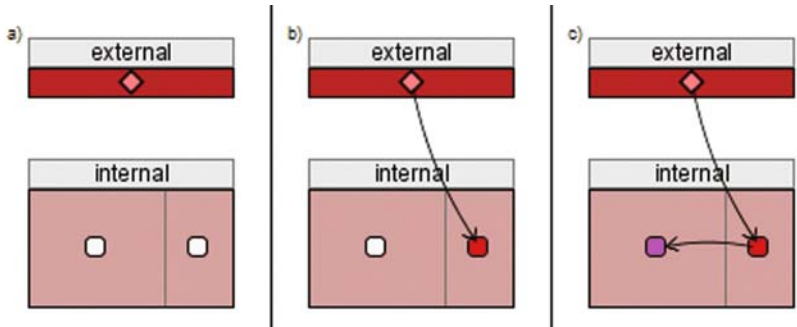


Fig. 5 Simplified display of MP attack graph for example network 1 (a) Before the attack starts, (b) after 1 attack hop or step, and (c) after 2 attack hops

Host X and the external network. Host X is an administrator machine and uses SSH to control Hosts B and C as root. The MP graph models a trust relationship between the administrator and hosts B and C that allows a user with root privileges on the administrator X to automatically log into hosts B and C as root using SSH. The administrator can also be reached via SSH from outside the firewall and there is an SSH vulnerability that allows root level compromise.

Figure 4 shows the MP graph in the original Graphviz dot format. It is relatively small and uncomplicated due to the simplicity of the network. However, the graph does not clearly convey the attacker’s possible paths and it does not at all depict the underlying network structure. The display in Fig. 5a, on the other hand, shows the two distinct subnets (internal and external) and the states contained in each. The pink diamond in the external subnet group represents the attacker’s starting location. The two white squares in the other group represent the collapsed state groups for the internal subnet – one contains both Hosts B and C, and the other contains only Host X. The subgroups are colored pink, indicating root access level for these states. Selecting a node brings up all of this information, such as access level, hosts that make up the group, and exploitable vulnerabilities from the group, in the separate



Fig. 6 Example network 2

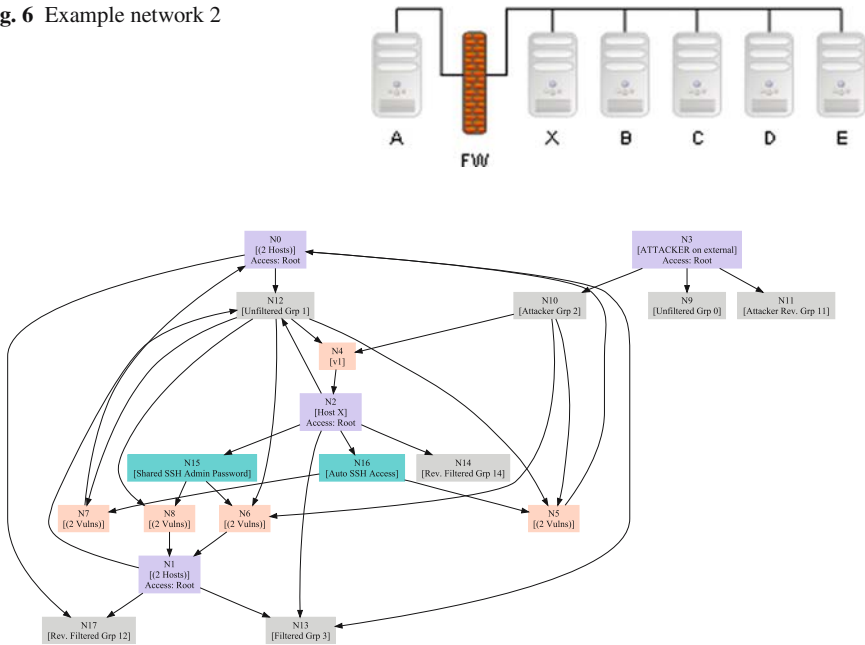
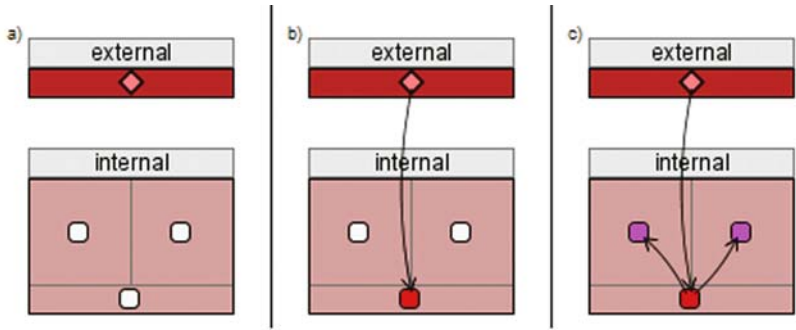


Fig. 7 Original Graphviz dot display of MP attack graph for example network 2

detail pane. Figure 5b,c show the progression of the attack graph. On the first hop, the attacker is able to go through the firewall and compromise Host X. From this point, the attacker can gain access to the remaining hosts and compromises them on the second hop.

Figure 6 presents a slightly more complicated network. It is similar to the first example network, with the addition of two workstation hosts. An administrative password is required to SSH into Hosts D and E as root. Host X possesses the necessary password. It is assumed that an attacker with root privileges on host X can capture the shared password used to administer hosts D and E and this password is modeled as another prerequisite in the MP graph. The remotely exploitable SSH vulnerability on host X has been patched, but there is a web server client-side vulnerability on host X which is the only host allowed to visit web sites in the external network through the firewall. This client-side attack replaces the SSH exploit as the first step of the attack graph in this example.

The Graphviz dot diagram for example network 2 (Fig. 7) shows that the MP graph becomes significantly more complex, even though there is a minimal addition of two hosts and one prerequisite relationship. The alternative display (Fig. 8a) remains just as simple, with only one additional collapsed state group representing root access on Hosts D and E. The layout of the internal subnet group has changed to accommodate the extra subgroup, but the relative sizes have been preserved – the upper two subgroups, each containing two hosts, are twice as big as the lower subgroup, which encloses the single Host X. As illustrated by Fig. 8b, the



**Fig. 8** Simplified display of MP attack graph for example network 2 (a) before the attack starts, (b) after 1 attack hop or step, and (c) after 2 attack hops

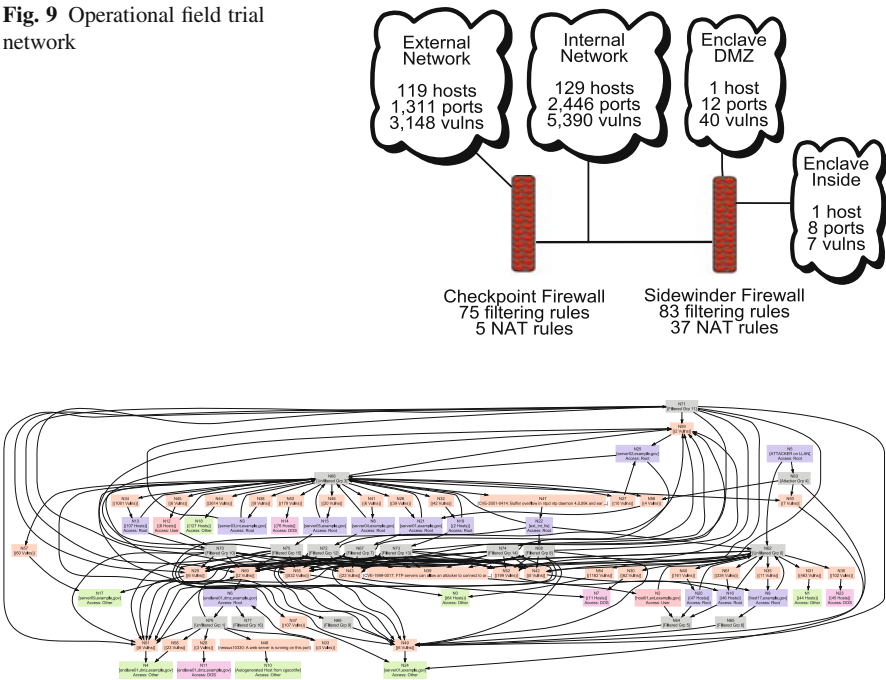
first step in the attack graph is identical to the previous example, and the administrator host is compromised using a client-side attack instead of an SSH server attack. Since the administrator host contains the two prerequisites required to gain access to the remaining hosts, these groups are both compromised in the next step shown in Fig. 8c. In this display, the attacker’s progression through the network is succinctly summarized by exposing the shortest, worst-case attack paths. Consequently, an immediate assessment of an attacker’s ability to penetrate a network can be made without having to deal with firewall rulesets and the complex relationships allowing reachability between hosts.

As was mentioned in the previous section, the display also provides a reachability mode. In this mode of interaction, the incoming and outgoing links for a given node indicate which groups can reach and be reached by that particular group of host states. Reachability between hosts is computed independent of vulnerabilities that are present on those hosts, thereby presenting a different view of the network. Due to the simplicity of the two example networks introduced above, the corresponding reachability displays are trivial and rather uninteresting. This functionality will be illustrated in the next section using a more substantial network graph.

**3.4 Field Trial Results**

To determine how well the new cascade display scales to a larger number of hosts, it was tested on a small operational network, shown in Fig. 9. The network has 250 hosts, 3,777 ports, and 8,585 vulnerability instances. The MP attack graph, presented in Fig. 10, contains 80 nodes and 190 edges. The resulting tangle of edges makes the graph virtually impossible to interpret, and renders it essentially useless. The simplified display of the graph in Fig. 11a contains only 21 collapsed state nodes partitioned into four subnet groups. As mentioned before, the color of the groups indicates the level of access: pink is for root access, dark blue for DoS, light blue for other, and gray for no access (meaning these hosts have no

**Fig. 9** Operational field trial network

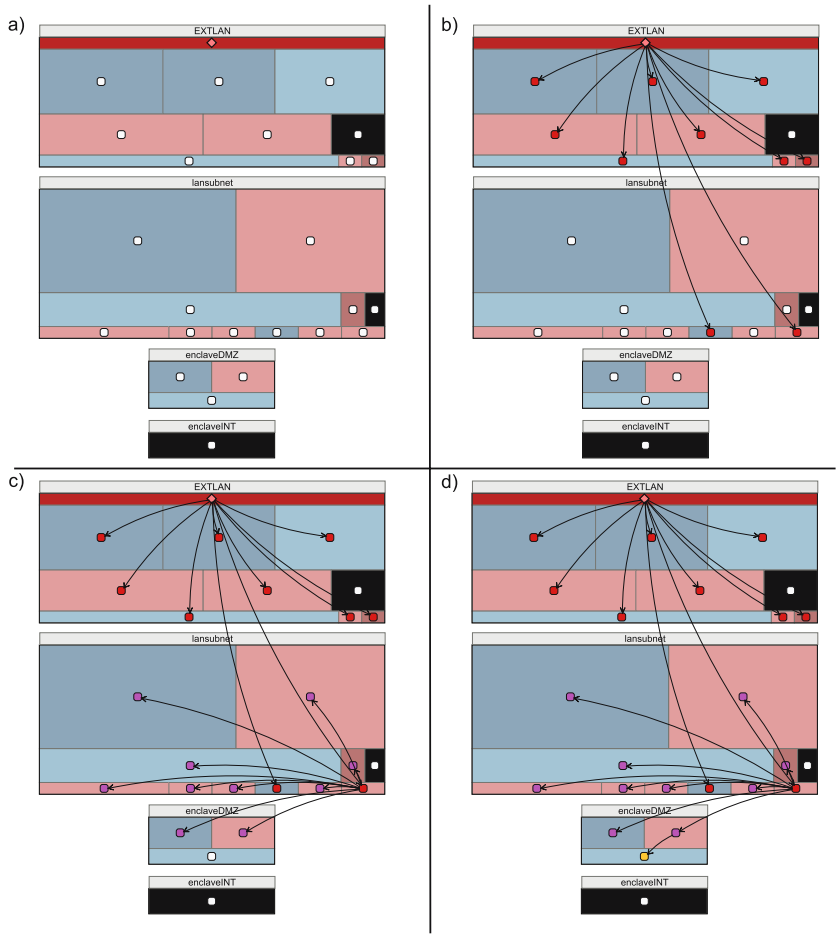


**Fig. 10** Original Graphviz dot display of MP attack graph for field trial network

exploitable vulnerabilities). The red area represents the attacker’s starting location in the external network. Subnets in this display were positioned manually to represent the normal hierarchical relation used by system administrators.

The first step in the attack graph is depicted in Fig. 11b. It is immediately evident that the attacker gains access to most of the external network on the first hop, with the exception of a small group of protected hosts. More importantly, the two edges that cross over to the internal network demonstrate that the attacker can gain control of these inside hosts from the outset. Vulnerabilities that allow these transition edges are critical to the security of the internal network. Figure 11c shows the second attacker hop, where the remainder of the internal network is compromised, as well as two states on the enclave DMZ host. The third and final step in Fig. 11d reveals the complete compromise of the host in the DMZ. The enclave inside network has no incident edges and is therefore fully protected.

Figure 12 gives a sample view of general reachability within the trial network. It displays the reachability for the node in the middle right gray subgroup of the external subnet. In Fig. 12a, the incoming reachability is displayed, and the links from all of the nodes in the two larger subgroups imply that the target node in the external subnet is reachable from each of these groups. Figure 12b shows the outgoing reachability for the target node, which is able to reach each of the nodes it points to. None of these links are present in the attack graph display since the hosts represented by this node do not possess any vulnerabilities that are exploitable by the attacker.

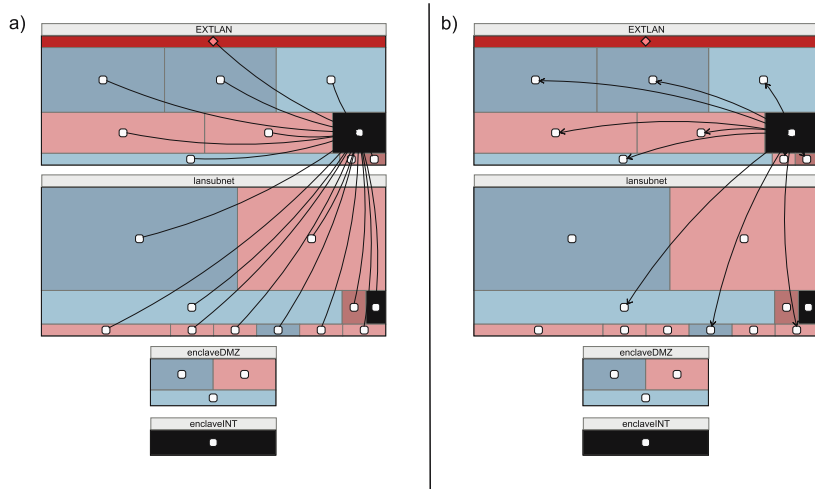


**Fig. 11** Simplified display of MP attack graph for field trial network (a) before the attack starts, (b) after 1 attack hop or step, (c) after 2 hops, and (d) after 3 hops

It is also interesting to note that the set of links representing outgoing reachability is not necessarily identical to that representing incoming reachability. This example illustrates the simplicity with which complex routing and filtering rules can be graphically represented and also underscores the need for having separate views for the attack graph and general reachability.

## 4 Future Work

We have used this initial attack graph cascade display to illustrate reachability and attack graphs for example networks and a field trial network. Security administrators and researchers who have been exposed to this new display are able to understand



**Fig. 12** Reachability display for a single node in field trial network (a) incoming reachability for one node, (b) outgoing reachability for one node

how the display relates to the analyzed network, determine which hosts can be compromised, and also identify those vulnerabilities that are most critical in enabling attackers to progress through networks. This was not true for our original Graphviz displays.

Several enhancements are currently under development for the current attack graph cascade display. In the current implementation, a starting subnet for the attacker can be selected and NetSPA automatically determines a set of worst-case IP addresses for that subnet. Support will be added for manually selecting a single starting IP address or a range of addresses. The display will also be extended to support some limited “what-if?” experiments. A list of recommendations generated by NetSPA will be presented with the option to select and apply one or more. Users can then see the effect of patching vulnerabilities and changing firewall rules when the attack graph is regenerated and the display updated to indicate which nodes are no longer accessible as a result of following a recommendation. A related feature will add support for introducing zero-day vulnerabilities into the network, allowing the user to view the changes in the regenerated graph. These will be vulnerabilities in server or client software versions that a user can propose to proactively plan for future attacks. In addition to the original MP attack graph, NetSPA also generates summary attack graph metrics that will be provided as part of the interactive display. These include calculation of the total assets captured vs. the number of attack hops, the number of unique exploits required by an attacker, and an estimate of the cost of development for the exploits required. Plots of assets captured vs. these measures of attacker effort will supplement the information provided in the attack graph display by presenting a different view of the data. There will also be support for importing historical data about a network and computing metrics representing a past state of

the network. This older data could be presented alongside or overlaid on the current plots, showing how the security of the network has changed over time.

Another goal in this development effort is to ensure that the display application is intuitive and user-friendly. To measure the usability of the graphical interface, user tests will be conducted with system administrators and blue team members. These assessments will aid in identifying additional features that may need to be included or existing functionality that may need to be modified. The resulting feedback will enable us to develop a simpler and more efficient interface for the display. We also plan to apply the cascade display to very large networks with many subnets and determine if some type of user-generated hierarchy or other approach is necessary to display such networks in a simple and uncluttered way.

## 5 Conclusions

We have developed an efficient attack graph cascade display that is intuitive, produces a compact representation, highlights critical attack steps that lead into new network areas, and displays both attack graph and reachability information. Because the display is easy to understand and interact with, a system administrator can confirm the accuracy of the analysis and the validity of the recommendations. This may encourage the practical use and acceptance of attack graphs as tools to improve network security.

This new display improves over past research and commercial attack graph displays in a few critical areas. First, hosts in subnets are presented in close proximity using a treemap. This makes it easy to associate hosts in the graph with hosts in actual subnets. Hosts in each treemap are automatically grouped based on level of compromise, how the hosts are treated by firewalls, trust relationships the hosts participate in, and prerequisites required to compromise hosts. These groupings provide visual indications of the network security and greatly simplify the display. The color of each group indicates the level of compromise and the size of the treemap rectangle for a group indicates the number of hosts in that group. At least one commercial company (RedSeal, 2007) uses a treemap, but it appears to be used to provide an overall indication of the number of vulnerabilities for all hosts in a network and not as part of an attack graph display. Other approaches to generating attack graphs collapse hosts into groups (e.g. Swiler et al., 2001; Noel and Jajodia, 2004), but our approach automatically forms collapsed groups, results in large computational savings, and leads to groupings and layouts that can be readily understood. A second improvement is that users position subnet treemaps themselves to represent an intuitive notion of the network hierarchy. This requires some extra user effort and may not scale when there are hundreds of subnets, but it results in understandable topologies. A third improvement is that we display all hosts that can be compromised from a specified attacker starting point. Some other displays (e.g. Skybox, 2007) require users to specify both an attacker starting point and a target destination and only show the shortest path between these two points. The cascade display provides an

overall picture of the attack graph and shows how patching specific critical vulnerabilities can protect not just one target, but many. A fourth improvement is that an interface between the Java cascade display software and the NetSPA C++ engine allows rapid interaction with and regeneration of the attack graph display. Computation times for computing attack graphs with 1,000's of hosts are typically less than a few seconds (Ingols et al., 2006) and this allows interactive display and analysis. Computation times for computing recommendations and showing the result of recommendations are similar. Finally, we display reachability information as well as attack graphs by using the same network display and operating it in two separate modes, where edges have a different meaning in each mode.

**Acknowledgements** We would like to thank Seth Webster and Rob Cunningham for interaction and feedback on initial display designs.

## References

- Gansner ER, Koutsofios E, North SC, Vo KP (1993) A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, vol. 19, no. 3, 214–230
- Graphviz (2007) Graph visualization software. <http://www.graphviz.org>. Accessed 6 Sept 2007
- Ingols K, Lippmann R, Piwowarski K (2006) Practical attack graph generation for network defense. *Proceedings Computer Security Applications Conference (ACSAC)*, 121–130
- Johnson B, Shneiderman B (1991) Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the Second Conference on Visualization '91*, GM Nielson and L Rosenblum, Eds. *IEEE Visualization*, IEEE Computer Society Press, Los Alamitos, CA, 284–291
- Lamping J, Rao R, Pirolli P (1995) A focus + context technique based on hyperbolic geometry for visualizing large hierarchies. *Proceedings of ACM CHI '95 Conference*, 401–408
- Lippmann R, Ingols K (2005a) An annotated review of past papers on attack graphs. MIT Lincoln Laboratory, Lexington, MA, Technical Report, 2005, ESC-TR-2005-054
- Lippmann R, Ingols K (2005b) Evaluating and strengthening enterprise network security using attack graphs PR-IA-2. MIT Lincoln Laboratory Project Report, 12 August 2005
- Lippmann R, Kukulich L, Singer E (1993) LNKnet: neural network, machine-learning, and statistical software for pattern classification. *Lincoln Laboratory Journal*, vol. 6, no. 2, 249–268
- Lippmann R, Ingols K, Scott C, Piwowarski K, Kratkiewicz K, Cunningham R (2006) Validating and restoring defense in depth using attack graphs, *MILCOM 2006*, Washington, DC
- Munzner T (2006) 15 Views of a node-link graph: an infovis portfolio. <http://www.cs.ubc.ca/~tmm/talks.html>. Accessed 6 Sept 2007
- Noel S, Jajodia S (2004) Managing attack graph complexity through visual hierarchical aggregation. *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*. New York, NY, USA: ACM Press, New York, 109–118
- Noel S, Jajodia S (2005) Understanding complex network attack graphs through clustered adjacency matrices. *Proceedings Computer Security Applications Conference (ACSAC)*, 160–169
- NVD (2007) National Vulnerability Database <http://nvd.nist.gov/>. Accessed 6 Sept 2007
- Ou X, Govindavajhala S, Appel AW (2005) Mulval: a logic-based network security analyzer. In *Proceedings of the 14th Usenix Security Symposium*, 2005, 113–128

- Plaisant C, Grosjean J, Bederson B (2002) Spacetree: supporting exploration in large node link tree, design evolution and empirical evaluation. In INFOVIS 2002, IEEE Symposium on Information Visualization, 57–64
- RedSeal Systems Inc. (2007) <http://www.redseal.net/>. Accessed 6 Sept 2007
- Shneiderman B, Aris A (2006) Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, 733–740
- Skybox Security Inc. (2007) <http://www.skyboxsecurity.com>. Accessed 6 Sept 2007
- SWIG (2007) <http://www.swig.org>. Accessed 6 Sept 2007
- Swiler LP, Phillips C, Ellis D, Chakerian S (2001) Computer attack graph generation tool, Proceedings of the Second DARPA Information Survivability Conference & Exposition (DISCEX II), Los Alamitos, CA. *IEEE Computer Society*, vol. 2, 307–321
- van Ham F (2003) Using multilevel call matrices in large software projects. *Proceedings Information Visualization (INFOVIZ)*, 227–232