Chapter 18

ON MODELING COMPUTER NETWORKS FOR VULNERABILITY ANALYSIS

C. Campbell, J. Dawkins, B. Pollet, K. Fitch, J. Hale and M. Papa

Abstract Enterprise security must take into account a holistic view of the network. Capturing security and vulnerability attributes of network services and systems is a critical aspect of effective vulnerability analysis and remediation. Unfortunately, this is not always possible due to the overhead associated with tracking distributed resources. Conventional tools create topological maps of a network and extract a signature of the state of individual components. However, these tools require human interpretation to be useful for security. The goal of network modeling for vulnerability analysis is to glean and interpret data from a variety of resources in order to create an abstract model of the security of a network. A sound network model is essential to the analysis of potential threats to a network.

Keywords: Network modeling, vulnerability analysis, network mapping, host-based agents

1. Introduction

A principal shortcoming of current network security practices is their failure to consider the entire network. Comprehensive vulnerability analysis requires consideration of complex behavior in a composite network. Unfortunately, the overhead associated with exploring a universe of combinatorial network interactions makes this impossible with conventional tools. While network mapping techniques exist, most view the network from a management or performance perspective. Data produced by security tools require a significant amount of human interpretation to be useful for network security analysis.

Existing network modeling tools provide a shallow view of hosts and information services. However, Network scanners cannot provide a complete view of a host, and often resort to inference to construct a host profile. Key system attributes such as application patch level, hardware and platform configurations, and network service security settings are routinely ignored. A com-

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35697-6_26

E. Gudes et al. (eds.), *Research Directions in Data and Applications Security*

plementary approach harnessed within a modeling methodology is needed for comprehensive vulnerability analysis in large-scale information networks.

This paper presents a network modeling methodology as well as a host-based profiling technique, which captures critical security and vulnerability attributes in an enterprise network. The following section presents four phases of network modeling: (i) discovery, (ii) enumeration, (iii) classification and (iv) correlation. Section 3 defines discovery and enumeration, focusing first on existing network-based techniques for these tasks. Section 3 also explains the requirements and design of host-based extensions to these processes. Section 4 provides background on NetOBJ, a network object model, and its relevance to classification and synthesis. In Section 5, a network-centric vulnerability analysis system is described. The final sections present related work and conclusions.

2. Network Modeling

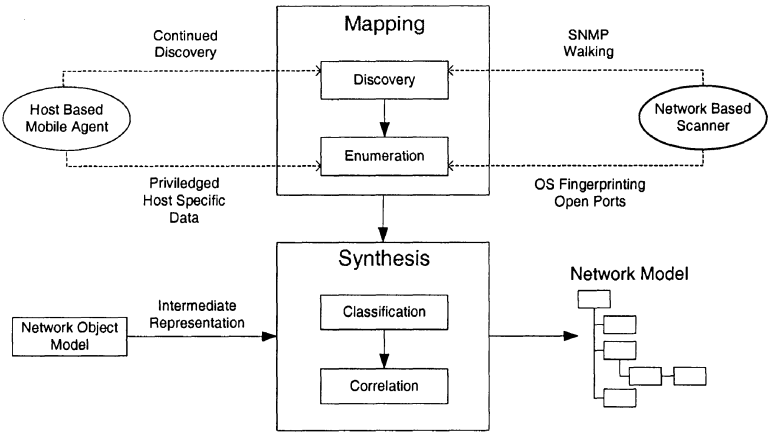

Figure 1. Network modeling.

Network modeling consists of four phases: (i) discovery, (ii) enumeration, (iii) classification and (iv) correlation (Figure 1). Discovery and enumeration are techniques for mapping – gathering raw information about networks. These techniques collect minimal data from network configurations, produce detailed information about network components and topology, and store results for subsequent analysis. Through continuous discovery, this process is repeated and incremental updates reflect the evolving network.

Network mapping encompasses traditional information gathering techniques but is only capable of obtaining limited information about system configurations. In contrast, host-based mapping techniques operate locally on each host, so they may have privileged access to system configuration details.

As seen in Figure 1, the output of the network mapping phase becomes input in the synthesis portion of the network modeling process. Classification, the first phase of synthesis, analyzes relevant information about network components detailed by the mapping process and outputs a set of entities as a network object model. The resulting model is passed to the correlation phase. This final phase utilizes network component and topology information to synthesize relationships between network objects.

3. Network Mapping

Network mapping is a two step process in which descriptive network information is gathered. In the discovery phase, individual network elements are identified but further information may not be gathered. The enumeration phase gleans network configuration data and forms a detailed signature of target elements. This section describes key concepts of discovery and enumeration, and identifies the limitations of each.

3.1 Discovery and Enumeration

The initial phase of the network mapping process, discovery, utilizes a variety of techniques to generically identify each component of a network. Identification is completed using techniques such as port scanning and SNMP walking [8]. Traceroute probing and DNS zone transfers are more sophisticated techniques that can be used to identify elements and services on a network [15].

Enumeration extends the discovery process by filling in informational gaps created as new network elements are identified. In the enumeration phase, the network modeler seeks to establish a signature of the system configuration for each network component. Techniques for the discovery and enumeration of networked systems often overlap. For example, NMap detects open ports and can predict operating systems running on scanned hosts. However, these tools lack the ability to capture many critical security and vulnerability attributes of networked systems. This failure is almost inevitable since the information gathered is based on statistical and predictive analysis. Thus, techniques allowing for precise and comprehensive analysis of host systems are needed.

3.2 Host-based Network Mapping

Host-based mapping reduces uncertainty in discovery and enumeration by analyzing systems from within. Exploiting host-based technology makes it possible to gather specific configuration information about host systems. While network-based scanners provide a statistical guess of services operating on a host, a host-based profiler may obtain precise information directly from systems, e.g., application identity, version and patch level. The following requirements support effective host-based network mapping:

1 A host-based solution must be dynamic enough to adapt to evolving network environments.

2 Platform independence is required to support heterogeneous systems in both traditional and converged networks.

3 Continuous discovery, the ability to detect changes in network environments, is vital due to fluctuating configurations in modern networks.

4 Host-based mapping tools must maintain confidentiality and integrity while gathering information.

5 Information gathered and synthesized must be stored in a standard format for analysis and reuse.

3.2.1 Design. The nature of host-based network mapping is that the functionality of a profiling system is distributed across the hosts of a network. A natural implementation strategy for such a system engages mobile code. Traditional cross-platform agent architectures meet the first two requirements above. Extensions to the agent paradigm offer a complete solution to host-based network mapping. Not only do agents move code and information between systems, they can accomplish this task securely. In addition, the dynamic nature of mobile code makes it adaptable to changing system requirements. Agents may be extended to support continuous discovery as they map network elements from varying perspectives in their propagation paths.

To accommodate these requirements in an agent-based architecture, the design of a host-based network mapping system must consider additional agent behavior, including propagation and information retrieval.

Propagation. The most basic decision in mobile code systems is the choice of agent destinations. A number of solutions to this problem exist. In the simplest case, a central server provides an agent with an itinerary of systems based on initial information gathered during network-based discovery [16]. An intelligent extension to this heuristic is to utilize the results of continuous discovery to determine which hosts have not been visited. This technique is limited only in that an agent service must be listening on each system.

Information Retrieval. The primary information retrieval technique used by mapping agents is host interrogation. Host interrogation makes use of host-resident libraries to gather detailed information from the hosting system; therefore, the appropriate libraries must be maintained on each system. Host interrogation uses operating system specific features to gather this information. For example, on Microsoft Windows machines, the system registry

is examined to determine the version of installed applications, patch level of the system and domain configuration options. Additionally, system level calls allow agents to correlate listening ports to running services without the uncertainty and incompleteness associated with network-based enumeration.

Continuous Discovery. Continuous discovery enables agents to gather information where host-based agent services are not, or cannot, be deployed. Continuous discovery merely extends traditional discovery and enumeration techniques by exploiting the ability of mobile code to "see" the network from all perspectives because of changing propagation paths.

The quality and quantity of information gathered via continuous discovery is a function of the complexity of agent code. A traditional port scanning approach is limited to identifying systems listening for connections and enumerating well-known services running on these systems. Superior techniques allow agents to make intelligent use of common network services. For example, DHCP servers and router tables contain valuable information about network topology that can be obtained by mobile code [15]. Microsoft Windows networks simplify this task by providing NetBIOS services. These services can be enumerated to discover shared drives, user logins, and a wealth of machine-specific information. Passive monitoring of Address Resolution Protocol (ARP) traffic exploits mobile code to identify active hosts, including those that are blocking incoming connections, from multiple perspectives.

4. Network Synthesis

In large networks, mapping generates a considerable amount of information. Network synthesis converts this information into representative network models. Synthesis inputs raw network mapping data and instantiates a network model against a given network object model. Network object models provide a standard means to identify and describe security and vulnerability attributes of a network. Therefore, a viable object model must meet specific criteria:

1 Network models must capture security and vulnerability attributes.

2 Systematic and efficient analysis require that physical properties of networks be abstracted.

3 Network models should allow for the extension and reuse of network components to accommodate future technologies and vulnerabilities.

4 Network models should be extensible to other domains, including converged environments, e.g. telecommunications and wireless networks.

4.1 NetOBJ: A Network Object Model

NetOBJ is a relational object model designed to capture critical network attributes in object hierarchies. As a high-level model, NetOBJ is distinguished from other low-level network modeling tools such as OPNET [5, 8, 12]. The building blocks of a NetOBJ model include three core network objects, *Component, Host* and *Trust*, explained below. This section also introduces the Network Modeling Language (NetML), a language designed to construct NetOBJ objects.

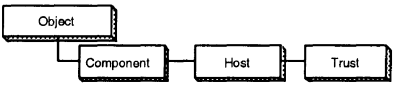

Figure 2. Basic NetOBJ objects.

4.1.1 Defining Components.

Components represent the aggregate pieces of network entities. While components do not exist independently on networks, they represent exploitable elements critical to the operation of systems. Such elements include operating systems, services, applications, and protocols, represented by the *OperatingSystem, Service, Application*, and *Protocol* NetOBJ types as seen in Figure 4.

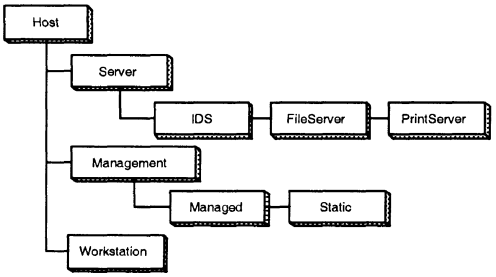

Figure 3. Host NetOBJ object.

4.1.2 Defining Host Objects.

Using individual *Component* objects, it is possible to describe complex host systems. A *Host* is defined as an entity capable of communication on the network. This definition is quite generic, and thus it is necessary to extend the host type based on the functionality of the target systems. The basic NetOBJ hierarchy includes three such subtypes, *Server, Management* and *Workstation.*

A *Server* is a host that provides a service, such as HTTP, FTP or POP, to other network elements. Similarly, *Management* objects control the actual operation of a network. For example, the *Router* subtype abstracts the class of

network components that regulate the flow of packets from source to destination within the network. Management objects are further classified as *Managed*, such as an intelligent switch, or *Static*, as in the case of a simple unmanaged hub. Finally, the *Workstation* subtype is a representation of a typical networked computer.

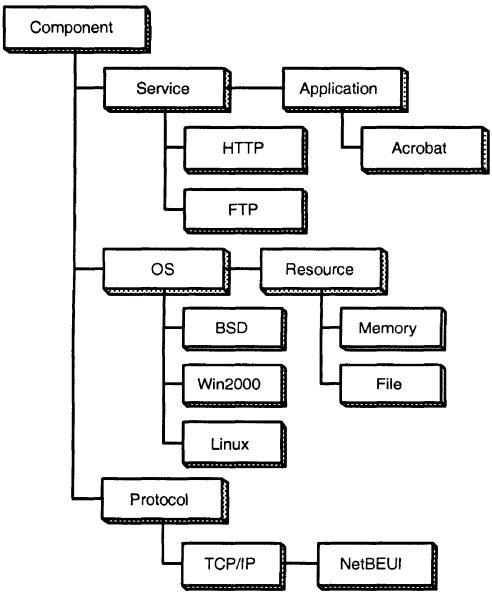

Figure 4. Component NetOBJ object.

4.1.3 Defining Trust Relationships. A network is essentially a set of related systems; however, *Component* and *Host* objects are limited to expressing independent entities within a network. The *Trust* type is one solution to this shortcoming. Trust relationships represent the intentions, integrity and nature of interactions between *Host* objects. Trust relationships offer substantial expressive power to the NetOBJ model since they can represent any number of relationships between systems, including a simple physical connection on a network, sharing of cryptographic keys or access privileges between clients and servers.

4.1.4 NetML. Just as components, hosts, and trust relationships are the building blocks of network object models, NetML is a language tool for constructing NetOBJ objects, based on traditional object-oriented languages. NetML expressions are parsed into representative NetOBJ object models stored in a relational database for further analysis.

Types and Elements. NetML programs consist of *types* and *elements*. Types are the equivalent of classes in modern object-oriented languages, while elements are instantiations of these types (Example 1). NetML types are composed of attributes cast as other NetML types or as primitive data types: string, boolean, integer, version or date. In addition, sets can be used to define multiplicity constraints, such as patch level or running services.

```
type Apache extends Service {
  ApacheMod set mod;
}

element ApacheHTTP_1.3 as ApacheHTTP {
  id = 1234;                          \\ inherited from Object
  name = "Apache HTTP Server";        \\ inherited from Component
  currentVersion = v1.3.11;           \\ inherited from Service
  patches = {ApacheSec12_14_01,       \\ inherited from Service
             ApacheSec3_18_02};
  port = {80, 443};                   \\ inherited from Service
  mod = {mod_ssl, mod_php4, mod_ldap}
}
```

Example 1: Type definition and element.

4.2 Classification

Using the NetOBJ object model, synthesis creates abstract models of target networks. Classification is the first step of this process. Classification uses simple algorithms to map network elements to corresponding NetOBJ elements via an intermediate representation language (NetML).

4.2.1 Correlation. Correlation combines the output of classification with network topology data, to create a complete network model abstracting relationships between systems. The correlation process must be limited since large networks contain indefinite numbers of relationships. The key to the process is to identify and express trust relationships critical to vulnerability analysis within the network.

The correlation engine is functionally similar to its classification counterpart. NetOBJ *Trust* objects are built from independent *Host* objects based on cues extracted from XML tags. As in classification, correlation associates *Trust* objects to the original XML data structure by including a key attribute.

Trust relationships make it possible to represent most kinds of network associations. For example, a group of systems that provide unauthenticated access through rlogin are grouped into a *RHostTrust* based on information obtained from host-based network mapping. In addition, a vulnerability analysis system may determine the members of a network subnet by examining a *SubnetTrust* object. Trust can be extended to represent nearly any communicative relationship, no matter how primitive or advanced. Moreover, the ability to model such relationships is critical to vulnerability analysis systems.

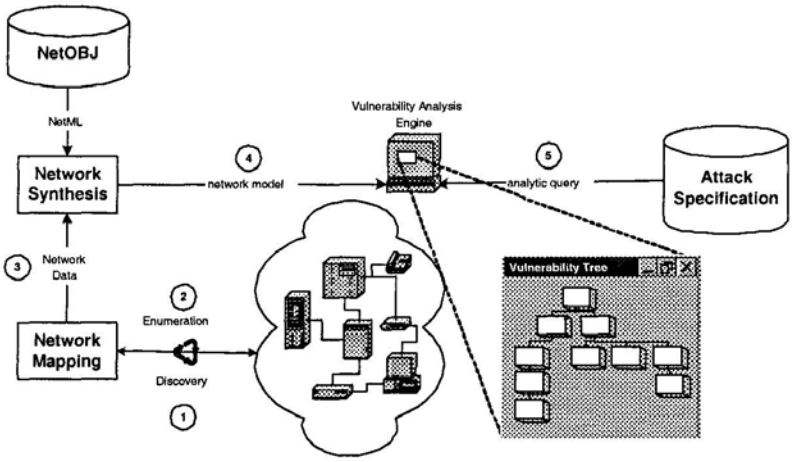

Figure 5. Vulnerability analysis architecture.

5. Vulnerability Analysis

With more recent advances in the dissemination of security related information, potential vulnerabilities and threats are more easily obtained by security personnel. The ability to identify relevant information given the onslaught of new vulnerabilities is critical for effective security administration. In addition, tracking the evolution of network topologies and system configurations is extremely important for vulnerability analysis, particularly in converged environments where heterogeneity, and therefore complexity, are intrinsic features.

Network modeling provides the foundation for a vulnerability analysis system capable of identifying and predicting attack scenarios based on a holistic view of the network. The system in this section uses attack models in conjunction with a network object model to identify and analyze potential vulnerabilities within a network [3]. Attack models are represented as attack trees, which express goal-oriented attack plans as hierarchical data structures [14]. The result is a set of scenarios useful for identifying threats to a given network.

A network model is used to enhance attack trees to provide a context sensitive attack model. Network types are used to identify potential vulnerabilities and their specific attributes. Network elements are used to identify a system specific vulnerability. Because system vulnerabilities exploit specific versions or builds of software and hardware, a vulnerability analysis system must have access to these properties. By incorporating such properties in a network model and making them accessible through attack models, false positive rates can be significantly reduced. A simple dataflow diagram illustrating vulnerability analysis in the distributed architecture is shown in Figure 5.

Network mapping (step 3) is obtained by data collected over enterprise networks through discovery and enumeration (steps 1 and 2). As seen here, these

initial steps are repeated indefinitely through the process of continuous discovery. The network data collected is Synthesized against known NetOBJ objects to create a specific network model (step 4). While the generated expressions may contain partial information, humans can augment incomplete specifications with salient details and attributes. Additionally, individual network elements are grouped into pertinent trust relationships through the process of correlation. Finally, the resulting object model is used by a vulnerability analysis engine in conjunction with a set of attack models to yield comprehensive vulnerability analysis.

By overlaying network and attack models users can issue queries such as "Find all vulnerabilities associated with host x" (step 5). By issuing such a query, the Vulnerability Analysis Engine binds network element arguments to parameters within matching attack templates. Recursive queries produce vulnerability trees identifying potential multi-stage attacks directed at system targets.

6. Related Work

Most modern techniques for network discovery and enumeration concentrate on topology visualization to provide users with a graphical representation of network operations [5, 6, 8]. Products such as Microsoft Visio's Network AutoDiscovery, use the Simple Network Management Protocol (SNMP) to inventory routers, switches, hubs, and workstations [8]. AutoDiscovery reports this information back to a server, which generates a visual representation of the network. Other tools, such as the CAIDA suite were developed to handle visualization for a wide variety properties of the Internet backbone, such as topology, workload, performance, and routing [5, 6]. Though they provide useful management and visualization tools, these applications lack the necessary information to discern vulnerabilities within networks.

Recent advances in security have inspired more sophisticated network enumeration tools. Retina is designed to scan machines of a network to identify existing vulnerabilities, and check adherence to established security policies [13]. This is accomplished by an "artificial intelligence" engine that simulates the thought process of a hacker. While Retina is a valuable tool for identifying vulnerabilities, our model provides a network-centric framework for developing more powerful vulnerability systems.

Mobile management applications provide an "intelligent network model" utilizing mobile Java agents [16]. Such systems are designed for use as a centralized management and visualization system. Furthermore, they typically rely on vendor-provided "properties sites," in which network element properties can be retrieved to build a network model. In contrast, our system focuses on security relevant network information.

7. Conclusions

The ability to precisely identify security related attributes of network resources is crucial to their protection. Network modeling systems must be capable of expressing security relevant information in a comprehensive network object model. To this end, we describe a modeling framework that adopts mapping and synthesis as core processes. Host-based network mapping moves beyond traditional network scanning by providing a comprehensive profile of networked systems from within. An agent-based prototype under development in Java targets system-wide host profiling. Synthesis encompasses network element classification and correlation. NetOBJ and NetML provide an object-oriented foundation for modeling network components, hosts, and trust relationships. These tools are designed to support comprehensive network vulnerability analysis in converged and evolving environments. Only by engaging more detailed network representations in analytical tools can system administrators defend their information enterprises from cyber attack.

Acknowledgments

This research was supported by the Department of Justice through the Institute for Security Technology Studies at Dartmouth College and by National Science Foundation Cooperative Agreement No. HDR-945-0355.

References

[1] A. Bieszczad, T. White, and B. Pagurek, Mobile agents for network management, *IEEE Communications Surveys*, Vol. 1(1), 4^{th} Quarter, 1999.

[2] C. Bryce and J. Vitek, The JavaSeal mobile agent kernel, *Proceedings of the First International Symposium on Agent Systems and Applications/Third International Symposium on Mobile Agents*, Palm Springs, California, 1999.

[3] J. Dawkins, C. Campbell, R. Larson, K. Fitch, T. Tidwell and J. Hale, Modeling network attacks: Extending the attack tree paradigm, *Proceedings of Third Annual International Systems Security Engineering Association Conference*, Orlando, Florida, 2002.

[4] A. Hayzelden and J. Bigham, Software agents in communications network management: An overview, Intelligent Systems Application Group, Technical Report, University of London, London, U.K., 1998.

[5] B. Huffaker, E. Nemeth and K. Claffy, Otter: A general-purpose network visualization tool, *Proceedings of the Internet Global Summit*, San Jose, California, 1999.

[6] B. Huffaker, E. Nemeth, D. Moore and K. Claffy, Topology discovery by active probing, *Symposium on Applications and the Internet*, Nara, Japan, 2002.

[7] J. Humphries, C. Carver and U. Pooch, Secure mobile agents for network vulnerability scanning, *Proceedings of the 2000 IEEE SMC Workshop on Information Assurance and Security*, United States Military Academy, West Point, New York, 2000.

[8] J. Lemke, Discover, diagram and report on your network, Technical Report, Microsoft Corporation, Redmond, Washington, 2001.

[9] S. Liang, *The Java Native Interface*, Addison-Wesley, Reading, Massachusetts, 1999.

[10] N. Minar, K. Kramer and P. Maes, Cooperating mobile agents for mapping networks, *Proceedings of the First Hungarian National Conference on Agent Based Computation*, 1999.

[11] N. Minar, K. Kramer and P. Maes, *Software Agents for Future Communications Systems*, Springer-Verlag, Heidelberg, Germany, 1999.

[12] Optimum Network Performance, *http://www.opnet.com*, Bethesda, Maryland, 2001.

[13] Retina, Eeye Digital Security, *http://www.eeye.com/*, Aliso Viejo, California, 2002.

[14] B. Schneier, *Secrets and Lies*, John Wiley, New York, 2000.

[15] R. Siamwalla, R. Sharma and S. Keshav, Discovering Internet topology, *Proceedings of the IEEE INFOCOM Conference*, New York, 1999.

[16] T. White, B. Pagurek and A. Bieszczad, Network modeling For management applications using intelligent mobile agents, *Journal of Network and Systems Management*, Vol. 7(3), 1999.