In Poc of IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation, Tuebingen, Germany, September 2006.

Automated Protection of End-Systems Against Known Attacks

Andreas Hess and Niels Karowski
Telecommunication Networks Group
Technical University Berlin
Address: Sekr. FT5-2, Einsteinufer 25, D-10587 Berlin
Phone: +49 (30) 314-23819, Fax: +49 (30) 314-23818
Email: [hess,karowski]@tkn.tu-berlin.de

***Abstract*— Many users do not maintain their systems and hence these systems remain vulnerable. An evidence for this is given by the German Honeypot Project that identified from November 2004 to January 2005 over 100 active botnets, some with up to 50 000 compromised hosts. To efficiently protect users' end-systems we propose the operation of a network-based intrusion prevention overlay network that analyzes network flows only for relevant attacks. This paper presents an approach to gather network-based vulnerability information necessary to appropriately configure the intrusion prevention overlay network. The approach proposed reduces the false-positive rate by following the principle of demand-driven intrusion prevention. We demonstrate the feasibility of the approach by conducting a proof-of-concept experiment.**

I. Introduction

Users and administrators are very slow in applying fixes to vulnerable systems [24]. Many private and professional users are not sensitive to security vulnerabilities affecting their own machines or they are just overstrained patching these. Furthermore, many users believe that they will never become the target of an attack, due to irregular on-line times, changing IP-addresses or having the perception that their system or data, respectively, is not of value for potential hackers. Unfortunately, this is not true: the W32/Blaster worm [5] which started on August 11th 2003 exploited a vulnerability that had been known for four weeks. Actually, since July 16th 2003 Microsoft had provided a patch in order to fix the flaw. But still, the worm could diffuse itself in a manner such that Symantec classed it as category 4 (severe threat, global distribution). The Internet worm Netsky-P was responsible for 17.5 % of all virus incidents that were registered in the first six months of 2005 although it was spotted the first time in March 2004 and an effective protection against it has been made available at the end of that month [27]. Another proof is given by the German Honeypot Project [23] which studied and observed *botnets*—networks consisting of compromised hosts—from November 2004 to January 2005. Botnets were used to launch 226 distributed denial of service attacks targeted at 99 different destinations throughout this period. The project identified over 100 active botnets, some with up to 50 000 compromised machines.

An *Intrusion Prevention System* (IPS) has the ability to detect attacks and prevent them from being successful. It combines the blocking capabilities of a firewall with the more thorough packet inspection of an *intrusion detection system* (IDS). An important metric to assess an IPS is accuracy [7], [18], [22]: the capability of an IPS to differentiate between legitimate and illegitimate actions (aka *false positive* rate). A false positive occurs when:

- an IPS thinks that the inspected network traffic belongs to an ongoing attack although current activity is benign or
- an IPS correctly identifies an attack but the attack is non-threatening or not applicable to the site.

Axelson identified the base-rate fallacy [2], saying that the effectiveness of an IDS is determined by the ability of the system to suppress false alarms. With respect to IPSs the false positive rate is critical: as harmless packets that trigger a false alarm are filtered and hence, valid communications are interrupted or even stopped. Moreover, IDSs are known to produce large amount of alarms per day, most of which are irrelevant to the overall security of a network [16], [26]; this is also valid for IPSs as they use the same set of attack detection techniques.

In this paper we extend our previous work on FIDRAN which is a network-based intrusion prevention overlay network on top of programmable routers [13]. With this architecture, intrusion prevention services can be dynamically distributed in the network such that the impact of doing intrusion prevention on the network performance is minimized [12]. This paper describes how to identify the required intrusion prevention services and how this information is used to limit the security checks. The approach proposed reduces the false-positive rate and accelerates the decision process of whether an attack is currently taking place.

II. Related Work

The traditional approach to reduce the false positive rate of an IDS is to manually update and configure it. For example, to appropriately operate the open-source IDS Snort, a user must configure environment variables (IP addresses, TTL values, etc.), preprocessor behavior (e.g. how to decode URLs), output module and the set of attack rules that the packets are checked for—a task too difficult for most users.

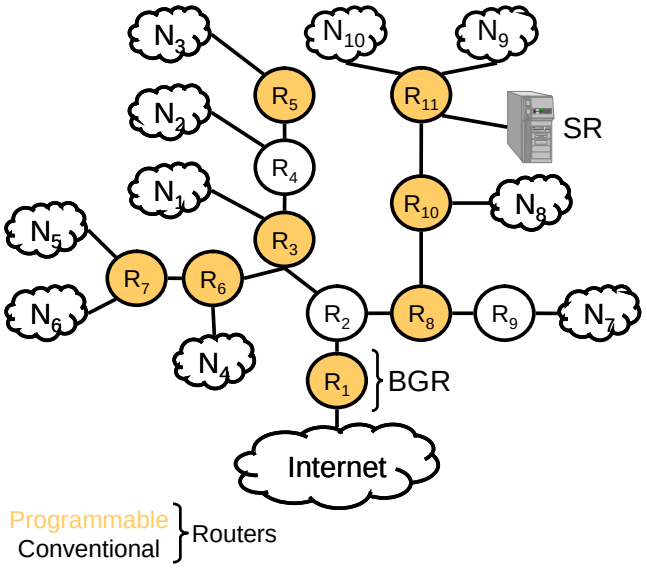

Fig. 1. An example Autonomous System (AS)

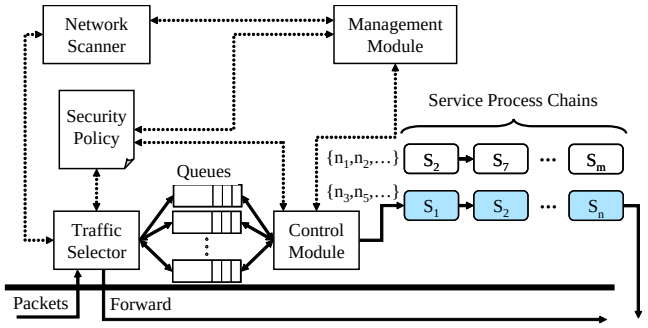

Fig. 2. The FIDRAN-architecture

Another set of approaches uses the *alarm-/alert-correlation* technique which combines alerts that are created by multiple IDSs [6], [14], [19]. The assumption is that a real attack would be identified by more than one IDS. Accordingly, alerts can be collected, pre-processed (e.g. transformation into a readable format as the alerts stem from varying IDSs) and aggregated into significant high-level alarms.

Julisch argues that each alarm that is triggered by an IDS can be backtracked to a limited set of *root-causes* [15]. He proposes an offline alarm clustering and data mining process which allows to identify the root causes. Both mentioned concepts of alert correlation are not applicable to IPS as they contradict the requirement for performance.

A promising approach to reduce the false positive rate is to incorporate network knowledge into the decision process of whether an attack is taking place, as proposed in [17], [25], [26]. For example, an ongoing *IIS*-specific attack targeted against an *Apache* web server is not relevant, since the targeted end-system is not vulnerable to this exploit. The proposed approaches have in common that they do not use the gathered network knowledge to configure the systems. Instead, the knowledge is incorporated at the output of the intrusion detection/prevention system, after the generation of an alarm. As a consequence, each intrusion detection/prevention system performs all possible security checks, many of them unnecessary. Vulnerability information could be used to reduce the amount of security checks.

III. FIDRAN AND DEMAND-DRIVEN INTRUSION PREVENTION

FIDRAN is designed to be operated in an *Autonomous System* (AS) which is a part of the Internet that is under the control of a single administrative entity—for example a university or an enterprise. An exemplary AS is depicted in Figure 1, it consists of a number of routers R_i that connect several subnetworks N_j to each other and to the Internet through the *Boarder Gateway Router* (BGR). A subnetwork consists of a varying amount of different end-systems. Each router in the AS can either be of passive or programmable nature which are part of the intrusion prevention overlay network. For protecting the end systems, there has to be at least one programmable router on the path between potential attackers (Internet and all subnets) and each subnet.

Demand-driven intrusion prevention makes use of the fact that most attacks require the existence of one or multiple concrete vulnerabilities to succeed. For example, the *Code Red* attack exploits a buffer overflow that exists in certain versions of Microsoft's *Internet Information Server* (IIS). In our approach, an intrusion prevention service provides protection against attacks exploiting a concrete vulnerability. Flows to an end-system are only analyzed by intrusion prevention services that protect against attacks that could actually harm it. In this way, our approach reduces the amount of signatures for which each flow is checked. This reduces the rate of false positives, as we explain next. Let p_i be the probability of falsely classifying a packet as malicious for signature i (false positive rate of signature i). $(1 - p_i)$ is then the probability of correctly classifying the packet, and $(1 - p_1) \cdot (1 - p_2) \cdot \cdots \cdot (1 - p_N)$ the probability that a benign packet is correctly classified by N signatures. The false positive rate for N signatures is then

$$1 - \prod_{i=1}^{N}(1 - p_i),\ 0 \leq p_i \leq 1. \quad (1)$$

Independently of the values of p_i, the product decreases for increasing N and the false positive rate increases accordingly.

For this, knowledge of the relevant end-systems and their vulnerabilities is a mandatory requirement and, since networks vary over time, that knowledge must be continuously gathered automatically (Section III-B).

A. The FIDRAN Architecture

This section briefly describes the FIDRAN architecture (cf. Fig. 2); for a detailed discussion we refer to [13]. The framework consists of core components which run permanently and of add-on components, the intrusion prevention services. The system is designed such that a dynamic reconfiguration

at runtime (insertion and deletion of intrusion prevention services) is possible. The core functionality comprises traffic selector, security policy, control/management module, network scanner and a varying amount of waiting queues. Intrusion prevention services are stored on a central server, the *service repository* (SR). In addition, the programmable routers provide the capability to securely communicate with each other.

All network traffic is redirected to the traffic selector, which—according to the rules specified in the security policy—assigns the traffic to one of the categories: *forward*, *process* or *drop*. Traffic that is assigned to the category *forward* is directly forwarded and not analyzed by any installed intrusion prevention service. Traffic selectors running on last programmable hops (e.g. router R_7 in Figure 1 is the last programmable hop for subnetworks N_5 and N_6) are assigned to do egress filtering: network packets originating from a subnetwork of the AS must have a corresponding IP source address otherwise they are dropped. Analogously, the traffic selector running on the BGR is doing ingress filtering.

The security policy instructs the traffic selector how to queue which network traffic of category *process* and it specifies which traffic must be analyzed by which security services. In addition, the security policy contains a list of valid destination addresses (see Section III-D). The management and the control modules are responsible for the configuration of the FIDRAN system. The management module is the interface between the overlay network of programmable routers and the FIDRAN system, and it is able to trigger the download of a security service from the SR. The control module coordinates the downloaded security services. In order to realize a demand-driven intrusion prevention system, the required security services for a specific set of destinations are concatenated and stored in a *service process chain*. A process chain is a linked list of security services with an unique identifier that is composed of IP-address(es), protocol and port(s). A security service can be part of one or more process chains. Once the chains are set up, the control module takes a packet from one of the waiting queues, inspects it, and forwards it to the appropriate process chain.

B. Gathering Network Knowledge

Since most attacks exploit one or multiple concrete vulnerabilities that are specific to an OS or application, it is essential to identify:

- the AS-topology,
- the reachable end-systems:
  - distance between Internet and end-system,
  - running OS and applications,
  - amount of traffic that is destined to an end-system

This information enables to specify both the intrusion prevention services requested by each end-system, and all placement possibilities for each service. The distance between Internet and an end-system is used to filter packets that would not reach its destination (based on the TTL). Further, the amount of traffic that is destined to an end-system is required to optimally deploy the intrusion prevention services [12].

Generally, three techniques exist to gather vulnerability information: *active scanning*, *passive fingerprinting* and *co-operation*. An active vulnerability scanner sends specifically crafted packets to well defined addresses and evaluates the replies. The scanner either operates in an *intrusive* or *non-intrusive* manner. The former technique identifies the vulnerabilities of an end-system by actually exploiting them. On the one hand the results gained are accurate but on the other hand this might result in an application-/system-crash. In contrast, a non-intrusive network scanner identifies a vulnerability on the basis of the type and version of a running application. Consequently, the results produced by a non-intrusive network scanner are not as accurate as the results of an intrusive one, but normal network operations are not affected by it. Moreover, the gathered vulnerability information is only as current as the latest scan and accordingly, the questions arise when and how often to scan? In addition, an active scanner creates traffic for the purpose of identifying end-systems. Considering, large networks this might result in a huge amount of scanning tasks.

The passive fingerprinting technique uses a network interface card (NIC) in promiscuous mode to sniff the network. It does not collide with normal network operations but it is impossible to predict the point of time when all end-systems of a network will be identified. End-systems that do not communicate cannot be detected by the passive fingerprinting technique even though those systems are potential victims.

The cooperation technique requires a consent between end-systems and network analysis tool. One approach is that each end-system runs a small application that registers at the analysis tool and subsequently, it provides the analysis tool with the required knowledge in terms of operating system and running applications. In a second approach, the network analysis tool logs into the end-systems and gathers autonomously the relevant data. Therefore, the end-systems provide a login to the network analysis tool. The advantage of the cooperation technique is accuracy. Operating system and running applications—including version number and patch level—are precisely identified. But not all users feel comfortable about either running a small network analysis client application on their machines or providing a login to the network analysis tool.

In our context, the gathering of vulnerability information must occur in an accurate and non-intrusive manner. An accurate identification of the end-systems' vulnerabilities allows for an adequate protection and further, normal network operations must not be disturbed. Finally, the knowledge gathering process must be able to consider networks dynamics like:

- Varying amount of reachable end-systems due to variable online times; for example a special purpose server is expected to be permanently reachable whereas a private end-system might only be connected to the Internet for a short period of time (changing IP address),
- Varying end-system configuration: an end-system might become vulnerable due to the start of a new application for which an exploit exists or a patch is applied to an

end-system eliminating a set of security holes.

The issue of *Network Address Translation* (NAT) is currently not addressed. But in some cases—depending on the operating systems of the hosts—it is possible to identify the number of systems that are located behind a NAT box [3].

C. Gathering Vulnerability Information

Our network analysis includes three functional parts. First, the BGR calculates the AS-backbone using the routing tables which can be polled via SNMP of all the routers of the AS. Second, an active scanning component is regularly started on all programmable routers to discover and analyze the end-systems of the AS. Finally, to catch changes, last hop traffic selectors scan the network traffic for new destination addresses (IP-address, destination port) that lie within the IP-address range of the AS.

Initially, the BGR analyzes how the routers of the AS are connected with each other. At system startup it requests the routing table of all routers of the AS. On the basis of these routing tables and the assumption of symmetric routes, it calculates the AS-backbone using Algorithm 1. The algorithm starts by identifying the children of the BGR which are the next-hop routers towards the AS's end-systems. For this purpose, the routing tables of all routers of the AS are searched for *default* routing entries (aka default routes). Normally, the default route points towards the router that has a connection to the Internet. Take the following entry as an example:

```
Destination     Gateway       Genmask  ...  Interface
0.0.0.0         192.168.50.1  0.0.0.0  ...  eth1
```

The default route tells the routing daemon to send all IP-packets that have not yet been routed to the next hop via interface *eth1*, which is specified via the gateway entry (192.168.50.1). In case that the default route lists the BGR as gateway, the router is directly connected to the BGR and is its child. This children identification process is successively repeated for all routers in the AS. To identify more generic AS topologies, topologies that cannot be modeled by a tree, a slightly modified version of Algorithm 1 is applied to analyze how the subnetworks are connected with each other. Finally, the BGR triggers the programmable routers next to subnetworks to scan the corresponding IP-address range.

D. Host Profiling with Respect to Network Dynamics

Each programmable router is supplied with an network scanning component which is used to discover and analyze end-systems. To do so, the scanner sends specifically crafted packets to the addresses specified by the BGR and evaluates the replies (see [10], [11]). The local security policy stores all reachable destination addresses—tuple IP-address, port number—of the corresponding subnetwork in a hash-table. Afterwards, each programmable router sends its results to the BGR which combines them to an overall map of the AS, showing running end-systems and their configurations. The AS-map is used by the BGR to calculate an optimal distribution of the intrusion prevention services, as described in [12]. Then, the BGR triggers the programmable routers to install the required intrusion prevention services and to remove those that are no longer requested according to the newly calculated optimal distribution.

```
program buildnetwork

Parameters:
  routerlist;  // Routers of the AS - first entry is BGR
1.) Init
    router r = routerlist.first;
2.) Identify the children of r
    for(allRouters of routerlist)
     for(allInterfaces of router)
      for(allRoutingEntries of interface)
       if((routingentry == defaultroute) && (gateway == r))
        r->addchild(router)
3.) Goto next router in routerlist
    r = r->next
4.) Identify the children of router r
    Goto step 2

end buildnetwork
```

Algorithm 1: Specifying the AS-Backbone

To address network dynamics—the vulnerability information represents the view of the last scan—the network analysis process is bipartite. On the one hand the active scanning process is repeated in regular intervals to update the AS map and security policies. On the other hand, to immediately catch changes, the last hop traffic selectors scan the downlink network traffic for new destination addresses. Whenever one is detected, the traffic selector triggers the BGR to actively scan it. Afterwards, the BGR updates the security policies.

IV. A Case Study

To assess the feasibility of our approach we implemented a Linux-based FIDRAN prototype including a collection of pattern-searching intrusion prevention services—using algorithms Aho-Corasick and Boyer-Moore-Horspool (a modified version of the original Boyer-Moore algorithm) in combination with the attack rule database of the current Snort release (containing over 6,000 attack signatures). The intrusion prevention services, the control module, the traffic selector and the queuing disciplines were implemented as loadable kernel modules, whereas management module and network scanner run in user-space. For the latter we used the network scanner *Nessus-3.0.2* [8] as it performed well in a comparison of existing vulnerability scanners [9], [21]. Moreover, Nessus provides an overall amount of $10,000$ security checks which are updated each day.

To specify the amount of intrusion prevention services that are required to protect a typical user's end-system, a host running Microsoft's Internet Information Server on a Windows XP Professional operating system—patched with Service Pack 1—was scanned. Nessus provides several output formats, but all of them are inappropriate for an automated evaluation. The output for two exemplary security holes is shown in Vulnerability 1 and 2.

Several approaches to name vulnerabilities exist and most of them are supported by Nessus. The *Common Vulnerabilities and Exposures* list [1] provides the most comprehensive index of standardized names for vulnerabilities. Microsoft

```
results|192.168.100|192.168.100.68|epmap (135/udp)|11890|
Security Hole|
A security vulnerability exists in the Messenger Service
that could allow arbitrary code execution on an affected
system. An attacker who successfully exploited this
vulnerability could be able to run code with Local System
privileges on an affected system, or could cause the
Messenger Service to fail. Disabling the Messenger Service
will prevent the possibility of attack.
This plugin actually checked for the presence of this flaw.
Solution : see
http://www.microsoft.com/technet/.../ms03-043.mspx
Risk factor : High
CVE : CVE-2003-0717
BID : 8826
Other references : IAVA:2003-A-0028, IAVA:2003-a-0017,
                   IAVA:2003-b-0007
```

Vulnerability 1: CVE-2003-0717

uses its *Security Bulletin ID* to name Windows-specific vulnerabilities. Further examples, are Bugtraq- and IAVA-ID. But unfortunately, none of them is exhaustive: For example, Vulnerability 1 is stored in the CVE list (*CVE-2003-0717*) but Vulnerability 2 is not. Hence, to automatically configure the FIDRAN overlay network a script parses and extracts the Nessus report for security holes.

To follow the principle of demand-driven intrusion prevention two types of intrusion prevention services were implemented:

- intrusion prevention services containing all Snort attack signatures that compromise a concrete vulnerability—the vulnerability is clearly identified via the *Common Vulnerabilities and Exposures* (CVE) number, and
- intrusion prevention services containing all Snort attack signatures to protect a concrete application—the application is identified via its name.

A Nessus plugin checks for the presence of one or multiple concrete vulnerabilities and in most cases they are reported via the corresponding CVE numbers. Accordingly, if *all known* vulnerabilities of an application have a CVE-number assigned then the vulnerable application can be protected by the intrusion prevention service that contains all CVE-specific attack signatures. Otherwise, if Nessus reports *at least one* vulnerability that does not possess a CVE number then the vulnerable application is protected by the intrusion prevention service containing all application-specific attack patterns, even if that application also has CVE-named vulnerabilities. In this way, at least for some applications, the number of attack signatures can be significantly reduced and consequently the false positive rate (see Equation (1)).

```
results|192.168.100|192.168.100.68|microsoft-ds (445/tcp)
|12209|
Security Hole|
Synopsis :
Arbitrary code can be executed on the remote host due
to a flaw in the LSASS service.
Description :
The remote version of Windows contains a flaw in the
function DsRolerUpgradeDownlevelServer of the Local
Security Authority Server Service (LSASS) which may allow
an attacker to execute arbitrary code on the remote host
with the SYSTEM privileges.
A series of worms (Sasser) are known to exploit this
vulnerability in the wild.
Solution :
Microsoft has released a set of patches for Windows NT,
2000, XP and 2003 :
http://www.microsoft.com/technet/.../ms04-011.mspx
Risk factor : Critical
/ CVSS Base Score : 10
(AV:R/AC:L/Au:NR/C:C/A:C/I:C/B:N)
Other references : IAVA:2004-A-0006
```

Vulnerability 2: LSASS-specific

To verify the principle of demand-driven intrusion prevention we conducted a proof-of-concept test. We setup a small testbed consisting of three Pentium III 800 machines:

1) host-1: the victim, running Windows XP, SP1, IIS
2) host-2: the programmable router
3) host-3: the attacker

All packets originating from the attacker and destined to the victim are routed via the programmable router and, initially, no FIDRAN system was running on the programmable router. To attack the victim we used the *Metasploit* framework [20] which is an open-source platform for developing, testing, and using exploit code. With Metasploit it can be demonstrated that an attacker is able to execute arbitrary code on the victim. The successful attack exploits the vulnerability described in Microsoft's security bulletin *ms03-026* (superseded by *ms03-039*). In detail, the victim is running a vulnerable implementation of the RPC interface which can be compromised by an attacker to execute arbitrary code. Worms like *Blaster*, *Nachia* or *Welchia* used, among others, that security hole to gain access to end-systems.

Next, FIDRAN was started on the programmable router and Nessus identified the following vulnerabilities:

1) **CVE-2003-0717**: A security vulnerability exists in the Messenger Service that could allow arbitrary code execution on an affected system
2) **CVE-2003-0818**: ASN.1 parsing vulnerabilities, arbitrary code can be executed on the remote host
3) **CVE-2003-0715, CVE-2003-0528, CVE-2003-0605**: RPC interface buffer overrun, arbitrary code can be executed
4) **CVE-2005-0048, CVE-2004-0790, CVE-2004-1060, CVE-2004-0230, CVE-2005-0688**: Arbitrary code can be executed on the remote host due to a flaw in the TCP/IP stack
5) **CVE-2005-1984**: Arbitrary code can be executed on the remote host due to a flaw in the Spooler service
6) **CVE-2005-1206**: Arbitrary code can be executed on the remote host due to a flaw in the SMB implementation
7) **CVE-2004-0212**: A remote code execution vulnerability exists in the Task Scheduler, arbitrary code can be executed on the remote host (task scheduler).
8) **CVE-2006-0034, CVE-2006-1184**: A vulnerability in MSDTC could allow remote code execution.
9) **CVE-2005-2119, CVE-2005-1978, CVE-2005-1979, CVE-2005-1980**: A vulnerability in MSDTC could allow remote code execution.
10) **No CVE number**: Arbitrary code can be executed on the remote host due to a flaw in the LSASS service.

This list would lead to the installation of 19 CVE-specific and 1 application-specific intrusion prevention services. But multiple CVE-specific intrusion prevention services contain the same set of attack signatures, so that the total number of services to install is actually 11. For example, vulnerabilities

CVE-2003-0715, CVE-2003-0528 and *CVE-2003-0605* (item 3 in the listing), the ones exploited in the attack, require one and the same intrusion prevention service, containing two attack signatures. FIDRAN automatically loaded the demanded intrusion prevention service; repeated attack attempts failed as FIDRAN recognized and filtered the attack. A default configured Snort system would check at least 76/810 signatures for each TCP packet addressed to the victim's destination ports 135/445—leading to a much higher false positive rate. Of course Snort could also be manually configured like FIDRAN, but that would require checking the applicability of each of the 886 rules by hand, and then keeping it up-to-date—clearly a much more expensive and time-consuming task for a single security hole.

Altogether, nine out of ten detected vulnerabilities can be protected by FIDRAN. It fails to protect the task scheduler (Vulnerability 7 of the above listing) because Snort does not provide adequate attack patterns (neither for CVE-2004-212 nor for the Task Scheduler application).

V. SUMMARY

In this paper we presented an intrusion prevention overlay network that autonomously gathers network knowledge to configure itself. Regular active scans are used to discover running hosts and identify their vulnerabilities; in addition, new hosts/applications are immediately identified by traffic selectors on the look out for new destination addresses. The information gathered is then used by FIDRAN to place the required intrusion prevention services—each service provides protection for a concrete vulnerability or application—in front of the end-system that need them. The principle of demand-driven intrusion prevention reduces the amount of security checks that are performed per flow and, as a consequence, reduces also the overall false-positive rate.

Based on a concrete vulnerability, we demonstrated that FIDRAN is able to protect against known attacks which exploit a vulnerability that can remotely be identified. The protection provided by the FIDRAN overlay network is limited by the availability of adequate protection mechanisms and the capability of the integrated network scanner to accurately identify vulnerabilities: end-systems remain vulnerable in case that the network scanner fails to identify security holes (false negatives), and superfluous intrusion prevention services are installed in case that Nessus reports vulnerabilities that actually do not exist (false positives). Unfortunately, identifying all false negatives is not possible per definition. Nevertheless, a more thorough study of the accuracy of Nessus would identify false positives.

Finally, an upcoming creation of *vulnerability signatures*—signatures that matches all exploits of a given vulnerability [4]—facilitates the matching between vulnerability and defense mechanism.

REFERENCES

[1] Common Vulnerabilities and Exposures. http://cve.mitre.org/cve/downloads/allcves.html.

[2] S. Axelsson. The base-rate fallacy and its implications for the intrusion detection security. In *Proc. of 6th ACM Conference on Computer and Communications Security*, Singapore, 1999.

[3] Steven M. Bellovin. A technique for counting natted hosts. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 267–272, New York, NY, USA, 2002. ACM Press.

[4] David Brumley, James Newsome, Dawn Song, Hao Wang, and Somesh Jha. Towards automatic generation of vulnerability-based signatures. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 2–16, Washington, DC, USA, 2006. IEEE Computer Society.

[5] CERT. W32/blaster worm. http://www.cert.org/advisories/CA-2003-20.html, August 2003.

[6] F. Cuppens. Managing alerts in a multi-intrusion detection environment. In *ACSAC '01: Proceedings of the 17th Annual Computer Security Applications Conference*, page 22, Washington, DC, USA, 2001. IEEE Computer Society.

[7] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Comput. Networks*, 31(9):805–822, 1999.

[8] R. Deraison, H. Meer, R. Temmingh, C. v. d. Walt, R. Alder, J. Alderson ando A. Johnston, and G. A. Theall. *Nessus Network Auditing*. Syngress, 2004.

[9] Jeff Forristal and Greg Shipley. Vulnerability Assessment Scanners. *Network Computing*, Jan. 2001.

[10] Fyodor. Remote os detection via tcp/ip stack fingerprinting. http://www.insecure.org/nmap/nmap-fingerprinting-article.html.

[11] Fyodor. The Art of Port Scanning. *Phrack Magazine*, 7, 1997.

[12] A. Hess, H. F. Geerdes, and R. Wessäly. Intelligent distribution of intrusion prevention services on programmable routers. In *Proc. of 25th IEEE INFOCOM*, Barcelona,Spain, May 2006.

[13] A. Hess, M. Jung, and G. Schäfer. Fidran: A flexible intrusion detection and response framework for active networks. In *8th IEEE Symposium on Computers and Communications (ISCC'2003)*, Kemer,Antalya,Turkey, July 2003.

[14] G. Jakobson and M. D. Weissman. Alarm correlation. *IEEE Network Magazine*, 7:52–59, Nov. 1993.

[15] Klaus Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information and System Security*, November 2003.

[16] Klaus Julisch and Marc Dacier. Mining intrusion detection alarms for actionable knowledge. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 366–375, New York, NY, USA, 2002. ACM Press.

[17] C. Kruegel and W. Robertson. Alert Verification Determining the Success of Intrusion Attempts. In *1st Workshop on the Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, Dortmund, Germany, Jul. 2004.

[18] C. Kruegel, F. Valeur, and G. Vigna. *Intrusion Detection and Correlation: Challenges and Solutions*, volume 14 of *Advances in Information Security*. Springer, 2005.

[19] Robert Mathonet, Herwig Van Cotthem, and Leon Vanryckeghem. Dantes: An expert system for real-time network troubleshooting. In *IJCAI*, pages 527–530, 1987.

[20] H.D. Moore. The metasploit project. http://www.metasploit.com.

[21] Kevin Novak. VA Scanners Pinpoint Your Weak Spots. *Network Computing*, Jun. 2003.

[22] Phillip A. Porras and Alfonso Valdes. Live traffic analysis of TCP/IP gateways. In *Internet Society's Networks and Distributed Systems Security Symposium*, March 1998.

[23] The Honeynet Project and Research Alliance. Know your enemy: Tracking botnets. `http://project.honeynet.org/papers/bots/`, March 2005.

[24] E. Rescorla. Security holes...who cares? In *12th USENIX Security Symposium*, pages 75–90, Washington, DC, USA, Aug. 2003.

[25] Umesh Shankar and Vern Paxson. Active mapping: Resisting nids evasion without altering traffic. In *IEEE Symposium on Security and Privacy*, pages 44–61, 2003.

[26] Joel Snyder. Taking aim. *Information Security*, Jan. 2004.

[27] Sophos. Sophos charts virus activity for first six months of 2005. `http://www.sophos.com/pressoffice/pressrel/uk/midyearroundup2005.html`, July 2005.