Advanced Widespread Behavioral Probes against Lateral Movements

Alessandro Greco[1], Giovanni Pecoraro[1], Alberto Caponi[1,2], Giuseppe Bianchi[1,2]
[1] University of Rome "Tor Vergata", [2]CNIT
Italy

Abstract

The fast evolving nature and the growing complexity of modern offensive techniques used in Advanced Persistent Threats attacks call for innovative approaches for defense techniques. The ability of modern offensive operations to acquire a foothold and then expand an infection inside the victim's local area network, usually referred to as lateral movement activity, is significantly critical. Not only a distributed monitoring infrastructure is necessary to overcome the lack of a single network point for detection (opposed to the traditional network perimeter defense relying on outbound network intrusion detection systems), but also new signatures appear necessary to model the stealthy and complex behavior of offensive lateral movement activities. In this paper we demonstrate how to effectively use eXtended Finite State Machine patterns to face a set of commonly used lateral movement techniques. With reference to real world lateral movement attacks (including those ones based on IP spoofing), we show how the relevant detection signatures can be gathered and formally modeled, also employing a widespread distributed security architecture. Numerical results on real world traces show the effectiveness of the proposed approach in avoiding false positives.

1. Introduction

Advanced Persistent Threats (APTs) are the emerging cyber threats well known for using complex attack techniques and concealing their presence in the network for long periods of time [1]. Despite the crucial attention and intense research effort spent by the cyber-security community, the time it takes to detect APTs is still measured in days or even months, with an average [2] of as much as six months. The main reason for this detection gap is the sophistication of infection and evasion techniques used by attackers, in most cases operating without any specific automatic tool (easily detectable, as traditional virus).

An aspect that makes it hard to defend against APTs is the stealth way in which such attacks spread. Most often, APTs first limit to acquire an internal foothold in a target network, by typically finding a single weak spot and exploiting it to gain access to the network [3]. Once inside, APTs start to spread all around the network, collecting and exfiltrating sensitive data. In spite of such a fundamentally different behavior with respect to the more traditional threats, network operators and target companies are still protecting their networks using defense techniques primarily targeting the protection of the network perimeter, e.g. via outbound Network Intrusion Detection Systems (NIDS). Indeed, a report from the security company Fireye [3] suggests that companies still spend more than $5 billion on traditional security measures, whereas, as reported in [4], they often completely neglect the internal network protection.

Unfortunately, the detection of these internal insidious activities is arduous for several reasons such as the granularity of controls to be performed and, above all, the complexity of indicators of compromise to be implemented, because of the use of administration-like operations and the consequent high false positive rate.

We think that an effective internal network defense, capable of detecting the spreading of infections via lateral movements, requires significant advances in *two complementary directions*. The first direction is related to the monitoring infrastructure. In fact, rather than relying on a centralized outbound NIDS, new generation monitoring systems should rely on the analysis of ***network-level patterns*** through a ***distributed monitoring architecture***, employing multiple (and ideally collaborative) probes in the form of software agents widely available in the network, in principle down to one single probe per device. Note that we foster an approach based on the analysis of network-level events, as this holds the promise to more promptly detect anomalies and end-point misbehavior if compared to the current log-based host analysis (e.g. [5,6]). The second direction, and the main contribution of this paper, consists of identifying ***new attack signatures which are amenable to network-level analysis and support scalable operation***. In fact, the obvious shortcoming of a network-level analysis, and to a greater extent of a widespread distributed infrastructure with multiple vantage points, is the huge amount of traffic to inspect and the inherent complexity in identifying malicious patterns in it. Rule-based approaches (Snort-like [7]), appear too simple to properly model and characterize complex behaviors, such as those involved in APT's internal infection spreads. Current literature approaches, which rely on Snort-like IDSs [8], usually take a two-step approach and thus lose the ability of being able to detect and respond to threats in (near) real time: they log, in real time, network events gathered from Snort probes, and then only later they can parse offline the network logs to identify malicious patterns.

The crucial questions, addressed in this paper, are thus: *how to formally model realistic complex patterns, such as those involved in offensive lateral movements, in a way that they are amenable to be detected via a packet-by-packet analysis*? *how to correlate the warnings generated by the widespread probes in order to detect complex distributed attacks?* Our answer resides in a monitoring architecture (Section 3), whose single agents, intended as local instances of the scalable StreaMon [9] probe, are based on the notion of finite-state-machine-based signature**,** i.e. signatures which are still based on the analysis of low-level packet events (expressible via the classic rules). It is possible if they are able to explicitly model the attack behavior change in time, through the concept of “attack state”, and to formalize such evolution. In addition, in order to detect complex distributed threats (especially those ones aiming at anonimizing the offensive operations), all the warnings are centralized and correlated. The effectiveness of the proposed architecture is evaluated by modeling lateral movements of real threat, by step-by-step showing (Section 4) how it can be formally modeled in a stateful composition of rules which can then be executed over the StreaMon probe [9]. A discussion of related works is then provided in Section 5, while conclusive remarks and further possible evolutions of this work are summarized in Section 6.

2. State of the Art on advanced cyber threats

The traditional defense approach against cyber threats primarily targets the protection of the network perimeter by means of outbound NIDSs. Such a monitoring topology allows the detection of malicious operations between the infected host inside the network (red host in Figure 1, most often called *pivot*) and the external Command & Control entities, but neglects the internal malicious connections between the *pivot* and the other victim hosts. This is a limitation in case of modern APTs, which, once gathered a foothold *inside* the network, aim at spreading the infection inside the LAN, taking control of other internal hosts (orange hosts in Figure 1) without any additional external interaction.

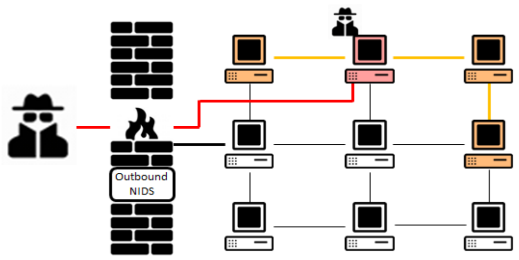


Figure 1. APT scenario

The defense against the lateral movements is complex both at network and host level. In particular, the security devices deployed for this purpose are called Intrusion Detection System (IDS) and can be host-based (Host IDS - HIDS) or network-based (NIDS). The first type is a system installed on single host, mainly dedicated to the monitoring of files integrity, application logs and system calls, while the NIDS attempts to discover any unauthorized access to a computer network by capturing the network traffic [10]. Almost all the detection techniques used for lateral movements are based on the Windows event logs analysis, through HIDSs and centralized SIEM applications. In [5] and [6], for example, in order to detect a pass-the-hash attack it is suggested to implement an host based detection process (using the Windows and antivirus logs in order to monitor the authentication events - comparing the results against an user and/or IP approved list); in addition, a network user habit based detection approach is also proposed (once created a baseline of normal system behavior, the detection techniques look for anomalies on the number of connection in a short period of time on a specific port). A similar and wider approach is introduced in the Microsoft guide dedicated to the mitigation of the pass-the-hash attack [11], where they propose a complete list of anomalous user behaviors, which can be interpreted as stolen credential use, and the relative Windows log events.

This kind of host-based analysis can be performed both in a decentralized and in a centralized way, using SIEM solutions whose high performance allows to implement complex statistical algorithms, as in [12]. For the detection of such threats, HIDS approaches are hence preferred to the NIDS ones, probably because of the use of licit protocols and administration commands, although a network analysis can be performed at run time and integrated with Software Defined Network devices for mitigation. One example of NIDS detection for this threat is proposed in [13] and is based on the conversion of the Windows log indicator. In particular, indicators of lateral movement proposed in (Windows event type and relative information) are converted in SNORT format [7] using regular expressions and text search. A mixed approach, similar to the one of this article, is provided by [4]: the detection algorithm is based on the SMB authentication process chain in a Windows architecture and performs DPI in order to extract credentials.

In this paper we focus on the detection of these threats at network level, which is complex for many reasons. First, there is not anymore a single point in the network such as an outbound NIDS which permits to monitor malicious traffic activities. Second, the attacker employs multiple different techniques in the lateral movement phase. Finally, such techniques are engineered to hide among legitimate traffic and activities, and use standard administration-like operations in this phase. Through the usage of such methods it is difficult to detect the lateral movement, because benign and malicious utilization of these methods looks similar and therefore distinction is complex [14].

A suitable defense against lateral movements thus must face two complementary needs. First, the network monitoring architecture must also spread inside the LAN via the monitoring agents widespread located inside the network. Moreover, since lateral movements are characterized by an administration-like traffic (most often derived by human-driven operations), whose single packets are legitimate events, to prevent false alarms it is necessary to detect how such events occur (and change) in time. As discussed in the next section, we suggest relying on state machines, executable as signatures to be implemented in deployable version of the StreaMon probe [9], called D-Streamon [15].

3. Proposed Monitoring Architecture

The implementation of a widespread monitoring infrastructure requires to deploy and to manage a high number of NIDS agents. Without an automated process, a manual approach does not scale in large networks and in general it is a cumbersome and error prone task. For this reason, we create a capillary monitoring architecture as shown in Figure 2, based on an extensive deployment of lightweight NIDS agents at host level, whose events/warning analysis is centralized by a NIDS Manager, which can correlate data and trigger mitigation actions.

3.1 NIDS Agent

Offensive lateral movements are very complex to be detected with a traditional NIDS monitoring approach, not only for an architecture issue (perimeter monitoring vs distributed monitoring), but also for the difficulty to summarize such attacks into a single matching rule. Despite the growing interest in behavioral-based NIDSs, designed to automatically identify "deviations" from the normal behavior of a host or network [16], the need for their extensive training in conjunction with their statistical (non-deterministic) operation implies that signature-based NIDSs still play a dominant role in the NIDS market. The key problem in signature-based NIDS is the appropriate definition of signatures for the variety of threats recognizable by the system. Emerging threats in lateral movements and APT scenarios are usually very complex, and are hardly summarized into a single matching rule. Rather, they do encompass a multiplicity of serial steps that, if taken alone and detected with atomic IOC by a stateless NIDS, could give many false alarms.

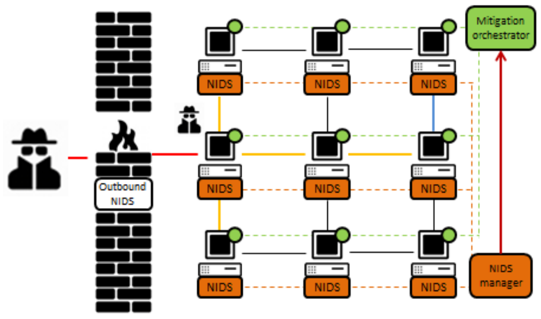


Figure 2. Monitoring architecture

To address such a problem, we propose to incorporate the "state" of a threat into a formal model of the threat signature, so that the detection of different atomic IOCs can be employed at different "stages" of the threat's evolution (summarized by explicit states). It readily follows that an IOC which could be normal or legitimate at a given stage, becomes an indicator of an ongoing attack only when it happens at a different stage, significantly reducing the false positives.

To our surprise, only limited research work has addressed the goal of devising stateful signatures. For instance, [17] promotes such an approach, but restricts it only to layer 3 rules. Conversely, the StreaMon probe proposed in [9] provides a monitoring probe architecture capable of executing and evolving state-machine-based signatures, but gives only a very limited insight on how to model real world threats and whether a state-machine-based signature model is actually suitable for this task.

In [18], we proposed to model behavioral pattern signatures as state diagrams which permit the programmer to ***both*** formalize the notion of "***attack state***", as well as define, for each state, the ***rules which model the state evolution in the detection process***. In details, a finite-state-machine-based signature is expressed in terms of:

- entities: the entity to which as time-varying "state" is associated; it can be a single entity, such as a target IP address, or a pair, such as [IP attacker, IP victim];
- states: for each entity, it formalizes an "attack state" and it associates to each state a possibly different set of features (events or conditions) that shall be monitored. Note that different states may entail the monitoring of different events;
- state transitions: similarly, for each state, one or more state transition rules are associated, again expressed as a combination of events and conditions. This permits to formally specify how the state shall evolve.
- actions: once a transition state is executed, the finite-state machine can also perform actions, such as changing a variable (e.g. counter) or giving a warning to a security alerts collector.

Once the programmer has identified the set of attack states, the main modeling task consists in associating state transitions to matching rules, expressed as a Boolean predicate on events or conditions (as defined below). In this paper we will adopt the notation and syntax shown in the table below.

Transition from <u>State#N</u> to <u>State#M</u>:
(*Event A* ***OR*** *Event B)* ***AND*** *Condition C*
where:
- <u>Event A</u>: description and indicator of the event A
- <u>Event B</u>: description and indicator of the event B
- <u>Cond.C</u>: description and indicator of the condition C
• <u>Action</u>: description of the action

An *event* refers to information that can be detected from the packets, including both information gathered from the packet header as well as information eventually extracted from the packet payload using Deep Packet Inspection (DPI). A *condition* refers to the test of logical operations on collected statistics associated to the entity and/or state (e.g. event counter greater than a threshold, communication direction, etc.). Finally, the signature developer may optionally associate an *action* to each state transition (for instance, send an alert or reset a counter).

Such pattern are then implemented as eXtended Finite State Machine (XFSM) signatures for StreaMon [9] probe, which offers elementary stream-based traffic analysis primitives (e.g. feature extraction, parsing, statistics collection, correlation, etc.), as well as Deep Packet Inspection (DPI) functionalities for particular protocols. In particular, for the proposed capillary and distributed monitoring architecture, we used a lightweight version of StreaMon [9], called D-StreaMon [15], conceived as distributed Network Functional Virtualization (NFV) solution. The events/warning generated at host level by the NIDS agents are sent to the Manager over a publish/subscribe ZeroMQ [19] architecture, allowing the asynchronous dissemination of warnings/alerts.

3.2. NIDS Manager

The Manager performs different tasks. First, it is the master of D-StreaMon architecture and it is committed to initialize the probes, in terms of signatures to be detected. It is also the brain of the security infrastructure able to receive the asynchronous events from the NIDS substrate. Once the warnings are sent by a single NIDS, the aggregated analysis can be performed in several ways, according to the attacks that need to be detected, using statistical or deterministic approaches. Finally, once an attack is clearly detected and attributed, the NIDS Manager triggers mitigation actions, managed by the *Mitigation orchestrator*, which can be integrated also in a Software Defined Network (SDN) architecture.

4. Use Cases of Advanced Detection

Once such capillary monitoring capability is available, it is possible to define and implement a variety of detection logics, which allow to adequately model most of malicious internal activities. In particular, in addition to the detection which can be implemented at host level, we propose to aggregate and correlate the events/warnings generated at host level, in order to detect, identify and attribute more complex attacks, most often aiming at evading the security monitoring checks.

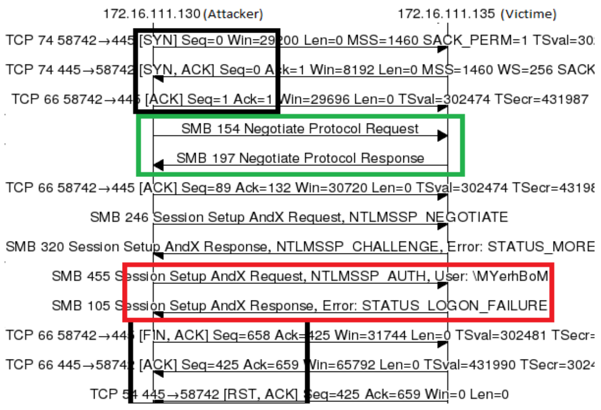
4.1. Complex behavioral signatures for lateral movements

As explained in Section 2, the sophistication of APTs is always increasing: in order to perform their malicious tasks, the attackers continually enhance the offensive techniques, implementing new way to silently spread all around the victim LAN and fulfill the objectives. In Windows domains, most often the lateral movements are based on administrative commands, both to deceive the network security analysts and to exploit the powerful built-in functionalities.

In [18, 20] we show how to use XFSM signatures to effectively model such threats. PSEXEC was the key case study used [18] to prove the effectiveness of the XFSM signatures. It is a widely used technique, which enables the remote control of a host, automatizing the process of copying a software on a shared directory of a second host and then remotely executing it as a service, exploiting the Server Message Block (SMB) and Remote Procedure Call (RPC) protocols functionalities.

In order to identify and then model the effective signatures, we analyzed noise-less traffic samples containing the patterns of interest. In our case, we created a virtual environment, representing a basic two-hosts (one WinXP and one Win7) victim local Domain based on Windows Active Directory, and we used an attacker Linux Kali Virtual Machine. After having performed many times the offensive activities on the victims, the traffic extracted from PCAPs was firstly visually analyzed through the *Message Sequence Chart* (MSC) representation, initially developed by the International Telecommunication Union as a requirement specification of protocols.

This representation is useful for extracting patterns since it gives an overview of the message exchange among communicating entities, considering order of messages and time constraints. As an example, Figure 3 shows the network pattern of a SMB authentication failure, extracted from a PCAP containing a SMB brute force attack (afterwards better explained).



```
172.16.111.130 (Attacker)                    172.16.111.135 (Victime)
TCP 74 58742→445 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=30
TCP 74 445→58742 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK
 TCP 66 58742→445 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=302474 TSecr=431987
              SMB 154 Negotiate Protocol Request
              SMB 197 Negotiate Protocol Response
TCP 66 58742→445 [ACK] Seq=89 Ack=132 Win=30720 Len=0 TSval=302474 TSecr=43198
        SMB 246 Session Setup AndX Request, NTLMSSP_NEGOTIATE
SMB 320 Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE
    SMB 455 Session Setup AndX Request, NTLMSSP_AUTH, User: \MYerhBoM
    SMB 105 Session Setup AndX Response, Error: STATUS_LOGON_FAILURE
TCP 66 58742→445 [FIN, ACK] Seq=658 Ack=425 Win=31744 Len=0 TSval=302481 TSecr=
TCP 66 445→58742 [ACK] Seq=425 Ack=659 Win=65792 Len=0 TSval=431990 TSecr=3024
        TCP 54 445→58742 [RST, ACK] Seq=425 Ack=659 Win=0 Len=0
```

Figure 3. MSC of SMB Authentication Failure

In order to interpret this chart, it is necessary to understand how the SMB protocol works. The SMB packet header contains the command to be executed (expressed as an hex code), which is appended to a command-related payload. For example, with reference to the *"Session Setup AndX Request"* command shown in Figure 3, at a fixed position in the header it is possible to extract the *SMB_Command* ("0x73" for SMB and "0x0001" for SMB2) or the *ResponseFlag* ("0" for request packet and "1" for the response one), while in the request packet payload the credentials are included too.

Once modeled a PSEXEC attack, we analyzed a broader set of lateral movement threats used by an attacker to spread infections inside the victim LANs. In particular, we formally modeled the threats listed in Tab.1 (used by real APTs) as XFSM signatures and validated their effectiveness by simulation.

Table 1. Lateral movements

Branch	Lateral movement
0 → 0.1 → 0	SMB Brute force
0 → 2.a4	SMB session creation, malicious file upload, service creation and start
0 → 2.a5	PSEXEC
0 → 2.b4	SMB session creation, malicious file upload, victim local time acquisition, scheduled task creation
0 → 1.b2	Scheduled task enumeration
0 → 1.c2	Service enumeration
0 → 1.d	Directory listing
0 → 1.e	File copy
0 → 1.f	SMB session delete

The single signatures were then integrated in a single overall XFSM signature, which can be considered as a complex cyber detection decision tree, shown in Figure 4.

Every branch models a particular offensive technique, whose XFSM signature can share state and transitions with others. In particular, with reference to Figure 4, the transition 0→1 refers to the remote authentication event inside a LAN, consisting in the creation of a SMB session.

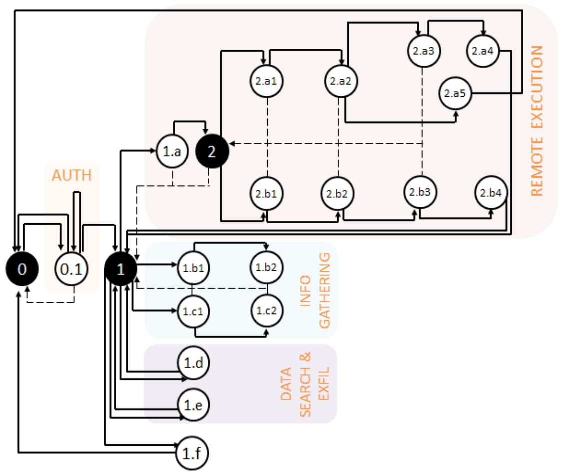


Figure 4. Complex behavioral signatures for lateral movements

For such operation, the attackers must authenticate (tr. 0→0.1); in case of valid credentials, the session is created and the attacker can access the victim resources (tr. 0.1→1), otherwise an authentication error is sent to the attacker (tr. 0.1→0). In such conditions, a SMB brute force can hence be modeled as a chain of such events. Once the attacker is authenticated, he can remotely perform many activities on the victim, such as executing commands, gathering information about the processes, listing and exfiltrating data, etc. We analyzed a subset of the lateral movements widely used by real APTs, only in the Windows XP case (SMB version 1); the same approach is applicable to the successive SMB versions, specifying the relative SMB command name and hex values.

Considering the possibility of remotely controlling the victim, as previously anticipated, we firstly considered the PSEXEC technique (from tr.0 to tr. 2.a5), which automatically allows to create the SMB session, to upload a malicious software in the victim and to launch it as a new service. In order to detect this attack it is necessary to interpret the SMB commands, through DPI (e.g. in case the SMB pipe "*svcctl*" is invoked, it means that the attacker is trying to interact with the Windows process "*services.exe*", responsible of the services management inside the operating system, and then the protocol SVCCTL is going to be used).

A similar attack can be manually performed, using Windows administration commands: once authenticated using SMB functionalities ("*net use*" command), the attacker uploads a malicious software on the victim host (e.g. at *"C:\..\mw.exe"* path), creates a new Windows service connected to it ("*sc \\IP create maliciousService binpath= C:\..\mw.exe"*) and then launches it ("*sc \\IP start maliciousService"*). These manual operations can be detected by means of a stateful signature (from tr.0 to tr. 2.a4), similar to the PSEXEC one. In particular, at 2.a2 state it is possible to identify if the attack is performed with the PSEXEC automatic tool or manually in fact, in the first case, since it is an automated attack, the creation and the execution of the service is performed invoking the SMB pipe "*svcctl*" just once, while in the second case it is invoked every time the "*sc*" command is used. Therefore, the transition 2.a2→2.a3 can be modeled as follows:

Transition from* <u>State#2.a2</u> *to* <u>State#2.a3</u>*:
(Ev.A ***OR*** *Ev.B)* ***AND*** *Cond.C* ***AND*** *Cond.D*
where:
- <u>Event A</u>: SMB Com NT Create AndX (SMB Command="0x2a")
- <u>Event B</u>: SMB Com Open AndX (SMB Command="0x2d")
- <u>Cond.C</u>: Request packet (SMB ResponseFlag="0")
- <u>Cond.D</u>: Pointed resource related to SVCTTL session ("SVCCTL" string in SMB Path)

while 2.a3→2.a4 is the same as 2.a2→2.a5.

Another way to remotely execute a tool in Windows platform is to create a <u>scheduled task</u> by means of the "*at*" command. In this case, once the attacker has created a SMB session and uploaded the malicious software (e.g. at *"C:\..\mw.exe"* path), he acquires the local victim time (*"net \\IP time"*) and then schedule to execute the malware a few seconds after ("*at \\IP C:\..\mw.exe*"). Similarly to the previous one (*"sc"*), the *"at"* command can be invoked by a dedicated SMB pipe (namely "*atsvc*") and the high layer protocol ATSVC allows to transfer the needed information to schedule a task. Such operations (referred to the branch from 2 to 2.b4), can be modeled with the following transition states:

Transition from* <u>State#2</u> *to* <u>State#2.b1</u>*:
(Ev.A ***OR*** *Ev.B)* ***AND*** *Cond C* ***AND*** *Cond D*
where:
- <u>Event A</u>: SMB Com NT Create AndX (SMB Command="0x2a")
- <u>Event B</u>: SMB Com Open AndX (SMB Command="0x2d")
- <u>Cond. C</u>: Request packet (SMB ResponseFlag="0")
- <u>Cond.D</u>: Pointed resource related to SRVSVC session ("SRVSVC" string in SMB Path)

Once the malware file has been transferred, the SMB client tries to open a SRVSVC session, through the same command as *Transition State 2-3* using \srvsvc pipe, used to manage the *lanmanserver* service (conceived to share file and print resources with clients over the network). This service is invoked in case of "*net time*" command, whose parameter are sent in the next state transition.

Transition from* <u>State#2.b1</u> *to* <u>State#2.b2</u>*:
Ev.A
where:
- <u>Event A</u>: SRVSRC NetrRemoteTOD (SRVSVC Command="0x1c")

Using such command, the attacker looks for the local time reference of the victim machine.

Transition from* <u>State#2.b2</u> *to* <u>State#2.b3</u>*:
(EvA ***OR*** *Ev.B)* ***AND*** *Cond.C* ***AND*** *Cond.D*
where:
- <u>Event A</u>: SMB Com NT Create AndX (SMB Command="0x2a")
- <u>Event B</u>: SMB Com Open AndX (SMB Command="0x2d")
- <u>Cond.C</u>: Request packet (SMB ResponseFlag="0")
- <u>Cond.D</u>: Pointed resource related to ATSVC session ("ATSVC" string in SMB Path)

Once the attacker knows the victim time reference, he schedules a task in order to execute the malicious command at a particular moment. In order to do that, it creates a ATSVC, which invokes Microsoft AT-Scheduler Service

Transition from* <u>State#2.b3</u> *to* <u>State#2.b4</u>*:
Ev.A
where:
- <u>Event A</u>: ATSVC JobInfo command refers to MWfilename previously transferred

In order to schedule a task, using the "*at*" command previously reported, the attacker must specify the malware path. In case it refers to the path of the file previously transferred (tr 1→ 1.a), it means he is trying to execute the malicious software.

The other branches shown in Figure 4 refer to other internal operations. In particular the 1.b describes the <u>enumeration of scheduled task</u>, using "*at \\IP*" command, which invokes the ATSVC service as before (tr.1→1.b1 has the same condition as 2.b2→2.b3). Instead, the tr. 1.b1→1.b2 refers to *"at"* job enumeration (ATSVC command *0x2*), as described hereafter:

Transition from* <u>State#1.b1</u> *to* <u>State#1.b2</u>*:
Ev.A
where:
- <u>Event A</u>: ATSVC JobEnum command (ATSVC Command="0x2")

The 1.c branch describes the operation needed to <u>enumerate the Windows services</u>, using "*sc \\IP query state= all*" command, which invokes the SVCCTL service as before (tr.1→1.c1 has the same condition as 2.a2→2.a3). Instead, the tr. 1.c1→1.c2 refers to *"sc"* service enumeration (SVCCTL command *0xe*):

Transition from* <u>State#1.c1</u> *to* <u>State#1.c2</u>*:
Ev.A
where:
- <u>EventA</u>: SVCCTL EnumServiceStatusW (SVCCTL Command="0xE")

Concerning the internal operations needed to exfiltrate data using basic Windows functionalities, we model the command needed to <u>list directories of interest</u> inside the victim machine (using the *"dir \\IP\C$\DIRECTORY"*). Such command can be detected looking for SMB packets with *transaction* request (*SMB Trans2 request* - command *"0x32"*), invoking *FirstFirst* function which searches a directory for a file or subdirectory with a name that matches a specific name:

Transition from* <u>State#1</u> *to* <u>State#1.d</u>*:
Ev.A ***AND*** *Cond B* ***AND*** *Cond C*
where:
- <u>EventA</u>: SMB Trans2 request (SMB Command="0x32")
- <u>Cond. B</u>: Request packet (SMB ResponseFlag="0")
- <u>Cond. C</u>: FindFirst flag = "0x00000006"
- <u>Action</u>: Pass to State 1

Once the attacker wants to copy a particular file, he can use the SMB functionalities to remote copy it with *"copy \\IP\C$\PATH."*. At traffic level, thus such operation can be detected looking for SMB *ReadAndXRequest* and hence the state transition can be modeled as follow:

Transition from State#1 to State#1.e:
Ev.A
where:
- EventA: SVCCTL ReadAndXReques (SVCCTL Command=" 0x2e")
- Action: Pass to State 1

Finally, once the attacker has concluded his malicious operations, ha can decide to terminate the SMB session, using the *LogOff AndX Response* command. The state transition can hence be modeled as follows:

Transition from State#1 to State#1.f:
Ev.A ***AND*** *Cond B*
where:
- Event A: SMB LogOff AndX Response (SMB Command="0x74")
- Cond. B: Request packet (SMB ResponseFlag="0")
- Action: Pass to State 0

The evolution of the states is tracked inside a dedicated *Look-Up Table*, which is flushed after a timeout value (configurable for each state). In Figure 4 such timeout transitions are reported as dashed lines.

4.1 Collaborative centralized detection

Many attackers exploit the IP spoofing technique, which consists in sending packets with a false source IP, in order to hide the identity of the sender (e.g. *anonymize* the attacker) or impersonate an host internal to the network in order to evade security control (i.e. a firewall or an IDS).

In order to demonstrate our approach, we identified two attacks, represented in Figure 6, which are based on the following techniques: decoy scan and spoofed Denial Of Service (DOS). In the first case, the attacker aims at scanning a victim host without revealing the attacking source; in order to be anonymized, the attacker spoofs IP decoys (called *zombies*), which are internal hosts reachable by the victim. The attacker is then able to scan the victim hosts without revealing its real identity (i.e. IP address) since the scan appears to be originated from different hosts supposed to be in the trusted victim's network. In the second case, the attacker aims at flooding a victim host by high rate connections, to make it unable to process other legitimate requests. By spoofing the victim's IP address, the attacker can conceal identity and hence the victim cannot identify the real attack sources in order to block the attack.

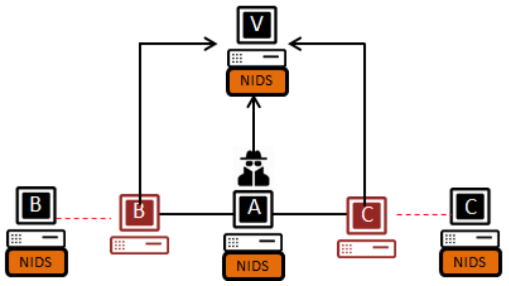


Figure 5. Decoy scan and spoofed DOS scenario

To successfully detect such attacks, it is required to differentiate the modeling of the signatures for each one of the detection levels. At host level, NIDS agents are able to perform a preliminary detection that reveals the specific protocol/technique exploited by the attacker and alert the upper layers. As an example, it is possible to detect a suspicious spoofed DOS or a host scanning by means of the finite-state machine approach proposed in [9], but it is not possible to reveal the real attack source at this stage.

In order to finalize the detection, the first detection level (i.e. D-StreaMon agents) has to send alerts for the mitigation actions. The NIDS Manager collects and correlates such events to perform the final recognition of the ongoing attack and the correct attribution of the attack's source. For this objective, the NIDS Manager observes the source of the incoming warnings: the NIDS of the spoofed hosts (i.e. the *zombies*) will not report anything, while the NIDS of the attacker's anchor host (i.e. the real source of the attack) will report a warning. More specifically, representing a warning through the notation $W_i^{y\rightarrow z}$, where i is the NIDS agent reference and y→z indicates the attack direction, the IP spoofing condition can be modeled as:

$$W_v^{B\rightarrow V} AND NOT W_b^{B\rightarrow V}$$

This detection algorithm can be implemented by means of a simple *Look-Up Table*, able to track all the events aggregated by the NIDS Manager. Once a spoofed IP event is detected, the attribution is performed aggregating, for each attack type, the warning referring to the same victim. The offensive activity is attributed to the host for which the event is recognized both by the own ($W_a^{A\rightarrow V}$) and the victim NIDS.

5. Signature validation

In order to validate the proposed detection technique, we implemented a virtual lab with a Kali attacker box, two WinXP victim hosts and two Win7 victim hosts, as shown in Figure 6. In particular, using this virtual lab, we specifically test both the two scenarios described in Section 4.

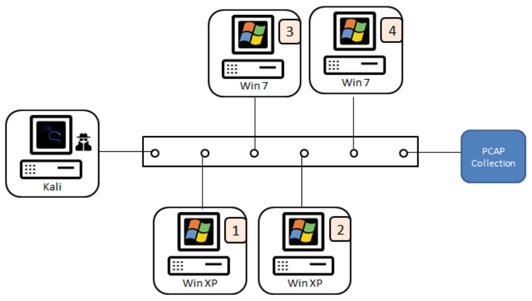


Figure 6. Validation architecture

Firstly, concerning the validation of the signatures described in Figure 4, we registered a 3-hour traffic track, emulating routine operations typical of an office host, as web browsing, shared directories accesses, operating system and antivirus updates. Additionally, we performed different lateral movements (opportunely scheduled) between all the hosts, using all the techniques described in Section 4.1 (for PSEXEC we used the tools described in [18]). Then, in order to assess how to face the truncated chain events and false alarms with an effective timeout management, in some cases we voluntarily used incorrect credentials. From Kali box to host (2), we also performed a SMB brute force (trying to access to shared resources). For this purpose, we used the implementation of the well-known hacking platform Metasploit, by means of the module "*auxiliary/scanner/smb/smb_login*" [18].

We then validated the technique proposed in Section 4.2 to detect and attribute IP-spoofing based attack inside the LAN, represented in Figure 5. In particular, we simulated a decoy scan from the Kali host to a Windows one (considering the remaining hosts as zombies), using the well-known network scanner *nmap* (with argument *"–D"*); the same scenario was reproduced for the spoofed DOS attack, using the opensource penetration tool *hping3*.

5.1 Test results

The offensive activities, that were expected to be detected, were correctly identified by the proposed solution, without any false alarm.

Table 2. Performed lateral movement for validation

HOSTS		PSEXEC TOOL	COUNT
Attacker	Victim		
K	1,2,3,4	Impacket	3
1	2,3,4	PsExec SysInt.	2
2	1,3,4	PsExec SysInt.	2
3	1,2,4	PsExec SysInt.	2
4	1,2,3	PsExec SysInt.	2

In order to understand the advantages related to the use of such approach with respect to the stateless NIDS one, we performed many PSEXEC attacks among the hosts (as shown in Table 2) and we logged all the warnings received using the state transitions individually considered as stateless signatures (apart from transitions 1.a→2 and 2.a1→2.a2, which requires information from the LUT).

Table 3. Avoided false alarms

	Tran. 0-0.1	Tran. 0.1-1	Tran. 1-1.a	Tran. 2-2.a1	Tran. 2.a2-2.a3	Tran. 2.a3-2.a4
Pkts	4615	70	54	222	47	68
Pkts/Tot [%]	6.59	0.10	0.08	0.32	0.07	0.10

Analyzing the results shown in Table 3 (related to a track of ~70k packets), it is possible to verify that:

- if we had analyzed this traffic with a stateless NIDS with those signatures, we would have received many false alarms, mainly associated to administration licit connections;
- the peak present for T0-1 column mainly relates to the SMB brute force attack intentionally performed from Kali (K) machine to WinXP (2) host, which was anyway correctly detected by the signature related to the branch 0→0.1 of Figure 4.

6. Conclusions and Future Works

This work shows how to effectively model malicious lateral movements for detection purposes, by means of finite-state machine signatures to be deployed in IDS agents (host level). The validity of such an approach is strengthened with additional more complex use cases. The possibility to capillary monitor the traffic at host level with the deployable version of StreaMon [9] enables additional detection scenarios. For instance, the events/warnings generated at host level can be aggregated in a centralized NIDS Manager, which can correlate them so as to detect and attribute advanced attacks, such as ones based on IP spoofing which aim at evading the security controls. The manager, correlating the stored information, can then trigger proper mitigation actions (blocking, honeynets and so on), that can be implemented with custom solutions or integrated in a SDN architecture.

7. References

[1] Tankard, Colin. "Advanced persistent threats and how to monitor and deter them." Network security 2011.8 (2011): 16-19.

[2] "APT Detection – Closing the Gaping Hole", 2014

[3] Dan Reis, "It's Only the Beginning for Endpoint Security", FirEye 05

[4] A. Oberle et al., "Preventing Pass-the-Hash and Similar Impersonation Attacks in Enterprise Infrastructures," 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), Crans-Montana, 2016, pp. 800-807.

[5] "Pass-the-hash attacks: Tools and Mitigation", SANS, 2010

[6] Jadeja et al. "Implementation and Mitigation of Various Tools for Pass the Hash Attack." Procedia Computer Science 79 (2016): 755-764.

[7] Roesch, Martin. "Snort: Lightweight Intrusion Detection for Networks." LISA. Vol. 99. No. 1. 1999.

[8] Marchetti, Mirco, et al. "Analysis of high volumes of network traffic for Advanced Persistent Threat detection." Computer Networks (2016).

[9] G. Bianchi et al. , "StreaMon: A software-defined monitoring platform" , Teletraffic Congress (ITC), 2014 26th International , pp.1 -6

[10] Raghav, Iti, Shashi Chhikara, and Nitasha Hasteer. "Intrusion Detection and Prevention in Cloud Environment: A Systematic Review." International Journal of Computer Applications 68.24 (2013).

[11] "Mitigating Pass-the-Hash and Other Credential Theft", Microsoft, 2014

[12] G. Rush et al., "DCAFE: A Distributed Cyber Security Automation Framework for Experiments," Computer Software and Applications Conference Workshops, 2014 IEEE 38th International, Vasteras, 2014.

[13] "Detecting “Pass-the-hash” attacks with Sagan in real time." https://quadrantsec.com/about/blog/detecting_pass_the_hash_attacks_with_sagan_in_real_time/

[14] M. Ussath, D. Jaeger, Feng Cheng and C. Meinel, "Advanced persistent threats: Behind the scenes," 2016 Annual Conference on Information Science and Systems (CISS), Princeton, NJ, 2016, pp. 181-186.

[15] Ventre, Pier Luigi, et al. "D-STREAMON-a NFV-capable distributed framework for network monitoring." arXiv preprint arXiv:1608.01377 (2016).

[16] Mitchell, David. "Intrusion Detection from Simple to Cloud" (2015).

[17] Sekar, R., et al. "Specification-based anomaly detection: a new approach for detecting network intrusions." Proceedings of the 9th ACM conference on Computer and communications security. ACM, 2002.

[18] A.Greco, G.Bianchi – ”Detection of offensive lateral movements using finite-state-machine-based patterns” – Global Wireless Summit, Aarhus, 2016

[19] Hintjens, Pieter. ZeroMQ: Messaging for Many Applications. " O'Reilly Media, Inc.", 2013.

[20] A.Greco, A.Caponi, G.Bianchi – “Facing lateral movements using widespread behavioral probes” - 11th International Conference for Internet Technology and Secured Transactions, Barcelona, 2016

8. Acknowledgements

This research was partially supported by the EU Commission within the Horizon 2020 program, SCISSOR project grant no 644425.