

Improving Security Management through Passive Network Observation

Yohann Thomas, Hervé Debar

France Télécom R&D

42 rue des Coutures

F-14000 Caen

{yohann.thomas,herve.debar}@orange-ftgroup.com

Benjamin Morin

Supélec

Avenue de la Boulaie - BP 81127

F-35511 Cesson-Sévigné

benjamin.morin@supelec.fr

Abstract

Detailed and reliable knowledge of the characteristics of an information system is becoming a very important feature for operational security. Unfortunately, vulnerability assessment tools have important side effects on the monitored information systems. In this paper, we propose an approach to gather or deduce information similar to vulnerability assessment reports, based on passive network observation. Information collected goes beyond classic server vulnerability assessment, enabling compliance verification of desktop clients.

1 Introduction

For many purposes, and for security in particular, obtaining a faithful, up-to-date and detailed view of the characteristics of a given information system is extremely important [1,2]. This has been the subject of years of research in system and network management, and multiple products provide inventory information for information systems and networks. Most of these products rely on the installation of software agents on the monitored hosts, that periodically report configuration information to a central inventory database. Unfortunately, these software agents may not be deployed in all hosts, particularly in hosts that are not maintained by a central IS department.

More recently, vulnerability assessment tools such as *nessus*¹ or network scanners such as *nmap*² have completed the library of system information mapping tools by providing both the capability to remotely (*i.e.* without local agents) analyze the configuration of devices, and to include information related to the security vulnerabilities that may exist on the remote device. In fact, vulnerability scanners are often mentioned as companion tools to inventory track-

ing, allowing the discovery of previously unknown or unspecified hosts or services.

However, active network mapping suffers from several drawbacks. First of all, active mapping does have side effects on the monitored information system; apart from the network traffic overhead induced by the approach, the performed tests sometimes turn out to be harmful for the environment, as they may crash hosts or services (*cf.* tests explicitly labeled *destructive* in *nessus*). Secondly, active network mapping approaches take advantage of a stimuli database in order to discover target host properties. Thus, only stimulated resources shall be discovered by active network mapping systems. On the opposite, since passive mapping does not stimulate hosts, any resource requested by a third party shall be observed, without the passive mapping system being specifically designed to discover every each resource. Moreover, discovery and updates of characteristics with active mapping systems are strongly linked with the time the stimuli are triggered. In fact, active mapping has the unfortunate side effect that, for example, transient ports may not be open at the time the stimulus occurs, which makes discovering rootkits and Trojan horses difficult. A similar issue appears with hosts presenting a low uptime, that is hosts which are likely to be on only for a few seconds or minutes. Indeed, in such cases, although active mapping may report characteristics by chance, passive mapping is to get characteristics with a high probability provided a communication occurs. Moreover, due to the quantity and the diversity of the stimuli, scanning an entire large network is a long task, so the discovered characteristics rapidly become out-of-date. Lastly, from an ethical point of view, the use of an aggressive technique as a network management tool is questionable.

These observations have prompted the development of several tools that provide topology information gathering through passive network observation. While there are several open source or commercial tools available, it remains unclear whether this category of tools provides benefits for operational security, how to best make use of them, and how

¹<http://www.nessus.org>

²<http://www.insecure.org/nmap/>

to integrate them with the security tools already in place.

This paper is organized as follows. Section 2 presents related work. Section 3 gives our requirements. In section 4, we describe the features we are interested in, and how they are gathered. Section 5 lays emphasis on the feature inference capability of the proposed approach. Section 6 shows how the characteristics provided by passive mapping systems can be used to correlate intrusion detection alerts. Section 7 describes our prototype implementation, followed by experimental results, in section 8. Finally, before concluding and evoking future work, we discuss issues with the approach in section 9.

2 Background and related work

Several tools exist to perform passive network mapping, like p0f³, Siphon, Ettercap⁴, Sourcefire RNA⁵ or Nevo⁶. However, surprisingly enough, few publications on this topic have been proposed.

In the intrusion detection context, Sommer and Paxson [9] as well as Lippmann [10] and Morin *et al.* [7] support the idea of using contextual information about the monitored environment to enhance the diagnostic capabilities of network-based intrusion detection systems. The approach basically consists in comparing the requirements of a target for an attack to be successful with the actual host characteristics. This allows to assess the real severity of the alerts provided by IDSes.

Approaches developed in these papers focus on alert enhancement using host characteristics, not on the way these characteristics are gathered. This is our primary contribution in this paper. Our second contribution concerns other means to correlate alerts with context.

In [3], Dayıođlu discusses the benefits of the information provided by passive network mapping tools to enhance the alerts triggered by misuse network-based intrusion detection systems. However, the author focuses on the host operating system only. Although this characteristic is quite relevant, we believe the approach can be extended to other characteristics, including applicative ones.

3 Requirements

We defined the following requirements for the characteristics gathered using our prototype implementation to fit within our operation constraints:

Comprehensiveness: the tool must be able to extract all the information available in network traffic. This in-

³<http://lcamtuf.coredump.cx/p0f.shtml>

⁴<http://ettercap.sourceforge.net>

⁵<http://www.sourcefire.com/products/rna.html>

⁶<http://tenablesecurity.com/nevo.html>

cludes the capability to follow at least TCP sessions, and possibly application sessions.

Timestamping: the tool must be able to timestamp and update detected features as they evolve periodically. Note that the issue with passive data acquisition is that event generation is not controlled by our tools – we rely on external interactions. Hence we cannot refresh information easily.

Deduction: the tool must be able to infer additional characteristics from the available observed information.

Consistency: the tool must be able to manage information consistency. If two information are in conflict, the deduction process should alert the operator to resolve the conflict.

Innocuousness: the tool must not introduce any additional traffic on the network. For example, ARP cache poisoning or man-in-the-middle traffic redirection were considered unacceptable in our tests.

4 Detected features

The background of our experiment is the desire to extract, from observation of network traffic, the features of our information system. In this section, we enumerate the feature types we are interested in and discuss their availability in the distinct layers of network protocol stacks.

4.1 Features of interest

The features of interest may be divided in three categories satisfying the *comprehensiveness* objective:

Host identification Reliably identifying hosts is a prerequisite for a network mapping system, but it is not a trivial task in complex and dynamic environments. IP addresses available in network packets are naturally good candidates for this purpose, but their lifetime is limited in environments where hosts dynamically acquire their address. Host names are more reliable in the long term but they are not available in the packets. Name resolution could be achieved by the mapping system using DNS requests but it would not be passive anymore. In fact, name resolution should take place in a third-party component in charge of the host identification task (possibly using DNS requests captured by a passive mapping system).

Operating systems Ambiguities in the specifications of the TCP/IP protocol lead to different network stack implementations in operating systems, the specifics of which are used to recognize the operating systems of

the hosts involved in the packet headers. The operating system feature is of special interest in our context because it is discriminant to assess the vulnerability of a host.

Software We are interested in gathering as detailed descriptions of the software running on the hosts as possible: their name, version and category (*e.g.* web server, proxy, web browser). The TCP layer reveals open ports which are further translated to software categories. Acquiring the name and version of products requires the application layer to be investigated. For instance, HTTP replies contain the web server name and version, as well as the use of specific modules such as php; URL contained in HTTP requests can reveal the existence of a cgi script on web servers, depending on the answer of the server.

Contrary to active mapping techniques, passive mapping is not limited to the discovery of software which listen on an open port; client-side software characteristics can also be discovered. For example, browser model and the operating system of a client can be found in HTTP requests. This information is of importance to monitoring, compliance testing and vulnerability assessment because malicious content often uses browser vulnerabilities to infect client machines and bypass firewalls or other access control mechanisms.

The characteristics of the applications should be considered with caution because they can be masqueraded. For example, the Opera browser allows users to change the agent identification banner sent to the web servers. However, this mutation is less likely in corporate environments.

4.2 Negative features

Network mapping systems only provide an incomplete and blurry picture of the monitored network hosts. In order for mapping systems to provide observation as close from the reality as possible, it is necessary not only to detect and keep track of positive features (*i.e.* features which really exist on a host), but also negative ones (*i.e.* features which do not exist on a host). Indeed, a positive feature might not have been stimulated by an active mapping system, or might not have had any side effect on the network in the case of passive mapping systems. In other words, a non-detected feature is a feature which *might* exist on a host, not necessarily a negative feature. Thus, in order to distinguish the former from the latter, it is necessary to capture explicit signs of negative features.

Active mapping systems generally only provide detected positive features. However, since the list of stimuli is known and finite, and if we assume that an invalid response to a

stimulus is equivalent to a negative feature, it is possible to infer detected negative features from the set of detected positive ones and the complete set of stimuli performed.

In passive mapping systems, negative features have to be searched explicitly in the network traffic because one may not infer anything from a packet that does not match a given rule (*e.g.* one may not state that a machine does not host an Apache server because the corresponding rule does not apply to an ICMP packet). This can be achieved at different network layers, by capturing *e.g.* ICMP “host unreachable” messages to detect the absence of hosts, failed TCP connections to detect the absence of a service on a host, or 404 replies from web servers to detect the absence of a web service for instance.

This impacts both the *deduction* and the *consistency* objectives of our implementation. For example, mutations in the response offered by the HTTP server to a vulnerable script request (*e.g.* from 404 to 500) need to be interpreted manually – while the response change on the server indicates a configuration change, we cannot infer why this configuration change occurred, as it could be the result of a compromise. Therefore, the implemented prototype only includes clear-cut responses from servers, which limits the deduction capability for the moment.

5 Property inference

In this section, we put emphasis on the additional analysis performed over the raw observation provided by the passive mapping sensors in order to enhance the detected features, and thus fill the *deduction* and *consistency* objectives.

5.1 Feature inference

The feature inference is a classical expert-system like process, which consists in deriving previously undetected features from existing ones and ontological expert rules. One example of such expert rule is to infer that a host’s operating system feature is Windows, if an IIS web server has been detected on this host because IIS is known to work only on Windows platforms.

5.2 Feature consistency checking

This analysis consists in detecting inconsistencies in the observed and derived features. These inconsistencies shall not be resolved by the system, but they are reported to the operator in charge of the monitoring of the network. Two distinct web servers hosted by the same machine on the same port at the same time is an example of such inconsistencies. Note that time is a crucial issue in the consistency resolution process. Indeed, one has to define the meaning

of "at the same time", considering the fact that a feature may have changed till the last observation. For example, a *normal* situation appears when machines hosting several operating systems have had enough time to reboot between the detections of the two features.

5.3 Disruptive components recognition

Some specific network components are disruptive in the sense that their normal behavior in the network leads passive network mapping sensors to draw false conclusions about the detected features. As an example, from a passive network mapping sensor point of view, and depending on the sensor's location, a machine acting as a web proxy *seems* to be using *many* web browsers and/or web servers. This is not surprising since web proxies are used to relay internal users http requests to the outer world, but from a sensor point of view, the proxy either acts as a http server or client in the analyzed requests.

The goal of the disruptive component detection process is to recognize such components from previous observation, remove the false features and provide a new observation about this host's newly detected feature (*e.g.* web proxy).

6 Alarm Correlation

Our purpose with cartographic information is to enable alert correlation. With a detailed view of the vulnerabilities existing on an information system, it is possible to adjust the priority of alerts generated by intrusion detection systems, according to the actual risk of successful intrusion [7].

The knowledge of services offered by the information systems allows us to (1) detect configurations that frequently trigger false alarms and (2) adjust alert priorities according to the existence of vulnerable services.

6.1 Correlation with vulnerabilities

Vulnerability assessment tools associate hosts and vulnerability references. This association indicates that the host is vulnerable to attacks exploiting the referenced vulnerability. Most intrusion detection systems also associate alerts with vulnerability references. A very simple and classic schema for prioritizing alerts is to increase the severity of alerts when signatures and hosts share common references.

Network mapping observation does not immediately offer the possibility to link vulnerability information and hosts. This linkage is obtained using additional information extracted from vulnerability databases such as the Open Source Vulnerability Database⁷ (OSVDB) or ICAT⁸. These

databases link service names and versions with vulnerability references. Our current implementation uses OSVDB. We use the `object_correlation` table of the OSVDB database to obtain version information, and the `ext_ref` table to associate the product with vulnerability references as found in the intrusion detection signatures.

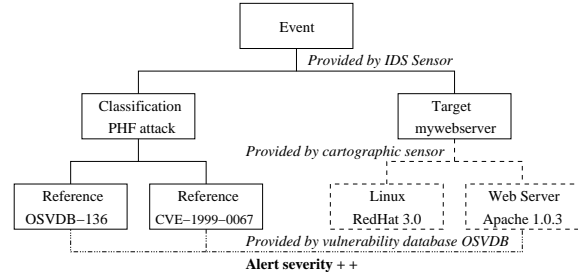


Figure 1. Correlation with vulnerabilities

An example of this correlation component is shown in Figure 1. My web server is an apache 1.0.3 on Linux Red-Hat, identified by the passive mapping sensor. The intrusion detection system sends an alert with the web server as target and the PHF message as classification. Using the information from OSVDB, one can easily conclude that the web server is vulnerable to the attack.

To successfully build this correlation component, one needs to associate the *(productname, productversion)* characteristics provided both by the network mapping database and by the vulnerability database. This is a challenge as the information that transits on the network is different from the one stored in the vulnerability database. For example, "Microsoft Internet Information Server" is known as "Microsoft IIS" in the vulnerability database. Also, many versions of Windows are known both by a product name (Windows NT, Windows XP) and by a version number that is not explicitly related to the application, but that is shown in the traces collected from the network.

6.2 Alert severity mitigation

Hosts characteristics provided by passive mapping systems can also be used to mitigate alerts severity. This is achieved by comparing the requirements of a target for an attack to be successful with the actual host characteristics. For instance, an attack exploiting a flaw in a Microsoft IIS product launched against a host which is known to use a Linux operating system will not work. Thus, the severity of the alert can be mitigated.

An example of this correlation component is shown in Figure 2. My web server is an apache 1.3.27 on Linux Red-Hat, identified by the passive mapping sensor. The intrusion detection system sends an alert with the web server as target

⁷<http://www.osvdb.org/>

⁸<http://icat.nist.gov>

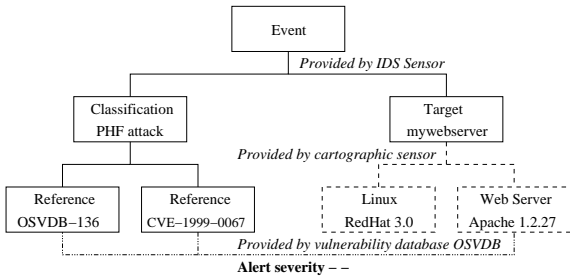


Figure 2. Alert severity mitigation

and the PHF message as classification. Using the information from OSVDB, one can easily conclude that the web server is *not* vulnerable to the attack.

6.3 False positive recognition

As suggested by Julisch [5] or Manganaris [6], the excessive amount of alarms triggered by intrusion detection systems is mainly provoked by false positives (*a.k.a.* false alarms). Most of these false positives come from activities misinterpreted as attacks by IDSes. These activities are induced by network components whose normal behavior includes misuse-like side effects on the monitored environment. Since misinterpreted activities are recurrent, the resulting alarms have a strong impact on the overall number of alarms. By comparing the characteristics of the hosts involved in the alarms with the configurations which are prone to false positives, it is possible to recognize alerts which are false positives.

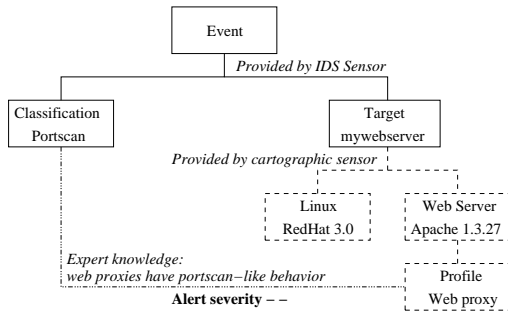


Figure 3. False positive recognition

An example of this correlation component is shown in Figure 3. The passive cartography sensor identifies my web server, and given the parameters in the HTTP request, is able to identify that this web server is acting as a proxy. The proxy profile is therefore attached to the web server. This proxy profile is known to generate portscan false alarms, hence the severity of the event is reduced.

7 Implementation

7.1 Prototype description

Following the requirements discussed in section 3, we defined a software package based on freely available tools. In fact, we realized early that network-based intrusion detection systems would satisfy most of these requirements, provided that we use a different signature set than the one delivered in standard. The resulting testbed therefore includes two open source tools, Bro (network intrusion detection system with custom scripts) and Ettercap (using only the passive monitoring part, to satisfy the *innocuousness* objective). The architecture of the prototype is described in Figure 4.

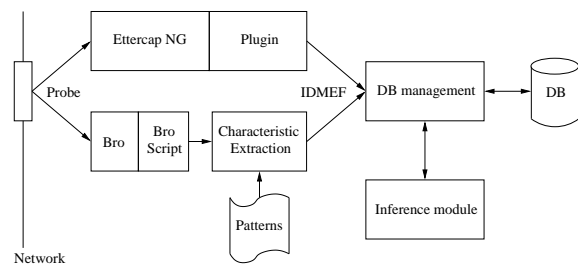


Figure 4. Architecture of our prototype

Ettercap NG⁹ is used for passive OS fingerprinting, host names and services discovery. It performs accurate OS fingerprinting, using a high number of metrics used to characterize TCP/IP stack implementation differences. Its OS fingerprints database currently contains about 1600 entries. Host names discovery is based on DNS protocol analysis and software categories are deduced from the observation of TCP SYN and ACK flags. Ettercap NG is interfaced to our environment using an internally written output plugin.

When used in passive mode, Ettercap NG cannot reconstruct network sessions. Searching patterns in network sessions can be done with existing misuse NIDSes, whose signature database is designed to extract the host characteristics of interest. Bro [8, 9] is used for this purpose because it provides a flexible language for expressing relationships between events.

Regular expressions are necessary to extract product names and versions from the analyzed events. Bro currently only provides string matching. Thus, Bro is used to reconstruct sessions, decode applicative protocols and store the packet payload. Pattern extraction is delegated to the *characteristic extraction* module which uses a set of regular expressions to extract characteristics of interest from the applicative payload.

⁹<http://ettercap.sourceforge.net>

The role of the *inference module* is to complete the characteristics gathered by the passive network mapping system by inferring new facts, based on a database of expert system rules and some background knowledge. For example, it should be able to deduce that the operating system is Windows if an Internet Information Server web server has been observed on a host.

The *database management* module interfaces the Ettercap and Bro tools with the rest of our security information management platform, described in [4].

The data structure adopted to store the observed characteristics is defined as follows. A product is a triple (n, v, c) where n , v and c respectively denote a product name, version and category. A profile is a 4-uplet (h, p, t, f) where h , p , t and f respectively denote a host identifier, a product, a timestamp, to fill the *timestamping* requirement, and a flag.

Product names and versions are mainly used for vulnerability assessment (see section 6). Product categories are used to recognize host function (router or proxy for instance) whose behavior provokes false positives. The product of a category is the only mandatory attribute; names and versions are not always available to the passive network mapping system. For example, (Apache, 1.3.27, http-server) and (Linux, _, OS) are correct products.

Intuitively, profiles connect host with products. Host identifiers are either identified by their IP address or by their host name. The timestamp represents the last time a product was observed running on a host. It is necessary to keep track of the product versions changes in order to assess a host vulnerability at the time an attack is detected, which may be prior to a software update. Thus, in case of a change in the version of a product, the prototype adds a new triple instead of updating the product of a profile. When no change in the product version of a profile is detected, the profile's date is updated.

The flag is a boolean used to say whether a host feature is negative or positive. This attribute is necessary to distinguish characteristics which are known not to exist on a host from characteristics for which we actually do not know.

7.2 Installation in network

As the tool is based on observing network traffic, one needs to maximize the amount of traffic seen by the tool to ensure that most of the information system is monitored and mapped. The deployment strategy is similar to a network intrusion detection sensor, attaching the passive network sensor to a hub or a SPAN port on a switched network. As usual, one needs to carefully choose the SPAN port location to ensure that most of the interesting traffic is captured. Fortunately, there are two interesting locations where passive network sensors can be deployed profitably.

The first location is the entry to server farms. In many

large enterprises, the servers are grouped together in a particular location, and access to this location is through a couple of switches or routers. Therefore, it is quite easy to listen to a major part of the internal traffic by locating the sensor close to the main company servers, either internal web servers, file servers, or mail servers. Taking the example of the France Télécom environment, sensors deployed in less than 20 locations will monitor the majority of internal network traffic and provide inventory information for the vast majority of clients and servers.

The second location is proxy servers. Proxies mediate a lot of application traffic between internal clients and the rest of the world. Therefore, SPANning the proxy port is also a possible and effective strategy for capturing interesting information, providing that the difference between internal and external sensors is defined with simple heuristics.

8 Experiments

The prototype has been installed in our exploitation network, on a machine that has the capability to access the complete network traffic (SPAN). Five captures of 2 minutes have been realized in the day, during working time. The experimentation first results in a quantitative assessment of the traffic volume. Then, we proceeded a detailed analysis of the captured traffic with the passive network mapping system prototype.

8.1 Quantitative assessment of network traffic

Table 1 shows traffic volume across captures, split according to the most important protocols.

Port	Capture number					TOTAL
	1	2	3	4	5	
389 (ldap)	355520	36125	131394	39572	120838	12,4%
445 (microsoft-ds)	220473	116966	76717	14862	61116	8,9%
137 (netbios-ns)	2914	2774	6320	2123	2918	0,3%
138 (netbios-dgm)	1034	1504	1632	778	1106	0,1%
139 (netbios-ssn)	165061	95470	292589	199770	657149	25,5%
138 (loc-srv)	1001	1193	1577	782	935	0,1%
53 (domain)	4041	2752	2082	1881	6888	0,3%
80 (www)	29149	16823	41582	8429	18106	2,1%
3128 (www)	58234	118715	51201	64724	145383	7,9%
8080 (www)	142	137	807	112	2701	0,1%
81 (www)	4769	9879	20129	2211	2757	0,7%
1433 (ms-sql-s)	40830	58858	105801	63543	42099	5,6%
5903 (vnc)	72700	75423	73749	1195141	0	25,6%
Others	137891	174465	96099	65261	97135	10,4%
Packets total	1093759	711084	901679	1659189	1159131	100%

Table 1. Traffic volume

The two most important protocols represented here are LDAP and netbios-ssn. These two protocols are representative of windows-based networks using active directory. With the other Microsoft-related protocols, they represent almost half of the activity occurring in our environment.

The second largest share of network activity is related to VNC (Virtual Network Computing) activity. This activ-

ity occurs always when desktops interact with one specific server. VNC here is used to support linux desktops accessing Microsoft applications.

Finally, the third largest share of network activity, with about 11%, is related to HTTP traffic, either internal (ports 80 and 81) or external (going through proxies on ports 3128 and 8080). The existence of proxies provides an easy way to distinguish internal and external activity, although this external activity may not be HTTP-only, as the same port is used to proxy FTP as well.

Finally, there remains about 10% of traffic which is not characterized. We expect this to include video-conferencing and multicast, which are commonly used in experiments in our environment. A complete analysis of this remaining network traffic still needs to be performed.

8.2 Passive network mapping results

While HTTP traffic represents a minority share of the total traffic, it is almost the only protocol going through our firewall, and as such is a primary target for detailed characteristics extraction. Going further, we proceeded a detailed analysis using the passive network mapping system prototype. Thanks to Ettercap NG and our plugin, we were able to get host names, OSes and services based on open ports. We also deployed a few rules for the data extraction module linked with Bro, in order to collect server, client and CGI information on HTTP traffic, as shown below.

```
SEARCH Server:\s([\s\^]+) IN rep_payload AS httpServer;
SEARCH User-Agent:\s([\s\^]+) IN req_payload AS httpClient;
SEARCH \/scripts\/(.*)\.cgi IN uri AS cgiScript;
SEARCH \/cgi-bin\/(.*)\.cgi IN uri AS cgiScript;
```

We provide in table 2 the results given by the network mapping process, listing the number of occurrences of relevant information.

Identified objects	Capture number					TOTAL
	1	2	3	4	5	
Hosts	344	342	323	274	322	742
Products	165	116	150	93	118	223
Runs	927	907	877	635	879	2162
Names	35	24	32	18	20	63
OSes	24	24	26	22	25	31
Web servers	39	35	34	38	47	71
Web clients	15	14	11	11	16	24
CGIs	1	0	0	0	0	1

Table 2. Prototype results

Given these experimental results, we first notice that aggregating the five captures of 2 minutes allows us to associate 742 different hosts to 223 different products within 2162 *runs* relationships (i.e. profiles). The results are as follows:

- Of the 742 hosts, 63 could be named (about 8%). This is considered small in our environment, since DHCP updates DNS information whenever a client obtains an address.
- Among the 223 products, 31 are OSes, 71 are web servers, 24 are web browsers and one is a CGI application.
- 702 hosts have been assigned at least one OS, that is to say a significant number (about 95%), considering that only a few packets have been captured for some hosts.
- Among the 2162 *runs* relationships, 760 associate OSes with hosts. All the other relationships represent installed applications or services on the hosts.

Note that all applications and services collected characterize properties existence on the corresponding hosts, that is the prototype was not able to discover non-existence of properties.

The large number of different web servers and operating systems (and web clients to a lesser extend) is due to the fact that product identification includes both name and version. This means that some analysis, currently manual, needs to be performed to extract version information from the product identification.

Among the 760 *runs* relationships concerning operating systems, 2 machines are associated with four operating systems, 4 are associated with three operating systems, and 43 are associated with two operating systems even in our short observation period. In fact, some OS fingerprints are very similar, which can result in the matching of multiple signatures for a same OS. However, what we face here is mainly a signatures reconciliation issue, since in most cases, Ettercap gives in fact the same OS, but with a different semantic in the denomination. This case potentially represents 35 machines among 49 being characterized with at least two OSes. Careful analysis for 8 of the 14 other machines shows that the results are acceptable and there would have been enough time in the observation period to reboot these machines from one operating system to another.

Note that a limitation currently deals with applications assigned to proxies, since they represent the source IP address when they relay packets. Therefore, a high number of *runs* occurrences concerning web servers are ascribed to web proxies. The system should be able in a next future to take into account host functions, in order to avoid this kind of mistakes.

Finally, we note that for 32 machines, we obtained detailed information combining host name, operating system and web client or server (or at least a service characterizing the host function).

While the results are preliminary, they are encouraging enough for us to attempt to apply them to alert correlation in an intrusion detection environment.

9 Issues and future work

A first issue with passive network observation is related to the fact that internal traffic may use different ports than the official ones. For example, HTTP traffic in our internal network occurs not only on port 80, but also on port 81 (internal web servers), 3128 (web proxy), 8080 (old web proxy) and 8888 (Sun service documentation). Therefore, there is a need for an automatic protocol discovery and verification module, to ensure that network traffic is properly associated with the underlying protocol independently of port information available in network packets.

Another challenge in this area is the implementation of a reliable protocol analyzer for microsoft protocols. While service platforms rely heavily on well known web protocols for which protocol analysis tools are widely available, internal traffic relies heavily on file sharing protocols such as CIFS and on direct connections between Microsoft Exchange and mail clients. This traffic is currently not analyzed, which indicates that we are missing a lot of information related to our client machines.

Finally, although we obtained encouraging results, we are to admit that passive network mapping suffers from limitations concerning temporality. In fact, properties are updated only thanks to interactions between hosts, that is to say it is challenging to ensure a consistent and up-to-date view of host properties using only the passive acquisition approach. Therefore, we wonder whether adding a few active functionalities with minor side effects on the monitored information system could provide a significant gain.

10 Conclusion

This paper has presented a tool for acquiring vulnerability assessment information through passive network observation. The proposed approach has no side effect on the monitored environment, and ensures frequent renewal of the most frequently used tools in the information system. Passive network observation extends this monitoring to desktops, including information about web browsers that may not conform to the specifications of the information system.

The information collected has the ability to support alert correlation through vulnerability assessment as it is currently performed in most security information management platforms. With respect to the state of the art, the proposed technique is non-intrusive and offers the capability to collect information from the client side of the connection, that is currently not available. We are currently working on qualifying the expected gain in terms of correlation, both from

a theoretical standpoint (how much of the information collected can be used in correlation) and from an experimental standpoint (are alerts collected in running systems impacted by this additional information).

References

- [1] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, and A. Silberschatz. Topology discovery in heterogeneous ip networks: the netinventory system. *IEEE/ACM Transactions on Networking*, 12(4):401–141, June 2004.
- [2] Y. Breitbart, M. N. Garofalakis, C. Martin, R. Rastogi, S. Shadri, and A. Silberschatz. Topology Discovery in Heterogeneous IP Networks. In *INFOCOM (1)*, pages 265–274, 2000.
- [3] B. Dayıođlu and A. Özgıt. Use of passive network mapping to enhance signature quality of misuse network intrusion detection systems. In *Proceedings of the Sixteenth International Symposium on Computer and Information Sciences*, November 2001.
- [4] H. Debar, B. Morin, V. Boissée, and D. Guérin. An infrastructure for distributed event acquisition. In V. Broucek and P. Turner, editors, *Proceedings of Eicar 2005*, Malta, April 2005. EICAR.
- [5] K. Julisch. Mining Alarm Clusters to Improve Alarm Handling Efficiency. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*, December 2001.
- [6] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz. A Data Mining Analysis of RTID Alarms. In *Computer Networks*, number 4, pages 571–577, 2000.
- [7] B. Morin, L. Mé, H. Debar, and M. Ducassé. M2D2 : A Formal Data Model for IDS Alert Correlation. In *Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID)*, October 2002.
- [8] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23–24):2435–2463, Dec. 1999.
- [9] R. Sommer and V. Paxson. Enhancing byte-level network intrusion detection signatures with context. In *Proceedings of ACM CCS*, Washington, DC, USA, October 2003. ACM.
- [10] S. Webster, R. Lippmann, and M. Zissman. Landscape: A passive network mapping system. MIT Seminar, 2003.