Multi-step Attack Modelling and Simulation (MsAMS) Framework based on Mobile Ambients

Virginia N. L. Franqueira
University of Twente

Raul H. C. Lopes
Brunel University

Pascal van Eck
University of Twente

franqueirav@ewi.utwente.nl raul.lopes@brunel.ac.uk p.a.t.vaneck@utwente.nl

ABSTRACT

Attackers take advantage of any security breach to penetrate an organisation perimeter and exploit hosts as stepping stones to reach valuable assets, deeper in the network. The exploitation of hosts is possible not only when vulnerabilities in commercial off-the-shelf (COTS) software components are present, but also, for example, when an attacker acquires a credential on one host which allows exploiting further hosts on the network. Finding attacks involving the latter case requires the ability to represent dynamic models. In fact, more dynamic aspects are present in the network domain such as attackers accumulate resources (i.e. credentials) along an attack, and users and assets may move from one environment to another, although always constrained by the ruling of the network. In this paper we address these dynamic issues by presenting MsAMS (Multi-step Attack Modelling and Simulation), an implemented framework, based on Mobile Ambients, to discover attacks in networks. The idea of ambients fits naturally into this domain and has the advantage of providing flexibility for modelling. Additionally, the concept of mobility allows the simulation of attackers exploiting opportunities derived either from the exploitation of vulnerable and non-vulnerable hosts, through the acquisition of credentials. It also allows expressing security policies embedded in the rules of the ambients.

Categories and Subject Descriptors

K.6.5 [Management of Computing and Information Systems]: Security and Protection—*Unauthorized access*

General Terms

Security, Management

Keywords

Network Attack, Vulnerability Assessment, Attack Graph, Hypergraph

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'09 March 8-12, 2009, Honolulu, Hawaii, U.S.A.

Copyright 2009 ACM 978-1-60558-166-8/09/03 ...\$5.00.

1. INTRODUCTION

One single hole in the network perimeter is enough to allow an attacker to penetrate the network and exploit hosts as stepping stones to reach valuable assets. Defenders need to tune into the mindset of attackers [29] to track those possible stepping stones and manage the trade-off between impact of attacks and cost of countermeasures. This task is rather challenging due to the complexity and size of current networks, and due to the variety of opportunities and strategies used by attackers. For example, attackers do not only take advantage of vulnerabilities in commercial off-the-shelf (COTS) software components but can also take advantage of credentials (i.e. any information used for access control) acquired dynamically while the attack is taking place. Such credentials greatly increase an attacker spectrum of possibilities since he gains the ability to exploit safe, i.e. not vulnerable, hosts as well. Therefore, attackers are dynamic, and rational, entities which “move” towards the resources of a victim; they accumulate resources along an attack, such as the knowledge of credentials. Furthermore, users and resources of a network might also “move”; in any case they are bound to the security policies enforced by the network. For example, an employee might switch from working with his laptop in the office or at home, but anyway he is subject to the ruling of the network which allows the access to some resources just from within the office environment. To the best of our knowledge, models which represent a snapshot of a network, such as Attack Graphs [14, 1, 15, 23, 32, 20, 25, 31, 34], are unable to deal with all these dynamic aspects.

We address these dynamic issues by proposing MsAMS, a framework for modelling and simulation of network attacks, the design of which draws heavily on Cardelli’s work on *Mobile Ambients* [5, 6] and formal biology [4], and on Milner’s work on bigraphs [21]. In this framework, a network is viewed as a so-called *Ambient* containing other ambients, such as subnets, hosts, and firewalls, on a tree hierarchical structure. As further explained in the paper, an *Ambient* defines a hyperedge [16] which represents the idea that a communication performed over it is seen by each ambient in that hyperedge, thus no link between sibling hosts (belonging to a same ambient) is required. Besides, an ambient boundary may contain rules which allow its interaction with other ambients, resulting in changes on the rules of the ambients involved. After the modelling is complete, MsAMS simulates an attacker (also an *Ambient*) dynamically finding an attack path allowed by the modelled ambients and their embedded rules.

We see this framework very much as a *working tool* whose

users are security practitioners, such as network administrators. It allows them to gain knowledge about their network. It also permits zooming in some parts of the network they want to investigate, and zooming out to a more abstract level of network as a whole. MsAMS is flexible enough to allow the modelling of a network in different ways and with more or less details, at the discretion of the person using it. MsAMS does not require complex sets of pre- and postconditions to model the composition of vulnerabilities in consecutive attack steps, as it happens with other approaches [15, 8, 33]. In fact, it uses the access-to-effect paradigm used by other researchers (e.g. [14, 20]) which can be obtained from the NVD [24]. Modelling the input requires (i) the network configuration, (ii) vulnerabilities in COTS present in the network which can be obtained automatically from vulnerability scanning tools such as Nessus [22], and (iii) their attributes, which can be obtained from vulnerability databases such as the NVD [24].

The paper is organised as follows. A running example is introduced in Section 2, followed by review of related work in Section 3. An overview of the MsAMS framework is described in Sections 4 and 5, using the running example previously presented. In Section 6, we introduce the concept of exposure and describe acquisition of credentials, using an insider attack example. An overview of the heuristic algorithms implemented to find attacks is provided in Section 7, and results are reported in Section 8. Finally, in Section 9, we conclude and list future work.

2. RUNNING EXAMPLE

We use the network illustrated in Figure 1 from Ingols et al. [14] as the basis for introducing core concepts and to give an overview of the MsAMS framework.

In this example network the attacker is initially located on host A and wants to reach either host E or F. The firewall only allows traffic from host C or D to host E. Additionally, all hosts have a single open port with a vulnerable service running. Each vulnerability is remotely exploitable and allows the attacker to gain privileged access to the host.

The example network can be represented in terms of *Ambient* as illustrated in Figure 2(a). In the figure, we see an ambient *net* containing five ambients *A, B, C, D, FW*, which represent hosts A to D and firewall *FW*. The firewall is viewed as a *membrane* protecting other ambients, i.e. hosts, *E* and *F*. Figure 2(b) provides a zoom view of ambient *A*. This ambient contains an ambient representing a listening service *sv_A*, containing itself an ambient representing a vulnerability *v_A* on that service. The exploitation of *v_A* results in the acquisition of privilege admin (*OS_admin_A*) over host A Operating System.

Notice that we presented above *one* intuitive way of modelling the example network. However, this is not at all the only option since MsAMS is flexible. For example, the firewall could be modelled as well as an ambient which interfaces with two ambients representing subnets (*A, B, C, D*) and (*E, F*).

3. RELATED WORK

The MsAMS framework has basically the same objective as an Attack Graph, both show “the ways an attacker can compromise a network or host” [14]. Therefore, we consider Attack Graphs as our main source of related work, although

we take the novel approach that finding attacks is a multi-objective optimisation problem.

Ou et al. [26] use Datalog rules in the MulVAL tool for modelling the input required for the generation of Attack Graphs. A Prolog reasoning engine then captures the relationship among those with further rules and generates the graph. Their framework, like ours, has a formal approach for input specification and for reasoning. Recent work from Sawilla and Ou [28] also uses the same running example as we do, illustrated on Figure 1, and the graph produced by MulVAL is shown in [28, Page 3]. Thus, we compare their approach and ours more closely. Their graph has 50 nodes while ours has 27 nodes¹. This significant difference may be explained by the presence of redundant nodes in the former graph, such as the ones related to firewall rules ($C > E$ and $D > E$). Firewall rules, modelled as “*haci*” clauses in MulVAL, appear twelve times in the graph. In our approach the *Ambient* firewall appears once in the graph but contains two filtering rules at its boundary, as it happens in reality.

NetSPA tool [14, 34] uses a so-called Multiple-Prerequisite (MP) graph to represent (i) state, i.e. the attacker level of access on a host, (ii) prerequisite, i.e. reachability or a credential needed for exploiting a vulnerability, and (iii) vulnerability instance. Prerequisites allow the exploitation of vulnerabilities, and when the attacker reaches a vulnerability, a change in the attacker state occurs. “Outbound edges from prerequisite nodes point to the vulnerability instances that require the prerequisite for successful exploitation” [34]. Therefore, they only represent hosts which contain vulnerabilities and their concept of credential differs from ours. We consider as a credential any information used for access control (e.g. passwords, public session keys) required to enter/log into a host, vulnerable or not. This way we can represent normal and abnormal behaviours of a network.

Jajodia, Noel et al. [15] rely on detailed pre- and postcondition rules to compose attack steps. The specification of these conditions enables TVA tool to find attacks composed both by vulnerable and non-vulnerable hosts (as illustrated in example on [15, Page 258], where host *ned* is not vulnerable). However, this is achieved as a consequence of the postcondition enabled by the exploitation of the preceding vulnerability, requiring a detailed analysis of dependencies among vulnerabilities. We have a different perspective and use a more simplistic approach, based on access-to-effect, to model vulnerabilities, but nevertheless consider the possibility of an attacker acquiring credentials which enable the compromise of non-vulnerable hosts.

Sheyner and Wing [31] provide a toolkit based on symbolic model checker. Like other model checker-based approaches [27], state explosion [3] is an issue. These graphs represent a path for every single possible combination of attack steps, thus its complexity becomes exponential. Our approach does not suffer from this problem since the graph only represents nodes (i.e. *Ambients*) and links which occur in practice, and the composition of attack steps relies on the matching of rules embedded on the boundaries of ambients. Sheyner also models trust relations among hosts. It means that the credential for a host h_1 gives the attacker access to host h_2 as well, if these two hosts trust each other. Therefore, the real-life situation that my password gives someone

¹Eight nodes illustrated in Figure 2(a) and additional three nodes for each host, as illustrated for host *A* in Figure 2(b), plus one attacking node

Figure 1: An example network

Figure 2: Modelling the example network as *Ambients*

access to my account, not to another account, in my workstation can be represented with MsAMS but cannot with their approach.

Ha, Chinchani et al. [12, 7] propose a type of graph which allows not only the modelling but also the simulation of an attacker searching through the graph. Nodes are associated with tokens and edges associated with minimum and maximum costs. Similar to MsAMS, this method can represent/model many types of nodes (e.g. firewalls, vulnerabilities, services, and accounts). During the simulation process, the attacker acquires tokens, and if he has a token he incurs on minimum cost to traverse an edge, otherwise on maximum cost. This approach has many similarities with our approach, e.g., it allows modelling insiders which “hold” tokens (i.e. credentials) at the beginning of an attack, but it does not allow the representation of Access Control Lists, and the acquisition of credentials along an attack path.

Gorodetski and Kutenko [11] have a grammar-based approach for the simulation of attacks in networks. A family of grammars for each attacker intention has to be specified and, when instantiated, generates attack paths. Among other characteristics, their approach differs from ours because they do not have the objective of *discovering* attacks given a network model.

4. MODELLING WITH THE MSAMS FRAMEWORK

We have seen in Section 2 the network topology of the example network represented in terms of ambients. In essence *Ambients* are environments where any computing activity can happen. They are abstractions introduced to represent hosts, services, vulnerabilities, networks, users and even credentials. Each ambient can possibly interact with other ambients in its neighbourhood, depending on its and their capabilities.

DEFINITION 1. An Ambient *Amb* is a tuple (*AmbList*, *Rules*), where *AmbList* is a list of ambients contained in

Amb, and *Rules* is a list of static rules defining the dynamic behaviour of ambient *Amb*.

We use the notation $Amb : [AmbList][Rules]$. The ambients of the network example, illustrated in Figure 2, are specified in MsAMS as follows.

```

1 net: ["A" "B" "C" "D" "FW"] []
2 FW: ["E" "F"] [Repeat (AllowIn "C" "sv_E"),
                  Repeat (AllowIn "D" "sv_E")]
3 A: ["sv_A" "OS_admin_A"]
   [Repeat (AllowIn "net" "sv_A")]
4 sv_A: ["v_A"] [Repeat (Accept "net")]
5 v_A: [] [Repeat (Accept "sv_A")]
6 OS_admin_A: [] [Repeat (Accept "v_A")]
Similar rules as 3-6 apply to ambients B-F

```

In this example we use three primitives, as described next.

1. The primitive *Repeat* allows the execution of a capability which follows, on an infinite loop.
2. The primitive *Accept* allows one ambient to accept that another ambient moves into it. The intention of entering an ambient is expressed in terms of the primitive *Enter*, used for example for simulating the actions of an attacker.
3. The *AllowIn* primitive allows one ambient to define which ambient is allowed to cross its boundaries towards another ambient. Although this rule can be modelled using *Enter* and *Accept*, it has been explicitly introduced in MsAMS for convenience when modelling firewalls and ports which allow access to listening services in hosts.

Details about the specification of ambients provided above by rules 1-6 follows.

- Line 1 specifies the ambient *net*. It contains five other ambients and no action rules.

- Line 2 specifies the ambient *FW* which contains two other ambients. It restricts the broadcasting of messages from hosts A-D towards its children ambients, *allowing* only ambients *C* and *D* to communicate with ambient *sv_E*. *AllowIn AmbientSource AmbientDestination* may restrict communication between any pair of ambients. For example, if we had defined that ambients *A* to *D* were inside ambient “dmz”, we could have firewall rules of the type: *AllowIn* “dmz” “F” or *AllowIn* “ssh/tcp/22” “sv_E”.
- Line 3 specifies ambient *A* containing two other ambients, a listening service *sv_A* and a privileged account *OS_admin_A*. It contains a rule representing a port which allows the communication from the network *net* to service *sv_A*.
- Line 4 specifies ambient *sv_A*, containing a vulnerability represented by the ambient *v_A*. This service is continuously *accepting* ambient *net*. It means that anyone allowed in the network *net* can enter, i.e. request, *sv_A*.
- Line 5 specifies an empty ambient *v_A*. Since this vulnerability is remotely exploitable, it does not require an attacker to be authenticated as a user of the host to enable its exploitation. Thus, it accepts ambient *sv_A* directly. Besides, this vulnerability results in the acquisition of admin ² privilege.
- Line 6 specifies an empty ambient *OS_admin_A*. This ambient is continuously *accepting* ambient *v_A*.

It is important to keep in mind that this example is oversimplified. For example, it only represents one type of privilege over the Operating System and no access control.

4.1 Matching Enter/Accept

In Cardelli’s Bioware Languages [4], an *Ambient* (called *membrane* in that paper) can possibly contain other *Ambients* inside it. In that sense, an *Ambient* has a set of children and a parent. Cardelli defines two conditions that must **both** be satisfied for an *Ambient* *X* to move into an *Ambient* *Y*:

1. *X* is a sibling of *Y* or *X* and *Y* are children of sibling parents.
2. *X* has an action request entry into *Y*, which we denote *Enter* “*Y*”, and *Y* allows the entry movement with an action *Accept* “*X*”.

We replace in MsAMS framework the first condition with:
X is a sibling of Y or X and Y are children of sibling parents, or an edge exists connecting X and Y.

5. SIMULATIONS WITH MSAMS FRAMEWORK

The architecture of the MsAMS framework and where the simulation engine fits is illustrated in Figure 3.

The simulation engine receives as input (i) network as ambients, i.e. a tree hierarchical structure defining the network

²Admin access represents generically privileged access to the operating system, for example root access to Unix-based, and administrator access to Windows-based systems.

topology (similar to Figure 2) and a hypergraph defining broadcast communication, (ii) a set of static rules determining *Capabilities* and *Actions*, as shown in Section 4. These rules can be used by the ambient and can be performed at the boundary that the ambient defines. The set of actions implicitly defines a non-deterministic choice of action to perform, and (iii) a set of computing *Agents*, i.e. one or more attacker ambients. The engine then performs two basic tasks using heuristics, as described in Section 7. First, it assigns automatically value and cost to ambients of the hypergraph. Second, it computes possible steps an attacker can perform on the ranked hypergraph. Each of these steps can either be accepted, if the actions of the attacker (ambient) and the actions of other ambients match, as described in Section 4.1, or rejected if the actions do not match. This match is achieved via reduction rules [5]. A match means that the attack can actually perform the step, and this is recorded by the engine. In the end of the simulation, we have the attacker complete trace until a target, i.e. a host of high value, is reached. This trace is a possible multi-step attack on the modelled network. An attacker trace for the running example (see Figure 2) is illustrated next.

```
Enter "sv_D"
Enter "sv_E"
Enter "v_E"
Enter "OS_admin_E"
```

This trace shows the possible attack *ADE*. Another possibility which can be obtained by executing further the simulation is *ACE*. Note that vulnerabilities *v_A* and *v_D* were not exploited because the attacker had more incentive to exploit vulnerability *v_E* which leads to *OS_admin_E*, a higher value ambient.

6. ACQUISITION OF CREDENTIALS

In this section we use an insider attack example, more appropriate for illustrating the acquisition of credentials in MsAMS. However, first we introduce the concept of exposure.

6.1 The role of exposures

We use exposures to represent stealthy ways to acquire credentials. An attacker can get remote or local access to a host by means of vulnerabilities but, most of the times, he does not automatically obtain credentials (e.g passwords or private session keys) for that host. Thus, an exposure is an abstraction to model the availability of credentials. It corresponds to the real situation of passwords saved locally, and passwords acquired using key logging mechanisms or social engineering. However, exposures can also be used for modelling Public Key Infrastructure (PKI), as shown in Section 6.2. To illustrate this concept let’s consider the scenario shown in Figure 4.

There are two ways to penetrate host *h*, either via login using a credential, i.e. passwords *p.us* (user) and *p.ad* (admin), or via vulnerability *v*. In this case, *v* is a remote-to-user vulnerability since it is accessible from any host on the internet and results in user privilege gained over the host Operating System. However, the host also contains an exposure *exp* which is forever repeating the action of releasing credential *p.ad*. The capabilities which constraint this scenario are listed next.

Figure 3: Architecture of the MsAMS framework

Figure 4: Example of exposure

```

1 int: ["alice.h" "net"] []
2 net: ["u1" "u2" "h"] [Repeat (Accept "int")]
3 h: ["p.ad" "OS.ad" "p.us" "OS.us" "v" "exp"]
   [Repeat (Accept "int")]
4 p.ad: [] [Repeat (Accept "u1")]
5 OS.ad: [] [Repeat (Accept "p.ad")]
6 p.us: [] [Repeat (Accept "u2")]
7 OS.us: [] [Repeat (Accept "p.us"),
   Repeat (Accept "v")]
8 v: [] [Repeat (Accept "int")]
9 exp: [] [Repeat (Accept "OS.us"),
   Repeat (ReleaseCred "p.ad")]
10 alice.h: [] [Repeat (Accept "alice")]
11 u1: [] [Repeat (Enter "p.ad")]
12 u2: [] [Repeat (Enter "p.us")]

```

In this case, the computing agent is an ambient *alice*, and it has the initial action *Enter* “alice.h”. Although the list of actions generated by the engine is non-deterministic, let’s suppose the following actions were generated: (i) *Enter* “v”, and (ii) *AcquireCred* “p.ad”. In this case, a match between *AcquireCred* “p.ad” and *ReleaseCred* “p.ad” can happen. This match represents the acquisition of a credential. It occurs by means of reduction rules [5], when agents are computed. This computation has two consequences: 1- *[Repeat (Accept “alice”)]* is included on the capability list of ambient *p.ad*, and 2- ambient *alice* acquires credential *p.ad*, allowing her to enter host *h* with admin privilege.

After being captured by an ambient, a credential is forever kept in the ambient set of *Credentials*. This list of credentials can be seen as an abstraction of a “bag of credentials” which an attacker can accumulate along a multi-step attack. We

assume that the monotonicity property [2] holds (as many other researchers [14, 1, 15, 20] do) and, therefore, once acquired a credential is never lost, i.e. the attacker does not need to backtrack to re-acquire a credential.

A possible attack trace is presented next.

```

Enter "alice.h"
Enter "v"
AcquireCred "p.ad"
Enter "p.ad"
Enter "OS.ad"

```

6.2 An example of insider attack

Figure 5 shows an example of public key acquisition using a Public Key Infrastructure (*pki*). This example has been adapted from [7]. In this example, we use generically the term *key* instead of credential to denote session keys, and passwords.

The environment is a *Bank* with a set of teller hosts, represented by *teller0* and *teller1*, a *manager* host, and a database, *DB*. There is a key to access the manager host, *managerKey*. The *DB* ambient has an user ambient, *dbUser*, that users can access through *In/Out* sequences. Primitives “In” and “Out” establish communication between two ambients representing, for example, the ability to read and write. Hence, the primitive *Seq* (line 4) indicates sequences of read-write, repeated forever. It has also a file system, *dbFS*, and an administrator account, *DBA*. Outside the bank, a *PKI* ambient gives a ticket for a *DB* session. Only the manager has access to the *pkiKey*.

The general rule for each ambient is that an ambient is a hyperedge connecting each ambient inside it. Notice that a list of actions denotes a nondeterministic choice.

Ambients in the example are specified as follows.

```

1 Bank: ["teller0" "teller1" "managerVuln" "dbUserKey"
   "dbaKey" "managerKey" "manager" "DB"] []
2 DB: ["dbFS" "dbUser" "DBA"] [Repeat (Accept "Bank")]
3 dbFS: [] [Repeat (Accept "DBA"),
   Repeat (Seq (In "dbUser") (Out "dbUser"))]
4 dbUser: [] [Repeat (Accept "dbUserKey")]
5 dbUserKey: [] [Repeat (Accept "teller0"),
   Repeat (Accept "teller1")]
6 DBA: [] [Repeat (Accept "managerVuln"),
   Repeat (Accept "dbaKey")]
7 dbaKey: [] []
8 managerVuln: [] [Repeat (Accept "Bank")]
9 manager: [Repeat (Enter "managerKey")]
10 managerKey: [] [Repeat (Accept "manager")]
11 PKI: ["pkiKey" "exp.dbaKey"] []
12 pkiKey: [Repeat (Accept "manager")]
13 exp.dbaKey: [Repeat (Accept "pkiKey"),
   Repeat (ReleaseCred "dbaKey")]

```

Figure 5: Modelling the insider attack example as *Ambients*

```

14 teller0: [] [Repeat (Enter "dbUserKey")]
15 teller1: [] [Repeat (Enter "dbUserKey")]

```

The *DBA* can be entered through a vulnerability, e.g. *Enter "managerVuln"*, or by having a *Enter "dbaKey"* capability which can be acquired by:

- using a capability *AcquireCred "exp.dbaKey"*, or
- giving some ambient the capability *Enter "dbaKey"* from the start.

7. HEURISTICS USED TO FIND ATTACKS

In a previous paper [9], we introduced the idea of representing attacks using a language based on CSP [13]. In that paper, we assumed that a network model was a graph with values assigned to nodes and costs assigned to edges. Searching for an attack was treated as an optimisation problem. In the present approach we still assume that values are assigned to ambients and we still have a cost measure for moves from ambient to ambient. However, we have developed algorithms for assigning values and costs. Borrowing from social network analysis [30] and page ranking we assign value to an ambient proportional to the value assigned to other ambients that point to it. An ambient *A* points to an ambient *B* when a move from the former into the latter is possible.

An algorithm based on PageRank [19] computes a score for each ambient in a network model. That score, *authority*, can be considered as the value for the ambients. The computation starts with a square matrix where cell (i, j) contains a zero when there is no link from *i* to *j*, and one otherwise. The search for an attack uses the values assigned to all ambients in the network model, computed using PageRank, and an initial set of suspicious ambients, the ambients where the attacker would be located. The algorithm can also be given a set of ambients that could be considered as hints in the sense that we expect that an attack would use them. This can be useful when a network administrator wants to know if the network is subject to a specific attack, as for example the massive RealPlayer exploit via SQL injection, reported by the press in January 2008 [17].

The search algorithm computes in each step the set of all possible moves, ordering them by a priority scheme, that is

akin to the cost value used in [9]. To compute the value of a move, the PageRank algorithm is extended with the concept of hub, borrowed from HITS (Hypertext Induced Topic Search) [18]. A hub is an ambient that can be source of a move. An ambient *A* has high hub score when it is possible to move from *A* to ambients with high value. In a local network, the file system where every user has her home directory should have a high asset (authority) rank. Assuming that each user gets access to the home directory using NFS (Network File System) service, an ambient modelling the NFS would have both high authority (accessed by all users) and high hub score, because the NFS uses the file system to satisfy each user request. The HITS algorithm assigns both authority and hub score to a set of ambients, given their neighbourhood, and an initial set of values for them. The search algorithm selects a fixed subset of the *hubbier* ambients, and proceeds in depth-first search, giving priority to moves with high scoring authorities.

8. PERFORMANCE AND SCALABILITY OF MSAMS FRAMEWORK

The time for computing an attack is dominated by the computation of assets ranks and hub scores. This is performed by an algorithm based on the PageRank algorithm [19], and the query-independent HITS (Hypertext Induced Topic Search) algorithm [18]. A naïve implementation of either PageRank or HITS can take $O(n^3)$, demanding a $O(n^2)$ matrix multiplication in each cycle. A more efficient implementation, however, takes into account the fact that the adjacency matrix is sparse and that the matrix multiplication performed in each cycle can be executed in $O(n)$. Assuming that *n* is the number of ambients represented, our implementation precomputes the matrix in $O(n^2)$, and then applies ranking algorithm in time that ranges from $O(kn)$ to $O(kn^2)$, depending on the density of the adjacency matrix and on *k*, the number of iterations necessary for convergence of the power method applied to the computation of either PageRank or the modified HITS. It is important to notice that even for a matrix with billions of nodes the PageRank algorithm tends to converge in less than a hundred iteration. In our tests it converged in less than sixty cycles for a test with 8000 nodes. In a previous implementation [10] we used a full matrix multiplication and fixed *k*, obtaining running

Figure 6: Performance of the MsAMS framework

times of $O(n^3)$ when using more than 8000 nodes. Currently, we have an implementation in Haskell using a sparse matrix multiplication and a matrix akin to the Google matrix [19]. The whole process of both ranking (with HITS and PageRank) and searching for an attack executes in less than 30 seconds for a network with more than 8000 nodes.

Figure 6 shows the computing time for experiments with a variation on our running example illustrated in Figure 2, performed on a machine with Intel Core 2 Duo T5250, 1.5 GHz processor, 2 GB RAM. In all experiments we used one percent of the total of nodes to the left of the firewall. Therefore, we used the following variation of nodes to the left and right of the firewall, respectively: (4,512), (8,1024), (16,2048), (32,4096), and (64,8192). That choice generates a dense adjacency submatrix for the part of the model representing the right side of the firewall. All experiments assumed the attacker positioned initially inside host A.

We express the network models input of our framework in a dedicated language that has also been implemented in Haskell. The 8256 nodes network used in the experiments, e.g., is described in this language with just 46 lines. It takes 7.18 seconds to compile those lines into the internal representation used by PageRank and HITS algorithms.

9. CONCLUSION AND FURTHER WORK

We presented in this paper the MsAMS framework based on the Mobile Ambients theory. It allows the modelling of a network in an intuitive way as *Ambients*, and the specification of static rules defining the dynamic behaviour of each ambient. Then, an engine based on heuristics, simulates attackers steps to find attacks which are possible, given the network modelled. The dynamic aspect of MsAMS allows the simulation of an attacker with the ability to compose attack steps either from the exploitation of vulnerable and non-vulnerable hosts, and the ability to accumulate resources along the way. Firewalls are modelled as intuitively as ambients with filtering rules.

Note that the graph generated has as many nodes as the number of ambients modelled, since no artificial entities are

created to model the reality of the network. This facilitates the relation between the model and the real network topology.

One advantage of the framework is its flexibility. It is up to its users, such as network administrators, to decide which level of details is needed. They can decompose the network at their discretion and model either fine grained entities and relationships, including ACLs (Access Control Lists), or abstract to a higher level and only model the minimum which still allows them to find possible attacks on the network modelled. Alternatively, they can adopt the latter option first as a way to prioritise a sub-graph of interest, and then adopt the former to zoom *in* this sub-graph.

Usability is an aspect that needs to be improved and, therefore, is listed for future work. We plan to build an user interface to allow the graphical manipulation of *Ambients*, when modelling, and the visualisation of possible attacks discovered by the framework, when simulating. Besides, we plan on continuing with testing to fine-tune the approach.

Acknowledgements

This research is supported by the research program Sentinels (www.sentinels.nl). Sentinels is being financed by Technology Foundation STW, the Netherlands Organisation for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

10. REFERENCES

- [1] P. Ammann, J. Pamula, J. Street, and R. Ritchey. A host-based approach to network attack chaining analysis. In *ACSAC '05: Proc. of the 21st Annual Computer Security Applications Conference*, pages 72–84, Washington, DC, USA, 2005. IEEE Computer Society.
- [2] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 217–224, New York, NY, USA, 2002. ACM.
- [3] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and software verification: Model-checking techniques and tools*. Springer-Verlag, Berlin, 2001.
- [4] L. Cardelli. *Bioware Languages*, pages 59–65. Monographs in Computer Science. Springer, New York, 2004.
- [5] L. Cardelli and A. D. Gordon. Mobile Ambients. In *Foundations of Software Science and Computation Structures: First International Conference, FOSSACS'98*, volume 1378 of *LNCS*, pages 140–155, Berlin Germany, 1998. Springer-Verlag.
- [6] L. Cardelli and A. D. Gordon. Types for Mobile Ambients. In *POPL'99: Proc. of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 79–92, New York, NY, USA, 1999. ACM.
- [7] R. Chinchani, A. Iyer, H. Q. Ngo, and S. Upadhyaya. Towards a Theory of Insider Threat Assessment. In *DSN 2005: Int. Conference on Dependable Systems and Networks*, pages 108–117. IEEE Publishing, July 2005. <http://ieeexplore.ieee.org/iel5/9904/31476/01467785.pdf>.

- [8] F. Cuppens and R. Ortalo. Lambda: A language to model a database for detection of attacks. In *RAID'00: Proc. of the Third Int. Workshop on Recent Advances in Intrusion Detection*, pages 197–216, London, UK, 2000. Springer-Verlag.
- [9] V. N. L. Franqueira and R. H. C. Lopes. Vulnerability Assessment by Learning Attack Specifications in Graphs. In *IAS'07: Proc. of the 3rd Int. Symposium on Information Assurance and Security*, pages 161–164, August 2007.
- [10] V. N. L. Franqueira, R. H. C. Lopes, and P. van Eck. Multi-step Attack Modelling and Simulation (MsAMS) Framework based on Mobile Ambients. Technical Report TR-CTIT-08-44, Centre for Telematics and Information Technology (CTIT), University of Twente, Enschede, The Netherlands, June 2008.
- [11] V. Gorodetski and I. Kutenko. Attacks against computer network: Formal grammar-based framework and simulation tool. In A. Wespi, G. Vigna, and L. Deri, editors, *RAID 2002: Proc. of the Fifth Int. Symposium on Recent Advances in Intrusion Detection*, volume 2516 of *LNCS*, pages 219–238. Springer, October 2002.
- [12] D. Ha, S. Upadhyaya, H. Q. Ngo, S. Pramanik, R. Chinchani, and S. Mathew. Insider threat analysis using information-centric modeling. In *Advances in Digital Forensics III*, IFIP International Federation for Information Processing, pages 55–73. Springer, Boston, 2007.
- [13] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International, second edition, June 2004. online version at <http://www.usingcsp.com/cspbook.pdf>.
- [14] K. Ingols, R. Lippmann, and K. Piwowarski. Practical attack graph generation for network defense. In *ACSAC '06: Proc. of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference*, pages 121–130, Washington, DC, USA, 2006. IEEE Computer Society.
- [15] S. Jajodia, S. Noel, and B. O'Berry. Topological Analysis of Network Attack Vulnerability. In *Managing Cyber Threats: Issues, Approaches and Challenges*. Springer-Verlag, Germany, 2005.
- [16] S. Jukna. *Extremal Combinatorics*. Springer, 2000.
- [17] G. Keizer. Mass hack infects tens of thousands of sites. Computerworld, published on January 7, 2008. http://www.computerworld.com/action/article.do?command=viewArticleBasic&taxonomyId=16&articleId=9055858&intsrc=hm_topic. Visited 10-July-2008.
- [18] J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. In *In Proc. Ninth Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 668–677, New York, 1998. ACM Press.
- [19] A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.
- [20] W. Li, R. B. Vaughn, and Y. S. Dandass. An approach to model network exploitations using exploitation graphs. *Simulation*, 82(8):523–541, 2006.
- [21] R. Milner. Pure bigraphs. Technical Report UCAM-CL-TR-614, University of Cambridge, January 2005.
- [22] Nessus. Tenable network security: The Nessus Security Scanner. <http://www.nessus.org>. Visited 10-July-2008.
- [23] S. Noel and S. Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *VizSEC/DMSEC '04: Proc. of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 109–118, New York, NY, USA, 2004. ACM. <http://doi.acm.org/10.1145/1029208.1029225>.
- [24] NVD. National vulnerability database v2. <http://nvd.nist.gov/>. Visited 10-July-2008.
- [25] X. Ou, W. F. Boyer, and M. A. McQueen. A Scalable Approach to Attack Graph Generation. In *CCS '06: Proc. of the 13th ACM Conf. on Computer and Communications Security*, pages 336–345, New York, NY, USA, 2006. ACM. people.cis.ksu.edu/~xou/publications/ccs06.pdf.
- [26] X. Ou, S. Govindavajhala, and A. W. Appel. Mulval: a logic-based network security analyzer. In *SSYM'05: Proc. of the 14th Conf. on USENIX Security Symposium*, Berkeley, CA, USA, August 2005. USENIX Association. www.cs.princeton.edu/~appel/papers/mulval.pdf.
- [27] R. W. Ritchey and P. Ammann. Using Model Checking to Analyze Network Vulnerabilities. In *SP'00: Proc. of the 2000 IEEE Symposium on Security and Privacy*, pages 156–165, Washington, DC, USA, 2000. IEEE Computer Society.
- [28] R. Sawilla and X. Ou. Googling Attack Graphs. Technical Report TM-2007-205, Defense Research and Development Canada, September 2007. http://www.ottawa.drdc-rddc.gc.ca/html/tm_2007_205_e.html.
- [29] B. Schneier. The Ethics of Vulnerability Research. Information Security Magazine, May 2008. <http://www.schneier.com/essay-211.html>.
- [30] J. R. Seeley. The net of reciprocal influence: A problem in treating sociometric data. *Canadian Journal of Psychology*, 3:234–240, 1949.
- [31] O. Sheyner and J. Wing. Tools for Generating and Analyzing Attack Graphs. In *In Proc. of Workshop on Formal Methods for Components and Objects*, LNCS 3188, pages 344–371, Germany, 2004. Springer-Verlag.
- [32] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. In *DISCEX II'01: DARPA Information Survivability Conference and Exposition Conference and Exposition*, volume 2, pages 307–321, Washington, DC, USA, June 2001. IEEE Computer Society.
- [33] S. J. Templeton and K. Levitt. A requires/provides model for computer attacks. In *NSPW'00: Proc. of the 2000 Workshop on New Security Paradigms*, pages 31–38, New York, NY, USA, 2000. ACM.
- [34] L. Williams, R. Lippmann, and K. Ingols. An interactive attack graph cascade and reachability display. In *VizSEC'07: Proc. of the Workshop on Visualization for Computer Security*, pages 221–235. Springer-Verlag, October 2007.