Improving Attack Graph Visualization through Data Reduction and Attack Grouping ⋆

John Homer[1], Ashok Varikuti[1], Xinming Ou[1], and Miles A. McQueen[2]

[1] Kansas State University, USA
[2] Idaho National Laboratory, USA

Abstract. Various tools exist to analyze enterprise network systems and to produce attack graphs detailing how attackers might penetrate into the system. These attack graphs, however, are often complex and difficult to comprehend fully, and a human user may find it problematic to reach appropriate configuration decisions. This paper presents methodologies that can 1) automatically identify portions of an attack graph that do not help a user to understand the core security problems and so can be trimmed, and 2) automatically group similar attack steps as virtual nodes in a model of the network topology, to immediately increase the understandability of the data. We believe both methods are important steps toward improving visualization of attack graphs to make them more useful in configuration management for large enterprise networks. We implemented our methods using one of the existing attack-graph toolkits. Initial experimentation shows that the proposed approaches can 1) significantly reduce the complexity of attack graphs by trimming a large portion of the graph that is not needed for a user to understand the security problem, and 2) significantly increase the accessibility and understandability of the data presented in the attack graph by clearly showing, within a generated visualization of the network topology, the number and type of potential attacks to which each host is exposed.

Key words: attack graph, attack graph visualization, dominator, graph clustering, network security analysis

1 Introduction

Attack graphs have been developed to aid in identification and correction of misconfigurations in enterprise network systems, by providing a visual representation of potential attack paths [1–8]. Much work has already been done in the generation of attack graphs, producing more efficient techniques for building them [6, 7]. Attack graphs, however, are difficult for a human to utilize effectively because of their complexity [9–11]. Even a network of moderate size can have dozens of possible attack paths, overwhelming a human user with the amount of information presented. It is not easy for a human to determine from the information in the attack graph which configuration settings should be changed to best address the identified security problems. Without a clear understanding of the existing security problems, it is difficult for a human user to evaluate possible configuration changes and to verify that optimal changes are made.

Previous works have introduced improvements in the visualization of attack paths and the overall presentation of attack graph data. Noel, *et al.* suggested that complexity can be reduced through the use of protection domains to represent groups of machines with unrestricted interconnectivity [9, 10]. Lippmann, *et al.* introduced visualization approaches to emphasize critical attack steps while clearly showing host-to-host reachability [11].

In this paper, we show that by utilizing the logical semantics of an attack graph, one can 1) distinguish attack steps based on their usefulness for a human to quickly understand the core security problems in an enterprise network, and trim those that do not contribute much for this purpose; 2) identify attack steps that share similar semantics, and thus can be grouped and presented as a single virtual node. These techniques can further improve the visualization of attack graphs to make them more useful in practice.

For our implementation, we use the MulVAL attack graph tool suite [7, 12], which provides reasonable performance and scalability for enterprise networks of a realistic size. MulVAL produces complete logical attack graphs, which are easily mapped back to a visualization of the network topology, based on the input data.

⋆ This work is partially supported by the National Science Foundation under Grant No. 0716665, and U.S. Department of Energy. Any opinions, findings and conclusions or recomendations expressed in this paper are those of the authors and do not necessarily reflect the views of the U.S. government agencies.

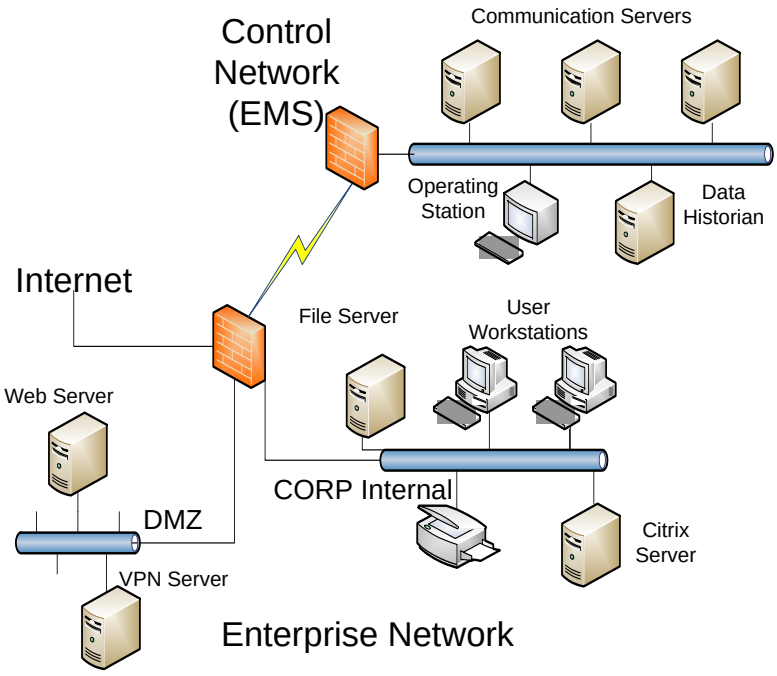

Fig. 1. An example enterprise network

Figure 1 depicts an example enterprise network that is based on a real (and much bigger) system; we will return to this example throughout the paper. The network includes three subnets: a DMZ (Demilitarized Zone), an internal subnet, and an EMS (Energy Management System) subnet, which is a control-system network for power grids. In this example, we will assume that host-grouping has already been applied, based on similar configurations; the workStation node, for example, might be an abstracted grouping of one hundred workstation machines with comparable setups. Both the web server and the VPN server are directly accessible from the Internet. The web server can access the file server through the NFS file-sharing protocol; the VPN server is allowed access to all hosts in the internal subnet (but not the EMS subnet). Outside access to the EMS subnet is only allowed from the Citrix server in the internal subnet, and even then only to the data historian. In this example, we assume that the attacker's goal is to gain privileges to execute code on the commServer. From the commServer, an attacker could send commands to physical facilities such as power-generating turbines, which can cause grave damage to critical infrastructures.

A visualization of the MulVAL attack graph is shown in Figure 2, identifying a large number of potential attack paths in this network. This visualization is produced by mapping the full MulVAL logical attack graph to the network subnet topology, using GraphViz to construct the image, and applying clustering techniques similar to Noel *et al.*'s approach [10]. The black solid lines represent connectivity between subnets and gateways (router, firewall, *et al.*). Machines in the same subnet (represented as a rectangular cluster) have unrestricted access to each other. The red dotted lines indicate attack propagation paths as mapped from the full logical attack graph. This visualization omits a large amount of information from the original full attack graph, such as pre- and postconditions for each attack step. We believe a simple visualization of attack paths directly on the network topology will be useful in practice, since it relates the information conveyed by the attack graph to the concrete entities in an enterprise network, and a system administrator will likely find it easier to understand than the full attack graph. The full logical attack graph (see Appendix A) contains all of the same information shown in Figure 2, portraying all possible paths by which the commServer could be compromised; the breadth of this information, however, leads to a graph so large and complicated as to be unreadable by a human user.

Even after this simplification, the attack paths in the visualization may still overwhelm a user. In this paper we will focus on how to further reduce the complexity through two techniques. First, we observe that there are a number of

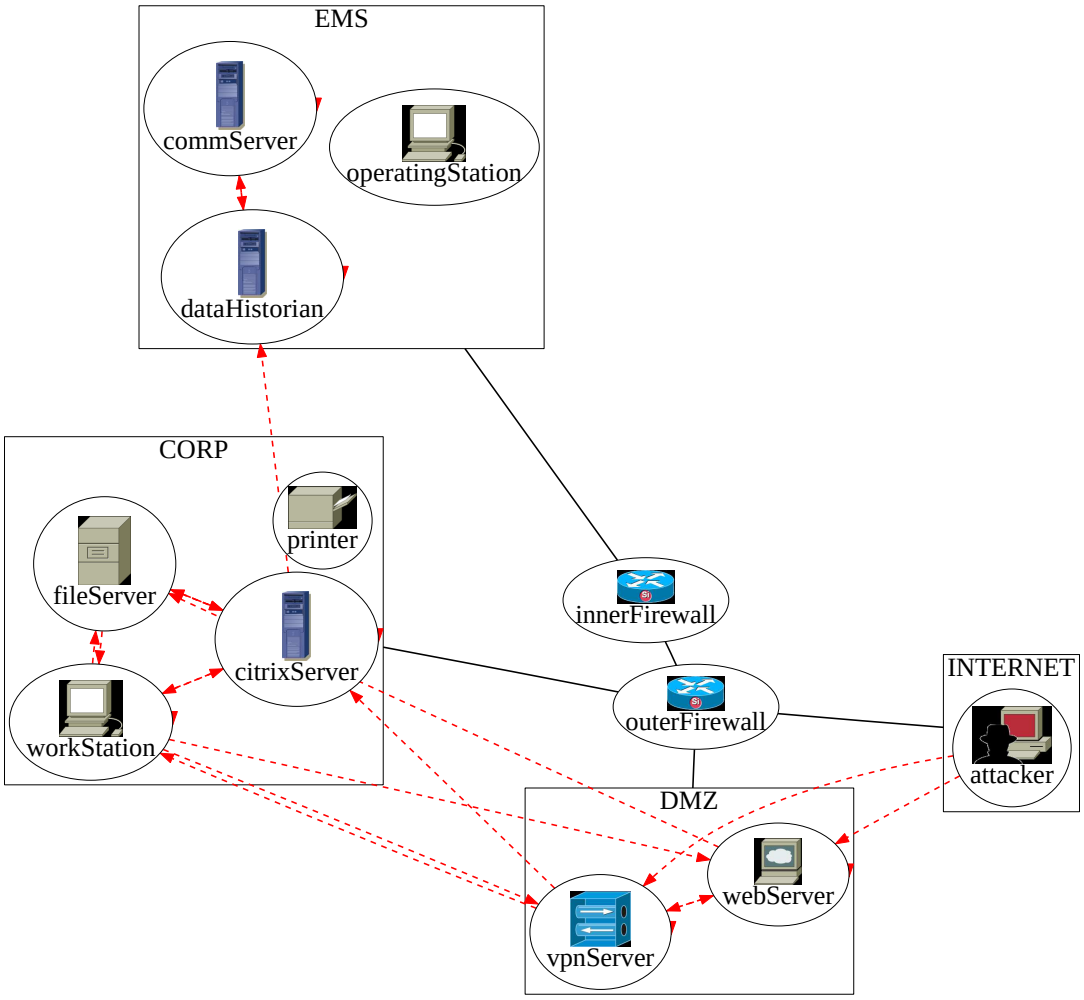

Fig. 2. Attack graph visualization

logically valid attack steps that probably need not be shwon to the user for him/her to understand the security problem. For example, an attacker who has gained privileges on the workStation machine in the internal subnet has opportunity to exploit the webServer in the DMZ. Though possible, this attack step is intuitively unhelpful to understanding the security problem: the attacker would have to compromise DMZ to gain privilege on workStation in the first place. Showing users the attack steps "back" to DMZ does not provide any additional insights. In reality, these attack steps are likely to be useless to the attacker as well. Another example of would be attacking the fileServer from the citrixServer; Since we assume the commServer is the attacker's goal, an attacker with privileges on the citrixServer already has all of the necessary privileges to attack the EMS subnet through the dataHistorian. The user gains nothing useful by knowing that the attacker can further attack the fileServer from the Citrix server. Second, we observe that the complexity of the attack graph does not necessarily reflect complexity in security vulnerabilities. Employing a compromised user account, an attacker can access the citrixServer from the vpnServer, workStation, and fileServer; using a Trojan horse attack, an attacker can gain access to the citrixServer from the fileServer. (The edge from fileServer to citrixServer represents both potential attack steps). Although there are four distinct attack steps that can enable an attacker to

compromise the citrixServer, these attack steps utilize only two distinct exploitations. This fact is obscured in the attack graph by the separate attack steps leading to citrixServer from different host machines.

We believe that the attack graph complexity can be further reduced. In order to make an attack graph a useful tool for configuration management, we identify as a research challenge the need for presenting the security problems expressed by an attack graph in a manner that enables a human user to more quickly grasp the core of the security problem. Our contributions are:

1. We developed an algorithm to identify portions of an attack graph that are not helpful for a user to understand the core security problems, and reduce the amount of data presented to the user by trimming those portions. When the amount of information presented in the attack graph is reduced, we believe that core security problems will be more quickly identifiable from the attack graph.
2. We developed a method to create virtual nodes to represent groupings of similar exploitations. In this approach, each attack step edge leading into a host represents a unique attack on that machine. We believe that this approach will increase the understandability of the attack graph data by more clearly displaying the exploitability of each host.

Our approach to trimming attack steps ensures that all distinct attack paths will be retained in the trimmed attack graph, while removing data not beneficial to the understanding of core security problems. Host-grouping techniques have already been shown to be effective for reducing complexity [6, 9, 10]. We show that further gains can be made in grouping similar exploits from multiple sources, which makes clearly visible the number of exploits available on a given machine and all of the possible sources for each potential exploit. It is easy then to see all attack steps that can be eliminated by resolving the vulnerability enabling a specific exploit.

The trimming algorithm is presented in Section 2. The exploit grouping approach is presented in Section 3. We will discuss related work in Section 4 and conclude with a discussion of future work on these approaches in Section 5.

2 Identifying and Removing "Useless" Attack Steps

In examining the attack graph, we found that many of the attack steps, while valid from a logical point of view, are not helpful for a human user to comprehend the core security problems in the network configurations. They share a common characteristic which is they do not reveal the most important vulnerability in the system since the attacker does not penetrate "deeper" into the enterprise network along those steps. While these steps contain important information that would be useful if one wished to block every possible attack path, they can also be distracting to a human reader and often hides the root causes of the security problems. It is thus beneficial to remove these less useful attack steps from the attack graph so that the security problems become easier to grasp for a human reader.

We refer to attack steps that are not useful for a human reader to understand the underlying security problems as "useless" attack steps. Generally, these "useless" attack steps involve an attack on a machine further from the goal machine than the machine from which the attack is made. In the example described earlier, for instance, one valid attack step enables an attacker with privileges on the workStation to gain privileges on the webServer, but this attack would not bring the attacker any closer to the presumed goal of accessing the commServer.

Our classification of "useful" and "useless" edges is meant to reflect a prioritization of the data contained in the attack graph. We can then provide a simplification of the original attack graph, highlighting attack reachability between hosts to enable a human user to more quickly comprehend fundamental vulnerabilities in the network. It is important to emphasize that the so-called "useless" attack steps are valid and important to consider when determining upon appropriate countermeasures. However, when the user is first presented with the attack graph, understanding these paths is not crucial for understanding overall security threats. It would be more beneficial if the user can quickly understand the core security problems from a simplified attack graph. For example, in Figure 3 the attacker's starting machine is host 1, and his goal is to compromise host 4. There are two paths: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ and $1 \rightarrow 3 \rightarrow 4$. Intuitively the attack step from machine 3 to machine 2 is not useful, since it does not help the attacker to reach his goal. We would like to trim those steps.

The immediately obvious solution for identifying these "useless" attack steps is to implement a breadth-first search algorithm to compute the distance of machine from the goal machine, as measured by the minimum number of inter-host attack steps necessary to reach the goal from that machine. Machine 2 in the above example would have a distance

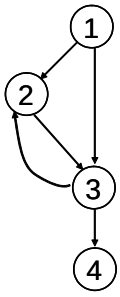

Fig. 3. Example useless attack steps

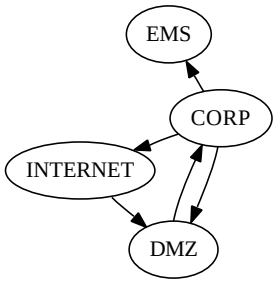

Fig. 4. Subnet graph, built from example in Section 1

of 2 and machine 3 would have a distance of 1. Attack steps that move from a machine to another machine with the same or greater distance (*e.g.* $3 \rightarrow 2$) could then be labelled "useless" and trimmed. While this approach would work in some cases, in many cases useful attack paths would be trimmed. For example, this algorithm would also trim the step $1 \rightarrow 2$, and thus lose the complete attack path $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. Actually only attack paths with the shortest length will be preserved and all other paths will be trimmed. This approach can lose valuable information, especially in cases when an attacker might follow a longer attack path due to ease of exploit or better stealthiness along the longer path.

Another seemingly correct solution is to perform a simple depth-first search and trim the back edges in the DFS tree. However, depending on the order of traversing a node's multiple children, both edge $2 \rightarrow 3$ and $3 \rightarrow 2$ could be back edges in the DFS tree. So this method does not work either.

We have developed a two-level approach to identifying and removing "useless" attack steps. First, we create a directed graph with subnets as nodes and possible inter-subnet attack steps as edges. From this directed graph, we then construct a dominator tree to recognize dominance and post-dominance relationships between subnets with respect to presumed attacker location and his goal. The subnet where the attacker is located will be the source node and the subnet where the goal machine is located will be the sink node. Let d, n, p be vertices in a directed graph. Then d dominates n if every path from the source node to n must go through d. We write d *dom* n for this fact. Also, p post-dominates n if every path from n to the sink node must go through p. We write p *postdom* n for this fact. In the subnet graph of Figure 4, assuming that INTERNET is the source and EMS is the goal, some examples of dominance relationships are DMZ *dom* CORP and CORP *dom* EMS. We also have EMS *postdom* CORP and CORP *postdom* DMZ.

For any two subnets X and Y, we then identify as "useless" all inter-subnet attack steps $X \rightarrow Y$ where Y *dom* X or X *postdom* Y. If Y *dom* X, then an attacker who has gained privileges in subnet X must already have privileges in subnet Y (or the attacker would not have been able to transition to X). If X *postdom* Y, then moving from X to Y will not help the attacker either since he would have to return to X in order to reach his goal. Therefore, any transitions between two hosts in different subnets that fits one or both of these cases is "useless" and will be trimmed from the attack graph. They are distracting for a human reader trying to comprehend other, more enlightening attack paths. In the attack graph shown in Figure 2, every transition from the CORP subnet to the DMZ will be trimmed since DMZ dominates CORP. On the other hand, the transition from CORP to the control network subnet EMS will be retained.

After applying the inter-subnet transition trimming, we then address intra-subnet transitions. An attack step between two machines $A \rightarrow B$ in the same subnet is retained in only two cases. First, if the subnet contains the goal machine, the transition is retained only if B is the goal. Any other transition within this subnet will be trimmed. In reality, an attacker might need to transition to other machines in the same subnet for the purpose of, *eg.* obtaining more computing power. However, these attack steps are not useful for a human to grasp the core security problem. Second, if the subnet does not contain the goal machine, the transition is useful only if B would provide an attacker with access to another subnet that would be deemed useful according to the subnet dominator tree, and even then only if that same access is not available from A. In the attack graph shown in Figure 2, the transition from fileServer to workStation is useless, since the workStation would not provide an attacker with new, useful access; however, the transition from fileServer to citrixServer is useful, since, from citrixServer, an attacker could access the EMS subnet.

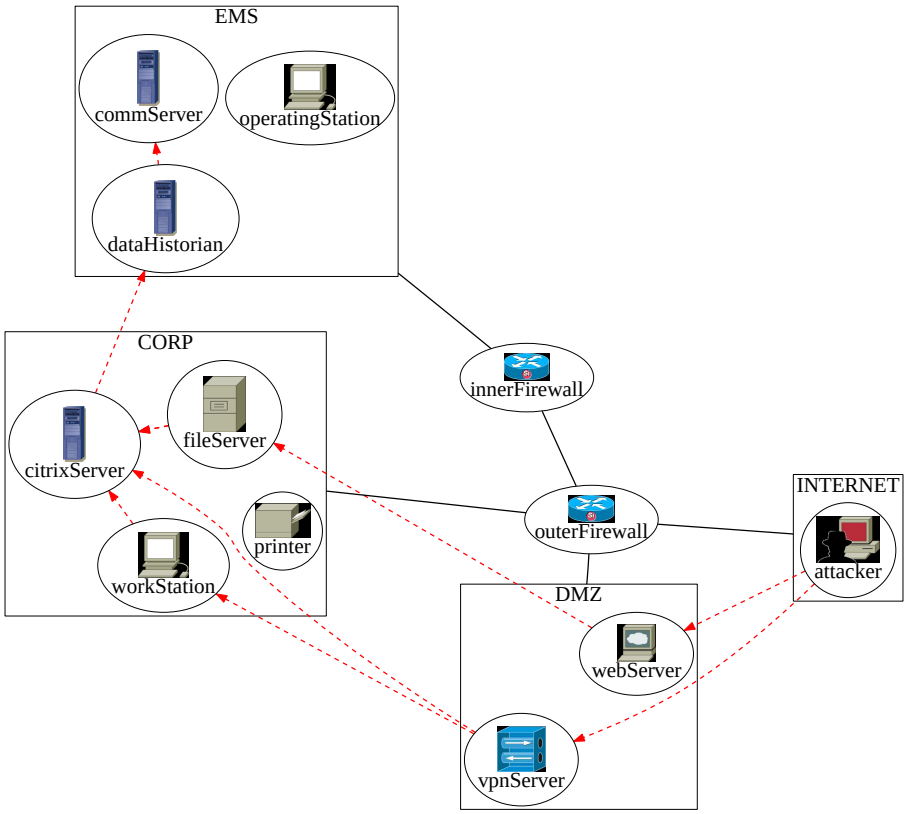

Fig. 5. Attack graph with both inter- and intra-subnet trimming applied

Figure 5 shows the resulting graph after both levels of trimming levels are applied to the sample network. The attack graph now shows three key attack paths to reach the Citrix server, from which subnet EMS is accessible:

$$Internet \rightarrow web\ server \rightarrow file\ server \rightarrow Citrix\ server$$
$$Internet \rightarrow VPN\ server \rightarrow workstation \rightarrow Citrix\ server$$
$$Internet \rightarrow VPN\ server \rightarrow Citrix\ server$$

A careful reader might ask why the second attack attack path is retained, given the existence of a shorter path (3). As mentioned above, shorter attack paths cannot always subsume longer ones, since the exploits along the path may be different.

3 Abstraction of Attack Traces

Even after identifying and removing “useless” attack steps, many edges will likely remain in the visualization. Humans assessing the data presented in the attack graph will benefit from the reduced amount of data, but still face other obstacles to clear and straightforward understanding of the underlying security issues in the current network configuration. One hindrance to easy understanding of attack graph data can be the number of edges directed into a single host machine in the attack graph. In Figure 5, for example, the citrixServer node in the Corp subnet is the destination point of four attack steps (shown in three edges), even after “useless” attack steps have been trimmed from the graph. It is not immediately clear to the user how many different possible exploitations are being represented, and how many sources for exploitations of the citrixServer are repetitions of a single attack type.

Our solution to this difficulty is to create an abstraction of each exploitation with multiple sources, from which only one edge will lead into the exploited node. In this way, it is much easier for a human user to see how many exploitations are possible on a given host and what potential attack steps could be eliminated by resolving the conditions enabling a specific exploitation. Potential exploits with only one source, on the other hand, will be represented by a direct attack step edge between the two machines, to maintain as much simplicity in the graph as possible.

In the attack graph shown in Figure 5, three of the edges that lead to the citrixServer represent different source points but only a single security issue in the network, namely the uncertain reliability of the user with account "ordinaryUser." If this user account is compromised, an attacker could gain access to the citrixServer from any of the three host machines with edges leading to the abstracted exploitation node. By creating a virtual node to represent the existence of this security concern, it is much easier to see now that if the reliability of this user account can be verified, most of the possible attacks leading to citrixServer will be eliminated. Our attack graph visualization will show transitions to an abstracted exploit node as blue lines, while red lines will indicate direct host-to-host attacks as well as attacks from an abstracted exploit node.

The full attack graph, shown in Figure 2, included a number of "useless" attack steps that are removed by the trimming algorithm. The trimming also reduces the number of multiple-source attacks and thus the number of abstract exploit nodes that can be created. For optimal effectiveness, this exploit abstraction technique is applied only to the trimmed attack graph. The final attack graph is shown in Figure 6.

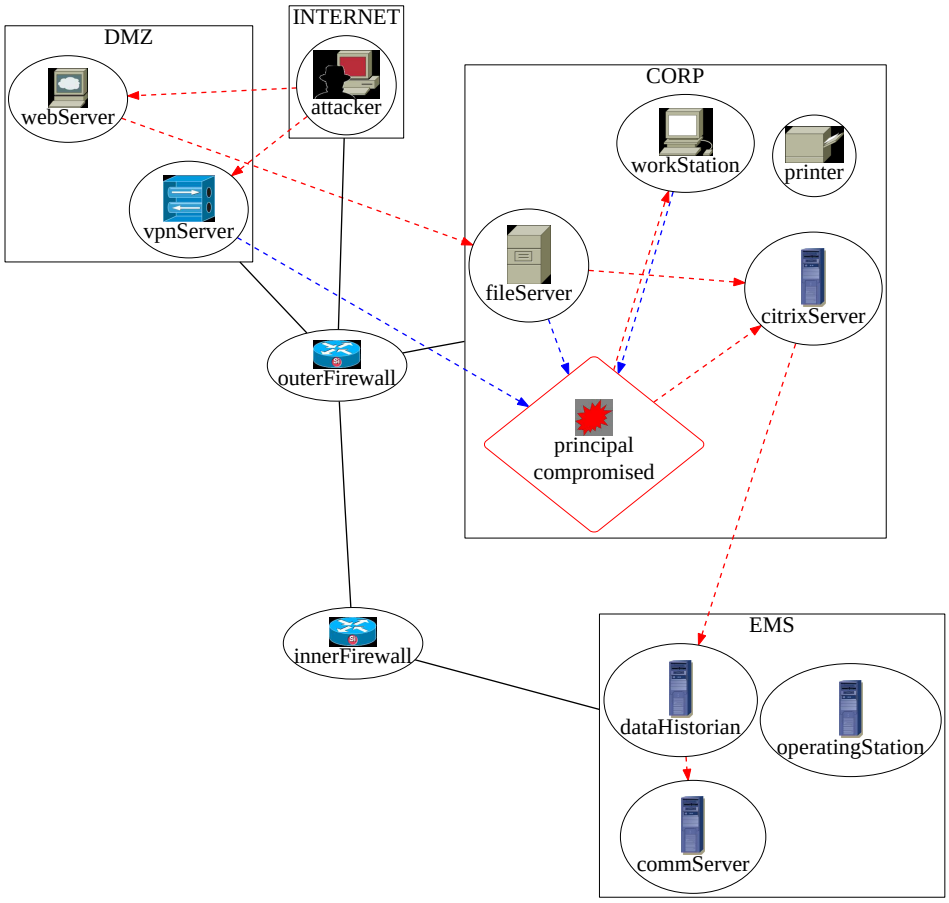

Fig. 6. Attack graph, trimmed, with virtual exploitation nodes

4 Related Work

A number of other previous works addressed the problem of how to use attack graphs to better manage the security of enterprise networks [13–17]. The observations and insights from these previous works helped us develop the approach in this paper, and our work either complements or improves upon them. Our contribution is the development of formal, logic-based approaches to simplifying an attack graph for a human to better understand.

Noel, *et al.* proposed a number of techniques for reducing complexity in attack graphs [10]. We adopted some of the approaches in our work, such as using clustering techniques to show the subnet topology of the network. Our approaches address complementary problems in visualization, namely identification and removal of attack paths that are not useful for a human to better understand the core security problems, and better represent attacks by grouping similar exploits targeted at a single host. Noel, *et al.* also presents a notion of graph trimming, by removing redundant exploits and allowing them to be implicitly conveyed in the graph. However, they do not systematically address how to identify and trim the "useless" attack steps described in this paper.

Lippmann, *et al.* have built on the multiple-prerequisite graphs produced by the NetSPA system with a goal of reducing attack graph complexity [11]. Their visualization employs spatial grouping and color-coding to represent levels of potential compromise. Groups of machines with similar levels of exploitability can then be collapsed, reducing the overall complexity of the graph. Our approach differs in that we do not group machines with similar vulnerabilities, but rather create abstract representations of attacks, with edges leading to the potentially affected machines.

5 Conclusion

We have proposed two techniques for improving visualization of attack graphs — reducing the amount of data by identifying attack steps that are not crucial for a human to quickly understand the core security problems, and grouping similar attacks targeted at a single host to better represent the number and type of security problems. These techniques, in combination with visualization techniques developed by previous researchers, will display attack graphs to a more human-readable manner. This is crucial for using attack graphs to further automate enterprise network security management, since a human can only trust a tool if she/he understands its output. Our techniques will help a human user quickly identify the core security problems in an enterprise network without being overwhelmed by the amount of information contained in the full attack graphs.

References

1. Laura P. Swiler, Cynthia Phillips, David Ellis, and Stefan Chakerian. Computer-attack graph generation tool. In *DARPA Information Survivability Conference and Exposition (DISCEX II'01)*, volume 2, June 2001.
2. Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 254–265, 2002.
3. Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of 9th ACM Conference on Computer and Communications Security*, Washington, DC, November 2002.
4. Sushil Jajodia, Steven Noel, and Brian O'Berry. Topological analysis of network attack vulnerability. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, *Managing Cyber Threats: Issues, Approaches and Challanges*, chapter 5. Kluwer Academic Publisher, 2003.
5. Richard Lippmann and Kyle W. Ingols. An annotated review of past papers on attack graphs. Technical report, MIT Lincoln Laboratory, March 2005.
6. Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical attack graph generation for network defense. In *22nd Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, Florida, December 2006.
7. Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *13th ACM Conference on Computer and Communications Security (CCS)*, pages 336–345, 2006.
8. Wei Li, Rayford B. Vaughn, and Yoginder S. Dandass. An approach to model network exploitations using exploitation graphs. *SIMULATION*, 82(8):523–541, 2006.
9. Steven Noel and Sushil Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 109–118, New York, NY, USA, 2004. ACM Press.
10. Steven Noel, Michael Jacobs, Pramod Kalapa, and Sushil Jajodia. Multiple coordinated views for network attack graphs. In *IEEE Workshop on Visualization for Computer Security (VizSEC 2005)*, 2005.
11. Richard Lippmann, Leevar Williams, and Kyle Ingols. An interactive attack graph cascade and reachability display. In *IEEE Workshop on Visualization for Computer Security (VizSEC 2007)*, 2007.
12. Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. MulVAL: A logic-based network security analyzer. In *14th USENIX Security Symposium*, 2005.
13. Somesh Jha, Oleg Sheyner, and Jeannette M. Wing. Two formal analyses of attack graphs. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, pages 49–63, Nova Scotia, Canada, June 2002.
14. Richard P. Lippmann, Kyle W. Ingols, Chris Scott, Keith Piwowarski, Kendra Kratkiewicz, Michael Artz, and Robert Cunningham. Evaluating and strengthening enterprise network security using attack graphs. Technical Report ESC-TR-2005-064, MIT Lincoln Laboratory, October 2005.
15. Richard Lippmann, Kyle Ingols, Chris Scott, Keith Piwowarski, Kendra Kratkiewicz, Mike Artz, and Robert Cunningham. Validating and restoring defense in depth using attack graphs. In *Military Communications Conference (MILCOM)*, Washington, DC, U.S.A., October 2006.
16. Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke, and Jeannette Wing. Ranking attack graphs. In *Proceedings of Recent Advances in Intrusion Detection (RAID)*, September 2006.
17. Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring network security using attack graphs. In *Third Workshop on Quality of Protection (QoP)*, 2007.
18. Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Trans. Program. Lang. Syst.*, 1(1):121–141, 1979.

A MulVAL Logical Attack Graph

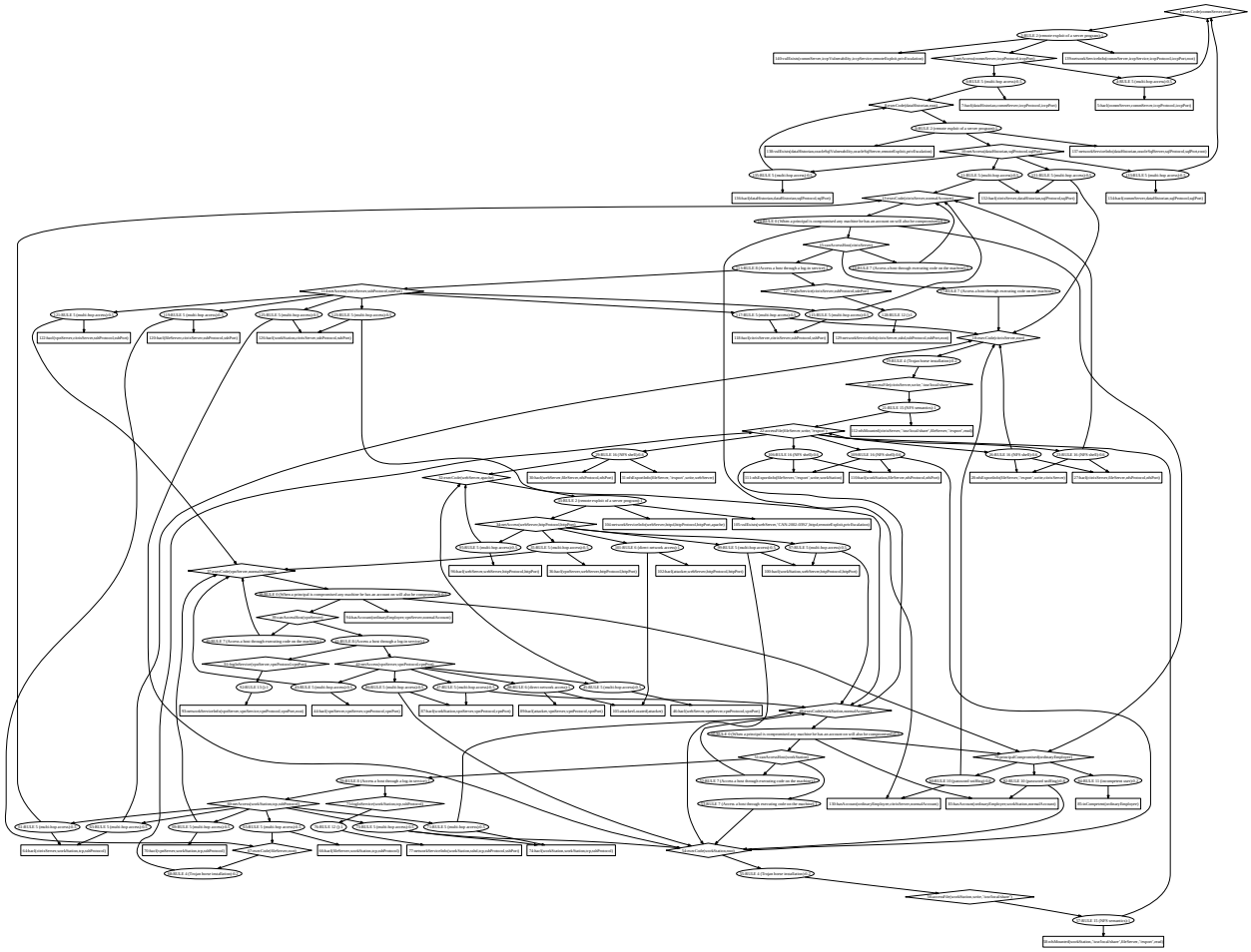

Fig. 7. Full logical attack graph, as generated by MulVAL