Representing TCP/IP Connectivity For Topological Analysis of Network Security

Ronald Ritchey [1]
ritchey_ronald@bah.com

Brian O'Berry
boberry@gmu.edu

Steven Noel
snoel@gmu.edu

Center For Secure Information Systems
George Mason University

Abstract

The individual vulnerabilities of hosts on a network can be combined by an attacker to gain access that would not be possible if the hosts were not interconnected. Currently available tools report vulnerabilities in isolation and in the context of individual hosts in a network. Topological vulnerability analysis (TVA) extends this by searching for sequences of interdependent vulnerabilities, distributed among the various network hosts. Model checking has been applied to the analysis of this problem with some interesting initial result. However previous efforts did not take into account a realistic representation of network connectivity. These models were enough to demonstrate the usefulness of the model checking approach but would not be sufficient to analyze real-world network security problems. This paper presents a modem of network connectivity at multiple levels of the TCP/IP stack appropriate for use in a model checker. With this enhancement, it is possible to represent realistic networks including common network security devices such as firewalls, filtering routers, and switches.

1. Introduction

A common approach used to analyzing network security is to focus on individual vulnerabilities on each host that makes up the network. There are many products designed to assist in this process, including products that scan for vulnerabilities on a host from across a network [7] and products that run on the host being analyzed [5]. These tools can help identify known vulnerabilities on these computers and can produce reports that help security administrators identify steps to reduce the vulnerabilities that exist on each of the scanned systems. This is a reasonable approach as the reduction in vulnerability for the individual hosts on a network is bound to have a positive impact on the security of the network as a whole. However, it does not offer any insight into vulnerabilities caused by interactions of systems on a network.

Previous work used a modeling approach to analyze network security based on the interactions of vulnerabilities within individual hosts and between hosts connected by the network [8]. This work applied model checking technology to analyze a simplified network security model and determine whether the network's security requirements were met or if there was a method that could be used to invalidate any of the requirements. The security requirements were encoded as assertions in the model checker. The model checker was used to determine whether any of these assertions could be proved false. In this case the model checker produced a detailed set of steps that it had used to invalidate the assertion. This set of steps constituted a potential path an attacker might follow to circumvent the security of the network.

While this work was useful for validating the model checking approach, substantial improvements are required to make it a practical tool in analyzing real-world network vulnerabilities. These include the development of automated tools to populate the model, the encoding of a large set of exploitation techniques into the model, and refinements in the model itself to allow it to more accurately represent modern computer networks. Our research center and others are actively working on these issues, but most of the work has concentrated on the first two problems [9]. This paper specifically addresses enhancements to the model described in [8] that support a sufficiently rich representation of connectivity for real-world networks.

A sophisticated depiction of network connectivity is essential to model network vulnerability. A primary means of defending against network attack is the use of firewalls and filtering routers. For a model to produce useable results, the effects of these devices must be represented. In addition, there are certain attacks that operate against the network itself as opposed to targeting a particular host on the network. These also must be represented for the model to be capable of analyz-

[1] This effort sponsored in part by Booz Allen Hamilton

ing real-world network security problems.

This paper extends the previous model checking approach by including representations of network connectivity in the analysis model. It is TCP/IP and Ethernet centric, but the methods used could easily be extended to support alternative networking protocols. A layered approach is used to capture specific vulnerabilities that exist at different layers of TCP/IP. In addition, careful consideration is given to efficiency of representation while still maintaining sufficient fidelity to produce meaningful results on real networks.

In summary, this paper presents extensions to the previous model checking approach in [9] that provides sophisticated representation of network connectivity while not excessively increasing the state-space of the model. This is interesting because it brings us much closer to the ability to produce a tool capable of analyzing the impact that interactions of vulnerabilities have between hosts on real-world networks.

2. The Network Security Model

Our approach, which we call Topological Vulnerability Analysis (TVA), uses a state-based model of network security to discover attack paths. In this section, we provide a short description of how the model is structured.

2.1 Model Elements

There are four major elements that make up the network security model.

- A network of hosts, including the network services, components and configuration details that give rise to vulnerabilities
- Connectivity of the hosts
- Exploits, or attacks, that change the state of the model
- A list of security requirements the model should attempt to validate

Hosts are described by their network services, components and configurations, and the current user privileges obtained by the attacker. Vulnerabilities might arise from obvious problems, such as running an outdated network service like Sendmail [4]. They might also depend on general configuration details, such as operating system version, type of authentication, and password length. Our definition of vulnerability is broad. In this model, vulnerability is any system attribute that can be used as a prerequisite for an exploit. This may seem unbounded, but in reality, because there is a finite number of known exploits each with a finite number of prerequisite vulnerabilities, the total set of vulnerabilities needed by the model is finite. This is important because it allows us to bound the total number of system features we must search for to populate the analysis model.

Connectivity is modeled using a variety of techniques depending upon the TCP/IP layer that is being represented. This is an extension of the previous connectivity model, which used a simple Boolean matrix to represent connectivity. The enhancements presented in the paper allow the model to more accurately represent real-world networks, making its results more relevant. More will be said about this in the following sections.

Exploits are modeled using functions that, given the right circumstances, can cause changes to the state of the model. Exploits are used by the model to affect changes to the security of the hosts under analysis. The quality and quantity of exploits encoded in the model have a direct relationship with the quality of the analysis that can be performed with the model.

Security requirements are represented by invariant statements made about the security of particular hosts on the network (e.g. an attacker can not obtain administrative access to host A).

2.2 Vulnerabilities and Exploits

Vulnerabilities and the methods necessary to exploit them form the core of the model. The technique relies on modeling the network attributes that give rise to vulnerabilities, then analyzing whether the exploits encoded into the model can take advantage of the vulnerabilities to circumvent the network's security. Vulnerabilities come from many sources and are difficult to eliminate because of several factors. For a network to be useful, it must offer services. These services are implemented in software and it is difficult to guarantee that any complex piece of software does not contain some flaws [2]. These flaws frequently translate into security vulnerabilities. In addition, simple network configuration errors can introduce exploitable security flaws.

To break into a network, it is not sufficient to know about the vulnerabilities on the network. You must also have an exploit to take advantage of these vulnerabilities. In addition, before an exploit can be used, its pre-conditions must be met. These pre-conditions may include the set of vulnerabilities that the exploit relies on, sufficient user rights on the target, sufficient user rights on the attacking host, and network connectivity. Results of a successful exploit could include discovering valuable information about the network, elevating user rights, defeating filters, and adding trust relationships among other possible effects. These post-conditions share a common feature; they reduce the security of the network. The concept of exploit pre-conditions and post-conditions is illustrated in Figure 1.

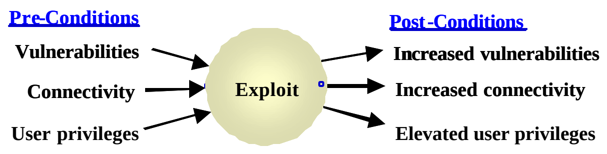


Figure 1 - Exploit Model

By successively employing exploits, an increasing number of vulnerabilities can be added to a host. Given the right set of circumstances, this may result in the complete compromise of the host. If an attacker gains control of a host, it can be used to attack other hosts on the network. It is in this fashion that an attacker can move through a network until they have achieved their final goal. This chaining of exploits to compromise multiple hosts is shown in Figure 2.

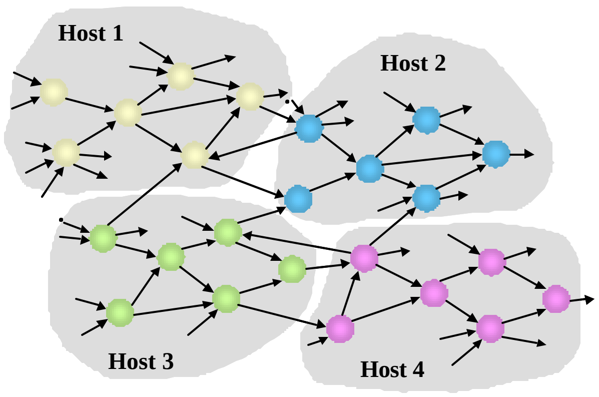


Figure 2 - A Network of Exploits

2.3 Model Checking

Initial versions of TVA relied upon model checkers to act as the analysis engine. Model checkers are attractive because they are designed to handle large state spaces [3] and they efficiently generate counterexamples that, for TVA, correspond to attack paths. Thus they provide a mechanism to avoid custom building the same capabilities into special purpose tools. They are not particularly well suited for this application, though, because the TVA model is monotonic. In it, exploits don't decrease the vulnerability of the network. They always increase it. There is never a need for the analysis engine to back track to come up with a solution. Model checkers on the other hand, must support non-monotonic and other more general problems. This requires them to be substantially more complex than TVA requires. While the work of this paper is not directly affected by the choice of the analysis engine, it is worth noting that a custom application has the potential to substantially outperform a general-purpose model checker.

3. Modeling Link Layer Security

As stated previously, we use a layered approach to modeling network connectivity within TVA. These layers are derived from the structure of the TCP/IP network protocol stack. The lowest layer of the TCP/IP protocol stack is the Link layer, which provides and manages access to the network medium.

At the Link layer, communication can only occur between hosts located on the same network segment. To deliver a TCP/IP packet to a host on the segment, the destination IP address must be resolved into a Link layer address. The protocol that performs this resolution is the Address Resolution Protocol (ARP). ARP-resolved addresses therefore identify hosts that share a common network segment, and we label such connectivity LINK_ARP. This type of connectivity is the prerequisite for many network attacks such as TCP session hijacking.

Another Link layer characteristic relevant to network security is packet sniffing. Sniffing is an activity through which a privileged user can eavesdrop on network traffic. Most network traffic is transmitted unencrypted, and might include usernames and passwords for protocols such as telnet, ftp, rlogin, pop3 and others. An attacker could use this ability to capture the authentication details for a particular user, then use them to impersonate the user on the network.

The ability to sniff network traffic is affected by whether a network is switched or not. Ethernet was developed as a bus-like medium where all hosts connected to a hub shared the network bandwidth. Hubs re-broadcast all received packets to every connected host. With this scheme, all network traffic is visible to every host's network interface card (NIC) connected to the same network segment. Normally, a NIC ignores traffic that is not specifically addressed to it. However, it is possible for privileged users to put NICs in promiscuous mode, allowing them to capture all local traffic. Sniffing on a non-switched network, then, enables an attacker to capture all traffic crossing the local network, whether or not it's addressed to or from the attacker's machine.

Ethernet switches essentially eliminate bandwidth sharing by directing traffic to those hosts specifically addressed in the Link layer frames. This filtering process limits the usefulness of sniffing because the only network traffic that the attacker will see are packets addressed to or leaving from the sniffing host. Note that broadcast packets (packets addressed to all hosts on the same network segment) will be visible to the attacker regardless of whether the network is switched or non-switched.

A TVA program must include switched and non-switched network details to effectively address sniffing attacks. In addition to filtering support, some switches have additional capabilities enabling them to be configured into multiple network segments. Also, mixed environments where hubs are connected to switch ports are common. Therefore, the TVA program must track Link layer connectivity at the host level to distinguish which hosts have such connectivity with each other, and which hosts have "sniff" connectivity with each other. The label LINK_SNIFF is used to designate a host's ability to sniff the traffic of another host.

Figure 3 depicts a generic ARP exploit (a) and a generic SNIFF exploit (b). Note that we show only the connectivity pre-conditions; other pre-conditions, such as the presence of a sniffer program or the attainment of super user access, would be required for a full TVA model. Also, the exploit post-conditions are not labeled because they depend on the exploit specifics (e.g., a password sniffing exploit would have a post-condition that indicated a password had been obtained).

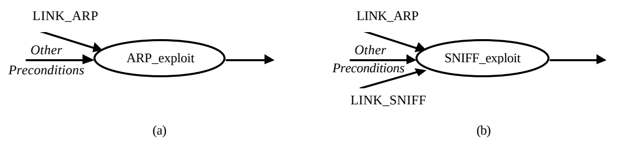


Figure 3 - Generic Link Layer Exploits

The following example shows how these exploits might map to a real network. In Figure 4, Fred and Helen share a hub, so they can sniff each other's communications with Sheila. Because a switch separates Sheila and the hub, Sheila cannot sniff traffic between Fred and Helen. The generic ARP exploit shown in Figure 3(a) can trigger for all three hosts, because all have LINK_ARP connectivity between them. However, the generic SNIFF exploit in Figure 3(b) can only apply to Fred and Helen because Sheila does not have any LINK_SNIFF connectivity.

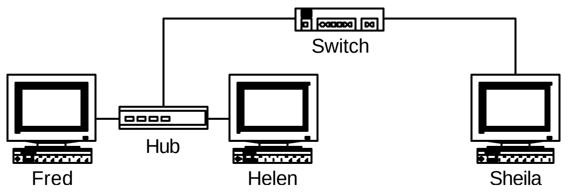


Figure 4 - Mixed Network (Switched & Non-Switched)

4. Modeling Network and Transport Layer Security

The Network layer of TCP/IP provides global addressing and routing of packets between network segments. The source and destination IP addresses of the packet are specified at this layer. The Transport layer controls the flow of data between different host services, which are addressed by a local port number. TCP/IP offers two transport protocols. The Transmission Control Protocol (TCP) is stream oriented and provides a reliable, byte-oriented data flow through control functions that are largely transparent at the Application layer. The User Datagram Protocol (UDP) is record oriented and simply offers the ability to send packets between hosts, so the application must provide control functions if reliable delivery is required.

Most network services communicate via transport protocols, so their packets contain both Network layer (IP) and Transport layer (port) addresses. These address details are commonly used by firewalls to decide whether a packet should be allowed between hosts. They may restrict access based on the IP address and/or port number of either the source or destination host. Capturing this connectivity-limiting firewall behavior at both the Network and Transport layers is therefore critical in analyzing network security.

In [8], connectivity was represented with a simple Boolean matrix, which is sufficient to characterize firewall restrictions based solely on IP address. Transport layer connectivity requires a more complex representation. One possible method is to simply add dimensions to the connectivity matrix for the UDP and TCP port numbers. This may work from an analytical point of view, but it would drastically increase the size of the model to account for the 65,536 port numbers associated with each transport protocol. An easy way to reduce this is to track only the ports on a network that are actually in use. Symbolic labels can be created for each of these in-use ports and the total number of labels would dictate the size of the matrix. While this would greatly reduce the size of the connectivity matrix, it does not address an important requirement of the security analysis problem.

From a security analysis perspective, raw connectivity is not the only pre-condition necessary for an attack to succeed. A vulnerability must also exist in the application that supports the network service. Such a vulnerability is usually specific to the particular application. For example, Microsoft's Internet Information Server (IIS) web server is susceptible to different attacks than the Apache web server [1, 6]. It is much more important to track the specific application details associated with the service than it is to track the port number on which the service runs. Therefore, we name Transport layer connectivity variables after the application that supports the network service (e.g., TRANS_APACHE_1_3_21). Such names can easily be extended to include patch level and other information required to delineate between different exploits' pre-conditions. By avoiding port numbers, this approach also eliminates complications associated with running services on non-standard ports. Web servers, for example, often run on ports other than 80. A port-based approach would require a table to map port numbers to application details. Instead, we collapse the required connectivity information into a single, appropriately named variable as shown in Figure 5.

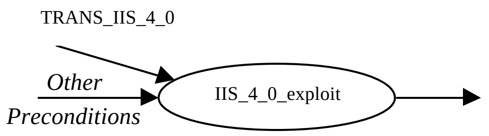
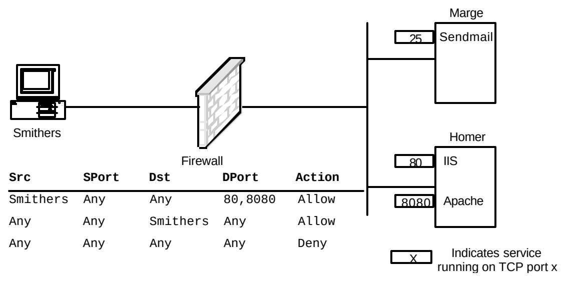


Figure 5 - Example Transport Layer Exploit

Figure 6 shows a small but typical network where a firewall policy limits Transport layer connectivity to some public services. Specifically, external hosts (represented by Smithers) are allowed to connect to internal hosts on ports 80 and 8080 (which would allow Smithers to communicate with the Apache and IIS servers located on Homer), but connections to all other ports (including Sendmail on Marge) are blocked. Homer and Marge, on the other hand, are permitted to any port on Smithers.



Src	SPort	Dst	DPort	Action
Smithers	Any	Any	80,8080	Allow
Any	Any	Smithers	Any	Allow
Any	Any	Any	Any	Deny

Figure 6 - Example Network with Connectivity-Limiting Firewall

As discussed earlier, the actual port numbers are irrelevant for TVA analysis. Rather, the ability of an attacker to launch an exploit is based on whether Transport layer connectivity exists to the service applications themselves. An attacker with access to Smithers, for example, might use exploits against Homer that require TRANS_IIS or TRANS_APACHE connectivity, but couldn't directly launch exploits against Marge's Sendmail service because the firewall blocks access to it. Homer, though, enjoys unrestricted access to Marge. If the attacker on Smithers could first gain control of Homer by exploiting one of the directly accessible services, Marge would be indirectly reachable. Figure 7 shows a possible exploit path from Smithers to Marge, where an IIS exploit first yields the ability to execute programs on Homer. This increase in access level, combined with Homer's Transport layer connectivity to Marge, then allows the attacker to apply a Sendmail exploit against Marge. One of TVA's core capabilities is it's ability to shift the locus of attack in this way, just as a real attacker would, to work around connectivity limiting facilities such as firewall policies.

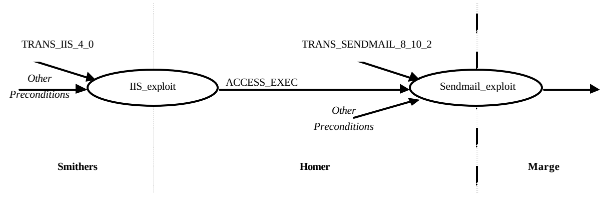


Figure 7 - Example Exploit Path

In addition to a symbolic label for each service on the network, the special symbol TRANS_UNUSED is included as a standard entry in the Transport layer definitions. This symbol refers to the collection of all ports on a system that are not currently being used by a listening service. Capturing this condition is needed to model a class of exploit called port forwarding, and results when connectivity-limiting devices permit connections to services that aren't actually running on the destination host. For example, Smithers can send packets to port 8080 on Marge even though a service isn't actually running on that port.

5. Modeling Application Layer Security

The first three layers of the TCP/IP reference model do not address all connectivity-related security issues. For example, an attacker might be able to connect with a Transport layer service, but might have to authenticate with a password to actually exploit it. TVA uses a separate Application layer to address these types of issues.
Some services such as telnet establish trust relationships based on password authentication. TVA represents this with an Application layer connection, e.g., APP_PW_AUTH. The sample telnet exploit shown in Figure 8 includes both the TRANS_TELNET and APP_PW_AUTH connections required to exploit it. Note that TVA includes the normal operation of services such as telnet in its exploit database, and applies them just as a real attacker would if he had successfully acquired a user's password.

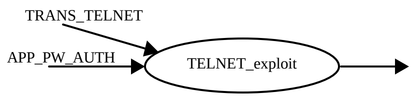


Figure 8 - Example Telnet Exploit

Some services such as the Berkeley `R' commands can be configured to trust hosts based on their IP addresses. Although an IP address is a Network layer field, the application itself applies it from its own configuration files, so such trust relationships are best represented at this layer. Figure 9 shows how TVA models the rcp command, where APP_RHOSTS represents the trust relationship configured in the system or user's rhosts file.

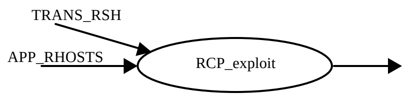


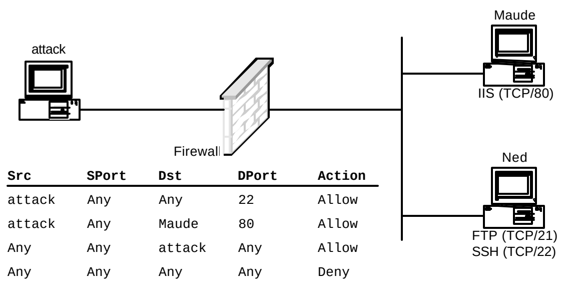
Figure 9 - Example Berkeley rcp Command Exploit

So far, we've presented several layers of connectivity, as they pertain to security, in the context of individual exploits. The real power of TVA, though, is its ability to chain exploits together into multi-step attack paths. The next section presents an example that highlights this capability.

6. An Example

To illustrate the ideas developed in this paper, we present a small network similar to the example in Section 4. Figure 10 is deceptively simple in terms of the number of hosts on the network, but offers an exploit path that might not be obvious to many administrators. The firewall policy is restrictive, allowing only Secure Shell (port 22) and IIS (port 80) traffic from the attack machine (i.e., inbound).

Secure Shell is allowed to both Maude and Ned, though you'll note it isn't actually running on Maude. Inbound IIS traffic is restricted to Maude, and inbound FTP traffic isn't permitted at all. Outbound traffic to the attack machine is unrestricted.



Src	SPort	Dst	DPort	Action
attack	Any	Any	22	Allow
attack	Any	Maude	80	Allow
Any	Any	attack	Any	Allow
Any	Any	Any	Any	Deny

Figure 10 - Final Example

A simplified TVA connectivity matrix for this configuration is shown in Table 1. The keyword 'ANY' indicates the source host can connect to any service on the destination, and is only used to make the matrix more readable. An actual TVA model would list all connections, but for simplification we only record those to and from machines Maude and Ned.

		Destination		
		attack	Maude	Ned
Source	attack	ANY	TRANS_IIS TRANS_UNUSED	TRANS_SSH
	Maude	ANY	ANY	TRANS_FTP TRANS_SSH
	Ned	ANY	TRANS_IIS TRANS_UNUSED	ANY

Table 1 - Example Connectivity Matrix

Now suppose TVA is configured with the exploits shown in Figure 11. The input arrows represent pre-conditions that must exist on the attacking machine to trigger the exploit, and the output arrows signify post-conditions that apply to the victim machine. For example, the FTP exploit shown in (b) requires Transport layer connectivity to an FTP service and the ability to execute programs on the attacking machine as pre-conditions, and yields the ability to execute programs on the victim machine as a post-condition. Besides the ACCESS_EXEC post-condition already mentioned, we included another condition that isn't related to connectivity in exploits (c) and (d). The PGM_PORT-FWD condition indicates that a port-forwarding program is installed on the machine. These conditions hint at other elements required in a full TVA model, and are included here to illustrate how exploit post-conditions can satisfy the pre-conditions of other exploits.

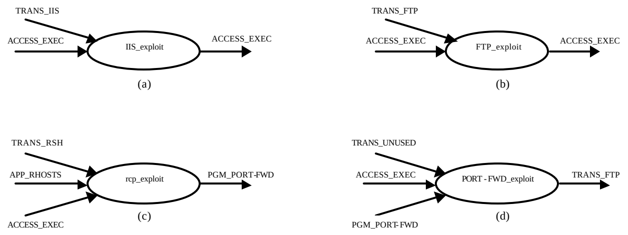


Figure 11 - Example Exploits

We set our attack goal as obtaining execute access on Ned. Assuming the attacker starts with the ability to execute programs on her own machine, the reader might compare the connectivity matrix and exploits to construct the exploit path shown in Figure 12, which realizes the goal even though the firewall blocks access from the attack machine to Ned. The attacker has TRANS_IIS connectivity to Maude, so she can execute the IIS exploit (a), which yields execute access on Maude. The new execute access enables her to copy a port forwarding program from the attack machine using RCP_exploit (c). Although we don't show it in the diagram, an rcp client is required to actually execute RCP_exploit. An rcp client is included in the default Windows NT installation, on which the IIS web server runs. Note that all the pre-conditions for (c) apply to Maude because the locus of attack has transferred there. Maude is exploiting its TRANS_RSH and APP_RHOSTS connectivity to hack the attack machine and download a program! Of course, it's the attacker that really drives the process, and she must set up her machine to allow the connections. The PGM_PORT-FWD download, combined with connectivity to an unused port, then triggers the port forward exploit (c), which in turn yields TRANS_FTP connectivity to Ned. Finally, the attacker takes advantage of the indirect access to Ned to execute the FTP exploit (b), which gives her the ability to execute programs on Ned. The goal has been realized.

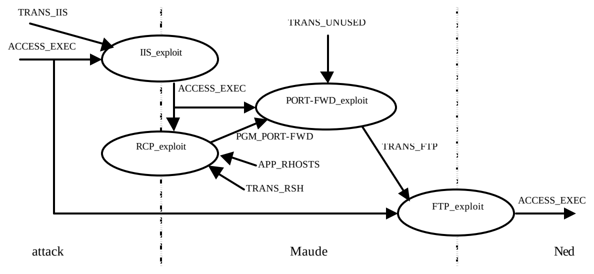


Figure 12 - Exploit Path

Manually constructing this exploit path is fairly trivial, but the complexity of a more realistic network with hundreds of machines and thousands of exploits is daunting. Fortunately, TVA automates the process so the problem becomes one of designing a network model with the flexibility to address all types of vulnerabilities and exploits.

7. Conclusion

The work presented in this paper substantially improves the ability for this analysis model to represent real-world networks. This is important because it allows the model to more closely represent the type of network connectivity issues that directly affect network security. These new enhancements were carefully designed to minimize their state-space requirements. In the small example networks that have been encoded so far, state-space issues are not much of a concern. However, when representing larger networks, the size of the state-space becomes a concern. By limiting the impact of the connectivity enhancements, larger networks are able to fit within the constraints of the current analysis tool.

Acknowledgements

The authors are grateful to several people for assistance with the creation of this paper. We would like to specifically thank Dr. Paul Ammann for his encouragement, Chuck Hutchinson for sharing his good ideas, and Dr. Sushil Jajodia and the GMU Center For Secure Information Systems for supporting the research.

REFERENCES

[1] Apache Web Server information and software on the web at www.apache.org.

[2] B. Beizer, "Software Testing Techniques, 2nd edition," Thomson Computer Press, 1990.

[3] J. Birch, E. Clark, K. McMillan, D. Dill, and L.J. Hwang, Symbolic Model Checking: 1020 States and Beyond, Proceedings of the ACM/SIGDA International Workshop in Formal Methods in VLSI Design, January, 1991.

[4] Coleson, Jay, An Elementary Introduction to Sendmail, The SANS Institute, 2000. http://www.sans.org/infosecFAQ/unix/intro_sendmail.htm

[5] Computer Oracle and Password System (COPS) information and software on the web at ftp.cert.org/pub/tools/cops.

[6] Internet Information Server information on the web at www.microsoft.com/iis.

[7] Internet Security Systems, System Scanner information on the web at www.iss.net.

[8] Ronald W Ritchey and Paul Ammann, Using Model Checking To Analyze Network Security, 2000 IEEE Symposium on Security and Privacy, May 2000.

[9] Oleg Sheyner, Somesh Jha, and Jeannette M. Wing, Automated Generation and Analysis of Attack Graphs, Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May 2002.