Algebra for capability based attack correlation

Navneet Kumar Pandey, S. K. Gupta, and Shaveta Leekha

Indian Institute of Technology Delhi,
Delhi, India
{npandey, skg, mcs052943}@cse.iitd.ernet.in

Abstract. Most of the existing intrusion detection systems (IDS) often generate large numbers of alerts which contain numerous false positives and non relevant positives. Alert correlation techniques aim to aggregate and combine the outputs of single/multiple IDS to provide a concise and broad view of the security state of network. Capability based alert correlator uses notion of capability to correlate IDS alerts where capability is the abstract view of attack extracted from IDS alerts/alert. To make correlation process semantically correct and systematic, there is a strong need to identify the algebraic and set properties of capability. In this work, we identify the potential algebraic properties of capability in terms of operations, relations and inferences. These properties give better insight to understand the logical association between capabilities which will be helpful in making the system modular. This paper also presents variant of correlation algorithm by using these algebraic properties. To make these operations more realistic, existing capability model has been empowered by adding time-based notion which helps to avoid temporal ambiguity between capability instances. The comparison between basic model and proposed model is exhibited by demonstrating cases in which false positives have been removed that occurred due to temporal ambiguity.

Keywords: - *intrusion detection, capability model, attack scenario.*

1 Introduction

Since long information system and network security experts have made considerable efforts to protect secure systems from exponentially increasing threats, despite this the hackers tools now includes technology that conventional security tools and services cannot sustain. Even critical systems and networks equipped with highly sophisticated security techniques are vulnerable to blended and multi-stage attacks which use stealth and intelligence to strategically compromise a target, escaping detection and penetrating the defences [1].

The surveillance and security monitoring of the network infrastructure is mostly performed using Intrusion Detection Systems (IDSs). Event streams are used by IDS in two different ways, according to two different paradigms: anomaly detection and misuse detection. In anomaly detection systems ([2], [3], [4], [5]

, [6], [7]), historical data about a systems activity and/or specifications of the intended behavior of users and applications are used to build a profile of the normal operation of the monitored system. The intrusion detection system then tries to identify patterns of activity that deviate from the defined profile. Misuse detection systems take a complementary approach ([8], [9], [10], [11], [12]). Misuse detection tools are equipped with a number of attacks descriptions (or signatures) that are matched against the stream of audit data and look for the evidence modeled attack. Misuse and anomaly detection systems have their own advantages and disadvantages [12].

Moreover, intrusion detection, audit and logging systems often provide sensory feedback data that cannot be effectively analyzed as they flag thousands of alerts which may overwhelm the analysts. Most of them are false positive and non relevant positives. Non relevant positives are alerts that correctly identify an attack, but the attack fails to meet its objective. Several alert correlation techniques have been proposed including approaches based on similarity between alert attributes, using pre-defined attack scenarios, pre/post-conditions of attacks, using multiple networks and auditing tools. Each technique has its own advantages and disadvantages, therefore none of the technique dominate the other [13]. For example similarity based approaches lack on finding attack step sequence, pre-defined attack scenario only work well for known scenarios, pre-post condition based approaches can detect new scenario but defining these conditions is itself error-prone and enumeration of these conditions is non trivial task whereas multiple information source based approaches suffer from sheer volume of data to process.

The require/provide model [14] used for alert correlation states that in a multistage intrusion comprising of a sequence of attacks, the early attacks acquire certain advantages, like information about the system under attack and the ability to perform actions on that system and use these advantages to support the later attacks that require them. Capability model [15] captures this notion of attacker capability and use it for logical alert correlation.

In this work we give algebraic property of capabilities. These properties give better understanding of capability characteristics. These characteristics help in designing the correlation process in a systematic and modular fashion. We group the identified algebraic properties into three classes i.e. operations, relations and inferences. Operations include join, split etc. which represent basic manipulation using one or more capability instance. Relations include overlapped, mutual exclusive, independent relations between capability instances. These relations help in identifying the preconditions to allow specific operations. As the whole system is based on require/provide model therefore to determine whether a capability satisfies a required capability, inferences are used. Inferences include comparable, resulting etc., which enumerate the possible inferences from different real life views. The paper also gives three derived version of the correlation process from basic correlation process using these algebraic properties. We have also enriched the basic capability model by adding time parameter in the definition of capability. This helps to remove temporal ambiguity between capability instances. The comparison between basic model and proposed model is exhibited by demonstrating cases.

In this work, we consider attack from single source to a single destination. However this can be easily generalized for distributed kind of attacks where multiple sources/multiple destinations are involved.

The remainder of this paper is structured as follows: Section 2 presents related work on alert correlation and capability model. Section 3 presents the proposed modified capability model. Section 4 presents the capability algebra. Section 5 provides a detail of the correlation algorithm and also shows the case study in which results have been improved. In the section 6, alternate way of correlation algorithm has been discussed with their benefits and pitfalls. Finally, Section 7 draws conclusions and outlines future work.

2 Related Work

In order to minimize false positives, the alert correlation techniques have been widely studied. Pouget et. al. [16] has classified these techniques into eight classes.(i) Rule based (ii) Scenario-based (iii) Uncertainty reasoning (iv)Time reasoning (v) State transition graphs (vi) Neural networks (vii) Bayesian belief networks (viii) Context reasoning. ([17] [18] [19] [20] [21] [22] [23] [24] [25])

To define logical relation between different attacks, Templeton and Levitt [14] proposed the require/provide model based on the system states. The proposed JIGSAW language for correlation uses simple predicates to define system state. However, they do not provide a systematic approach to develop predicates. Another similar approach given in Ning et. al. [26] also defines predicate for alert correlation. However definition of predicates used is ambiguous and also the paper does not give consistent way to develop it.

Our work is motivated by Jingmin et. al. [15] which uses capability model for attack correlation. This capability model uses capability as basic building block and used it for developing several algorithms in correlation based on alert abstraction and inference rules. Their work also shows that the approach is capable of handling missing attacks and is promising at alert fusion and correlation. However the paper does not discuss the algebraic operations and relation between capabilities.

In our work, we give a new definition of capability which is closer to the semantics of real life attacker capability and also avoid temporal ambiguity between definitions of capabilities. We also give several algebraic operations. The work also identified relations that exist between capabilities and derived inference rules to define logical association between two capabilities. These relations are helpful in understanding the capabilities properly and for defining the semantics of algebraic operations which in turn are used in correlation algorithm.

Li et. al [27] and Wijeskera et. al. [28] defined algebra for access control policies [29] which is related to capability. However the definition of capability is generalized as compared to access control and is not limited to a specific service e.g it applies for database, network, OS etc.

3 Capability Model

In the capability model, capability represents facilities and accesses that an attacker gains by making a connection. Capability describes the ability of an attacker during intrusion. An attacker can have many capabilities at a particular instance that may or may not belong to the same intrusion.

3.1 Capability Model

Let D be a set of network addresses, C be a set of credentials, A be a set of actions, SP be a set of pair of services and their property and $[t_1, t_2]$ is a time interval where t_1 and t_1 are constant time and $t_2 > t_1$.

Definition 1 (Capability). *Capability is a six-tuple capability = (source, destination, credential, action, (service, property), interval)*
where source $\in D$, destination $\in D$, credential $\in C$, action $\in A$, service $\in S$, property $\in P$, interval $\in [t_1, t_2]$ in which capability is valid. It may be noted that we have added the attribute interval to the definition of capability given by Jingmin et. al. [15].

Example 1. The capability (pushpa, dblab, user1, read,('/etc/passwd', content), $From : \langle 1997 - 07 - 16T19 : 20 : 30 + 01 : 00 \rangle$) means there is capability from host pushpa (source) to host dblab (destination) with credential user1 for read action of content of the file "/etc/passwd" in interval $[1997 - 07 - 16T19 : 20 : 30 + 01 : 00, \infty]$.

3.2 Attributes

In the modified capability model the definition of Source, Destination, Action and Credential are same as those given by [15]. As service and property attributes are tightly coupled, therefore they have been merged into a single attribute of two tuples. This modification helps us in defining operation more clearly. Interval is a new attribute added in the definition of capability. Table 1 shows description of each attribute along with examples.

3.3 Direct and Indirect capability

After connection has been established from source (attacker host) to destination (victim host) by attacker, he/she may gain some privilege or knowledge. This type of capability will be called as direct capability. Implication of direct capability will be called indirect capability. For example if attacker succeed to make connection for reading a file in which mail and credit card passwords have been stored then direct capability is being able to read that file and indirect capability is being able to use mails and credit cards.

Table 1. Attributes of Capability

Attribute	Description	Example
Source	source address	IP:10.20.3.2, Ethernet:006097981E6B etc..
Destination	destination address	IP:10.20.3.2, Ethernet:006097981E6B etc..
Action	Actions that can be performed by an attacker	read, write, communicate etc.
Credential	Credential using which action can be done	root, system, user etc.
Service	Service used by connection	IIS 3.0, \etc\passwd etc.
Property	Property of service	version, content etc.
Interval	Capability Time interval during	From:tstamp, Between:tstamp+tt, at:tstamp etc.. Where tstamp time-stamp and tt is length

3.4 Significance of time parameter

Time parameter which is denoted by the interval is a crucial parameter in reducing the false belief that each capability will last forever during correlation. Some capabilities especially indirect capabilities that depend on service running in target host and may only be valid under certain conditions. It is not necessary that these conditions will always be present in the network or system for example in case where service is scheduled to run for a specific duration. Therefore, it is clear that these conditions are bound to the validity of a session for capability and cannot be assumed that once gained by attacker they will always be with him. Ambiguity due to the assumption that capability once gained will always exist is called temporal ambiguity.

Time interval is represented by predicate $between : [t1, t2]$ which shows that capability will exist from timestamp t1 to timestamp t2. It is also true that some capabilities (e.g. of knowledge type) once gained will always exist with the attacker. To denote it *from* predicate is used i.e. $From : t1$. For example if attacker gained that target machine running on Solaris OS at time t1 then interval of this capability is $From : t1$.

There are various sources of information that may help in specifying the closed time interval of the capability e.g. host integrity checker, HIDS etc... From these sources it can be identified that capability gained earlier is no longer valid. Other sources may be administrator knowledge especially when some service is allowed for a limited period. For example there is one connection having capability to execute a program in host at time t_1 and later at time t_2(where $t_2 > t_1$) service has been blocked . In this case former established connection will not have same capability as in earlier.

Timestamp can be taken in different format. In this model following format of timestamp has been used:-

YYYY-MM-DDThh:mm:ss.sTZD

For example 2007-07-16T19:20:30.45+01:00

where YYYY = four-digit year, MM = two-digit month (01=January, etc.), DD = two-digit day of month (01 through 31), hh = two digits of hour (00 through 23) (am/pm NOT allowed), mm = two digits of minute (00 through 59), ss = two digits of second (00 through 59), s = one or more digits representing a decimal fraction of a second, TZD = time zone designator (Z or +hh:mm or -hh:mm).

3.5 Correlation process

The correlation process is based on the require/provide model in which capabilities gained from the previous attacks are used to satisfy the prerequisite of subsequent attacks. The model has following components.

H-alert An H-alert is a three tuple (require, provide, raw) and represents transformed object of alert in terms of capability, where

Require: - It is a set of capabilities that are required for alert to be a true attack.

Provide: - It is a set of gained capabilities after an alert has been generated. Most IDS generate two kinds of alerts for each attack step, one for incoming traffic in victim host and the other for outgoing traffic from victim host. Alerts that have been generated for incoming traffic may be either successful or failure. This information is available in outgoing traffic. Attacker may even gain capability in failed attack therefore provide set contain those capabilities which have been gained by either successful attack or failure attack whichever is applied.

Raw: - Raw contains other information available in alert message such as time of alert generated, traffic direction etc.

M-Attack An M attack is a three tuple $(haset, capset, tmpstmp)$ which is a collection of correlated alerts where *haset* is a set of alerts $(h - alerts)$, *capset* is a set of capabilities provided by h-alerts in *haset* and *tmpstmp* is the timestamp of last correlated alert which can be considered as *timestamp* of M.

Capabilities are tagged to be considered as mandatory and optional (can be ignored while correlation in some conditions) in the *capset*.

In other words, M-attack represents the set of correlated alerts and correlation process correlate the newly generated alert (H-alert) with these M-attack/M-attacks. Overall correlation algorithm has been explained in section 5.1.

4 Capability Algebra

To modularize the whole correlation process, it is necessary to analyze the properties and characteristics of the capability model. By identifying the algebraic properties of capabilities, capability extraction from IDS signatures can be made automatic. It gives better insight and puts clarity and separation between definitions of capabilities. This also helps to determine the level of granularity in defining the capability. Capability algebra can be divided into three groups i.e. operations, relations and inferences. These are described in the following section.

For comparing two capabilities, it is required to determine the relations between two capabilities, their inferences and relevant operations. It may be noted that attributes of capabilities form a hierarchy. We identify following operations, relations and inferences base on this hierarchy.

4.1 Operations

Operations represent manipulations in the capabilities required in the correlation process. There are three kinds of operations identified for the correlation process.

Join Join operation merges two capabilities in presence of a join condition (see Algorithm 1). Two capabilities can be joined if both capabilities belong to the same source and destination. Also other attributes should be same except an attribute based on which join operation will be performed. For example capability C_1 (srcS, dstD, daemon, block, (ftp process, port 80), from:2008-07-16T19:20:30.45+01:00) is join capability of C_2 (srcS, dstD, daemon, block, (ftp process, port 80), between:[2008-07-16T20:10:30.00+01:00, 2008-07-16T21:00:00 .00+01:00]) and C_3 (srcS, dstD, daemon, block, (ftp process, port 80), from:2008-07-16T19:20:30.45+01:00).

Algorithm 1 Joining two capabilities

```
Require: Two capabilities $C_1$ and $C_2$
Ensure: Resultant capability $C_3$ if $C_1$ and $C_2$ can be joined else $NULL$.
  Let $S = (cred, action, (service, property))$
  procedure JOIN($C_1$,$C_2$)
      if $C_1.src = C_2.src$ and $C_1.dst = C_2.dst$ then
          if $\forall A_i \in S$ s.t. $C_1.A_i = C_2.A_i$ then return $C_1$
          else if $\exists$ an attribute $A_k \in S$ s.t. $C_1.A_k \neq C_2.A_k$ and $\forall A_i \in S - A_k$,
              $C_1.A_i = C_2.A_i$ then return $C_3$ with $C_3.A_k = C_1.A_k \cup C_2.A_k$
          else if $C_1.interval$ and $C_2.interval$ overlaps and other attributes are same
              then return $C_3$ with $C_3.interval = C_1.interval \cup C_2.interval$
          end if
      end if
      return NULL
  end procedure
```

Join operation reduces the redundancy which in turn minimizes the number of comparisons (while finding inferences) between h-alert require set and M-attacks Capset (see section 5) during correlation process.

Split Split breaks a capability into two capabilities based on the given attribute and its value. For example (srcS, dstD, userU, modify, (file, content), from:t1) can be split in (srcS, dstD, userU, append, (file, content), from:t1) and (srcS, dstD, userU, delete, (file, content), from:t1). It may be noted that split is semantically inverse of join operation. Split can be performed on the attributes (a) Action (b) Credential (c) Property (d) Time, if their value is composite[1]. After split resultant capabilities would have same values of src(source), dst (destination) and service, however no split will be done on the basis of these attributes.

Algorithm 2 Split a capability into two capabilities for given attribute

```
Require: Capability $C$, Attribute $A$ and value of attribute $v$
Ensure: Resultant capability $C_1$ and $C_2$ if $C$ can be split else $C$
  procedure SPLIT($C$,$A$,$v$)
      if $C.A$ is not $composite^1$ then return $C$
      else $C_1.A = v$, $C_2.A = reduce(C, A, v)$, $\forall A_i \in S - A$ set $C_1.A_i = C_2.A_i = C.A_i$
          where S=(src, dst, cred, action, (service, property), interval)
          return $C_1$ and $C_2$
      end if
  end procedure
```

Split is a special case of Reduce (defined in section 4.1) where one capability C when split in two capabilities C_1 and C_2 then by joining C_1 and C_2 we can form C again which may not be the case in Reduce. In other words, split is lossless reduction (see Algorithm 2).

Algorithm 3 Reducing a capability

```
Require: Capability $C$, Attribute $A$ (must be composite) and $v$ is value of A
Ensure: Reduced capability $C_d$
  Let $S = (src, dst, cred, action, (service, property), interval)$
  procedure REDUCE($C$,$A$,$v$)
      Create a new capability $C_d$ with $C_d.A = C_d.A - v$,
      $\forall A_i \in S - A$ set $C_d.A_i = C.A_i$, return $C_d$
  end procedure
```

Reduce The Reduce operation weakens a capability by reducing strength of any of its attribute. For example capability (srcS, dstD, root, modify, (program, code), from:t1) can be reduced to (srcS, dstD, userU, modify, (file, content), from:t1). Difference between *split* (Algorithm 2) and *reduce* (Algorithm 3) is

[1] Attribute A is composite if it contains multiple values or a value that can be divided into distinct components for eg. RW action can be split into R and W actions.

that *split* operation always gives two capabilities whereas in the case of *reduce* it is not mandatory that reduced part will be a capability.

Subtract The Subtract operation takes two capabilities C_1 and C_2 and returns C_3 which is deduction of capability C_2 from C_1. For example (srcS, dstD, userU, send, (IIS, Ftp), from:t1) is result of subtraction of (srcS, dstD, userU, receive, (IIS, Ftp), from:t1) from (srcS, dstD, userU, communicate, (IIS, Ftp), from:t1).

Algorithm 4 Capability Subtraction

```
Require: Capabilities $C_1$ and $C_2$
Ensure: Resultant capability $C_s$
  procedure SUBTRACT($C_1$,$C_2$)
      if $C_1.src$=$C_2.src$ and $C_1$.dst=$C_2$.dst then
          if ∃ an attribute $A \in S$ s.t. $C_1.A \neq C_2.A$ and $\forall B \in S - A$, $C_1.B = C_2.B$
              where S=(action, (service, property), interval) then
              $C_s$= Reduce($C_1$,$A$,$C_2.A$)
          else $C_s = C_1$.
          end if return $C_s$
      end if
  end procedure
```

Subtract is similar to reduce in which minuend Capability is reduced by subtrahend capability. For the substraction it is necessary that both capabilities have same source and destination and only one attribute is different among the rest (see Algorithm 4).

4.2 Relations

A relation represents a logical association between two or more capabilities. Following three types of relations are identified for the correlation process.

Overlap Two capabilities overlap if there exists a common capability between them (see Algorithm 5). For example capabilities (SLab, Dlab, RW, (/home/user1, content), user1, from:t1) and (SLab, Dlab, WX, (/home/user1, content), user1, from:t1) overlap because the capability (SLab, Dlab, W, (/home/user1, content), user1, from:t1) is common in both. If any of the following attributes are common in two capabilities, then there is overlapping: (a) Interval (b) Credential (c) Action and property of service.

Independent Two capabilities are independent if they cannot be joined (see Algorithm 6).In other words, two capabilities are called independent if either both have different source/destination or have different values of more than one attributes among the rest of attributes. For example capabilities (SLab, Dlab,

Algorithm 5 Test two capabilities whether they are overlap

```
Require: Two capabilities $C_1$ and $C_2$
Ensure: $true$ or $false$
  Let $S = (src, dst, cred, action, (service, property), interval)$
  procedure OVERLAP($C_1$,$C_2$)
      if ($C_1.interval$ and $C_2.interval$ overlaps) and $\forall A_i \in S - \{interval\}$
          s.t. $C_1.A_i = C_2.A_i$ then return $true$
      else if $\exists$ a credential $cred_k$ s.t. $cred_k \in C_1.cred \cap C_2.cred$ and $\forall A_i \in S - \{cred\}$
          s.t. $C_1.A_i = C_2.A_i$ then return $true$
      else if $\exists$ an action $act_k$ s.t. $act_k \in C_1.action \cap C_2.action$ and $\forall A_i \in S - \{action\}$
          s.t. $C_1.A_i = C_2.A_i$ then return $true$
      else if $\exists$ a property $p$ s.t. $p \in C_1.property \cap C_2.property$ of the same service and
          $\forall A_i \in S - \{(serivce, property)\}$ s.t. $C_1.A_i = C_2.A_i$ then return $true$
      else return $false$
      end if
  end procedure
```

W, /home/user1, content, user1, from:t1) and (SLab, Dlab, X, httpd, (Apache 3.2, apacU), from:t1) independent.

Algorithm 6 Test two capabilities whether they are Independent

```
Require: Two capabilities $C_1$ and $C_2$
Ensure: $true$ or $false$
  procedure INDEPENDENT($C_1$,$C_2$)
      if join($C_1$,$C_2$) is NULL then return $true$
      else return $false$
      end if
  end procedure
```

Mutual Exclusive Two capabilities are mutually exclusive if their corresponding attribute's value cannot coexist (see Algorithm 7). Mutually exclusive capabilities are less likely to belong to the same attack. This information helps in reducing false correlation. For example capabilities (SLab, Dlab, R,(/etc/passwd, content), user1, from:t1) and (SLab, Dlab, X, IIS, Ver4.0, user1, from:t1) are mutually exclusive.

The conflict set used in algorithm 7 is a knowledge base having pair of attributes that cannot coexist e.g. service of windows and Linux cannot exist simultaneously in the same IP.

4.3 Inferences

Inference means causal relationship involved in process of deriving result or making a logical judgment on the basis of known evidence. Inferences identified

Algorithm 7 Test two capabilities whether they are Mutual Exclusive

```
Require: Two capabilities $C_1$ and $C_2$
Ensure: $true$ or $false$
  procedure MUTUAL-EXCLUSIVE($C_1$,$C_2$)
      if $\exists$ an attribute $A$ s.t. conflict($C_1.A$,$C_2.A$) is $true$ then return $true$
      else return $false$
      end if
  end procedure
```

here are used in comparing capabilities of require set of h-alert with capabilities in M-attack's capability set based on require/provide model during correlation process. Almost all inferences given in this section are same as given in [15].

Comparable Inference Comparable inference denotes semantic comparability of two capabilities. Two capabilities can be compared only if they hold same type of service and property while other attributes must be same. This inference will be used to correlate two capabilities to construct attack scenario. Capabilities can be correlated only if required capability can be satisfied with some of the capability of M-attack set by comparable inference (see Algorithm 8).

Algorithm 8 Test whether C_1 and C_2 can be compare directly

```
Require: Two capabilities $C_1$ and $C_2$
Ensure: $true$ or $false$
  procedure COMPARABLE($C_1$,$C_2$)
      if $\forall A_i \in$ { src, dst, cred, action }, $C_1.A_i = C_2.A_i$, with overlapped time interval
          and both have same type of service and property then return $true$
      else return $false$
      end if
  end procedure
```

Service and property belong to the same type when services belong to same category as given in [15]

Resulting Inference In many cases logical relations between capabilities cannot be represented by comparable inference due to strict conditions. One capability is the resulting inference of other if it gives the other capability on its execution. These inferences are nothing but a single step of whole correlation process and are used in making attack scenario through multi step correlations (see Algorithm 9).

Administrator knowledge, topology of network are some of the major information sources to identify the capabilities which can be logically derived by exercising a capability.

Algorithm 9 Test whether C_2 is resulting inferable from C_1

```
Require: Two capabilities $C_1$ and $C_2$
Ensure: $true$ or $false$
  procedure RESULTING_INFERABLE($C_1$,$C_2$)
      if exercise of $C_1$ logically derive $C_2$ then return $true$
      else return $false$
      end if
  end procedure
```

Other Inferences Several other inferences are also possible along with the given inferences. For example compromise inference and external inference as given by Jingmin et. al. [15].Through compromise inference one capability can be inferred from other capability for compromising the destination machine (executing arbitrary program).Capability C_1 can be externally inferred from capability C_2 if C_2 is the capability to execute arbitrary program on destination machine which is the source of C_1.

5 Correlating alert using modified capability model

5.1 Correlation Algorithm

Correlation algorithm correlates new h-alert (created from alert generated by IDS) with the existing M-attacks. Initially there is a set of M-attacks M. Whenever a new alert comes, then it is abstracted into an h-alert. Correlation algorithm searches minimal and ordered subset of M-attacks from M such that all the require capabilities of h-alert are satisfied by the capabilities of a subset of M-attacks. Then the algorithm combines h-alert with the identified subset of M-attacks in a single M-attack. This M-attack contains all capabilities of selected M-attacks along with the h-alert's provide capabilities. This new M-attack replaces the subset of M-attacks. The whole correlation process is presented in Algorithm 10. Algorithm 11 shows the search procedure of M-attacks that satisfy the required capabilities of newly generated h-alert.

5.2 Case Study

We have extended the existing capability model by adding a new attribute i.e. time. The modified capability improves the correlation by reducing the cases of false correlation and by increasing correlation strength. Some of the major cases are as follows.

Case1: We have a require capability C_1 (srcX, dstX, credX, $\{RW\}$,(/home/user1, content), intvX) of a newly generated h-alert and two M-attacks M_1 and M_2 in M-attack set having capabilities C_3 (srcX, dstX, credX, $\{R\}$,(/home/user1, content), intvX), C_3(srcX, dstX, credX, $\{W\}$,(/home/user1, content), intvX) in their *capset* respectively. Using former approach capability C_1 cannot be

Algorithm 10 Correlate a new h-alert which is an abstract form of recently came alert with M-attacks

```
Require: h-alert $h_1$ and set of M-attacks $M$=$\{M_1, \cdots M_n\}$
Ensure: a new M-attacks set M'=$\{M_1$,.... $M_k\}$
  procedure CORRELATION ALGORITHM($h_1$,$M$)
      Find a minimal and ordered subset $M^k$ of set $M$ (as given in Algorithm 11)
      such that h1.requires is satisfied by capabilities in M-attacks of set $M^k$
      if $M^k \neq \phi$ then Make new M-attack $M_{new}$ as
          $M_{new}.capset = C_M \cup h_1.provide$ where $C_M = \bigcup_i M_i^k.capset$ and $M_i^k \in M^k$,
          $M_{new}.haset = h_M \cup h_1$ where $h_M = \bigcup_i M_i^k.haset$ and $M_i^k \in M^k$
          and replace all M-attacks in $M^k$ by $M_{new}$
      else Make new $M_{new}$ as $M_{new}.capset = h_1.provide$ and $M_{new}.haset = h_1$
      end if
      $M_{new}$.timestamp equal to the timestamp of newly correlated alert.
  end procedure
```

correlated with either of M-attacks (M_1 or M_2) capability because the action attribute of C_1 cannot directly be compared with that of C_2 or C_3. Therefore, the former approach is unable to correlate it. But in the modified approach when C_1 and C_2 will be correlated, C_1 will reduce to (srcX, dstX, credX, W,(/home/user1, content), intvX) and it is directly correlated with C_3 i.e. C_1 is correlated by $M_1 \cup M_2$. Consequently, the enhanced model is able to detect these kinds of true correlations that would have gone undetected in earlier approach. These kind of cases have been handled in the modified approach because of flexibility by defined operations.

Case2: Consider another case where the require set of an incoming h-alert is satisfied by the *capset* of M-attacks M_1, M_2 and there exists a capability in M_1 which is mutually exclusive of other capability that belongs to M_2. In this case M_1 and M_2 actually have no correlation. But the former approach could correlate these kind of capabilities. Whereas, in the proposed model such capabilities are not correlated because they are mutually exclusive and logically donot belong to the same attack.

For example a capability C_1 (eth0:12ffdd3453, eth0:12ffee1234, credX, $\{RW\}$, (/home/user1, content), intvX) belongs to M_1 and other capability C_2 (srcX, dstX, credX, $\{RW\}$, (IIS, content), intvX) belongs to M_2.

Administrator knowledge, services running in the network, topology of network are the major sources of domain knowledge in identifying the mutual exclusive capabilities discussed in section 4.2.

Case3: Modified process also handles the correlation conflicts that arise due to temporal ambiguity as explained in section 3.4. For example, suppose attacker has a capability to read and write in a host H, then attacker can also read and write the mail of a user whenever he opens his mail account on that machine i.e. attacker will have the capability of reading/writing mail from a particular user account only for the duration in which the user is logged in. However in this case, there is no upper limit of interval for

Algorithm 11 Find a minimal and ordered subset M^k of set M

```
Require: h-alert $h_1$ and set of M-attacks $M$={$M_1, \cdots M_n$}
Ensure: M-attacks set M'
 1: procedure FIND_MIN_ORDERED_SUBSET($h_1$,$M$)
 2:     Order all M-attacks based on decreasing Timestamp and let $cap_{req\_set}$ is set of
        capabilities in $h_1.require$, $CapM_{sat} = \phi$ and $M_{result} = \phi$
 3:     for all M-attacks $M_i \in \{M_1, \cdots M_n\}$ do find subset $cap_{satisfied} \subseteq cap_{req\_set}$
        $inferable$ (see section 4.3) from $M_i.capset$, $CapM_{sat} = CapM_{sat} \cup cap_{satisfied}$
 4:         if $CapM_{sat} = h_{req\_set}$ then $M_{result} = M_{result} \cup M_i$ and return $M_{result}$
 5:         else if $cap_{satisfied} \neq \phi$ then
 6:             $cap_{req\_set}$ =$cap_{req\_set}$-$cap_{satisfied}$ and $M_{result} = M_{result} \cup M_i$
 7:         else
 8:             find $cap_{sub} \in cap_{req\_set}$ that can be obtained from $M_i.capset$ by subtract.
 9:             if $cap_{sub} \neq \phi$ then
10:                 $cap_{req\_set} = cap_{req\_set} - cap_{sub}$ , and $M_{result} = M_{result} \cup M_i$
11:             end if
12:         end if
13:     end for
14:     return $\phi$
15: end procedure
```

reading/writing other files. To avoid this ambiguity time attribute has been added with every capability.

Apart from the cases discussed above, there are several other cases where proposed model helps in making overall process efficient. For example *Join* operation helps in reducing the redundancy which in turn saves the number of comparisons while correlations. Suppose there are two capabilities C_1 (srcX, dstX, credX, $\{RW\}$, (/, content), intvX) and C_2 (srcX, dstX, credX, $\{RW\}$, (/home/, content), intvX) then we can join these two into one capability as they are forming contain-ship relation. Therefore it is clear that if two capabilities of M-attack's cap set are joined then further correlation needs only one comparison instead of two. Overlapped and independent relations help in defining join condition accurately to test unambiguously that two capabilities can be joined or not.

6 Discussion and other issues

In this section other possible ways of correlation process are discussed. It is clear that join algorithm has significant impact in minimizing the number of comparisons in correlation because it combines the capabilities in M-attacks's capset. However join itself is costlier operation in terms of time as described below. Following are the alternate methods of doing correlation using various combinations of join and split.

Algorithm 12 (Alternate Method 1) Correlate a new h-alert which is an abstract form of recently came alert with M-attacks

```
Require: h-alert $h_1$ and set of M-attacks $M$=$\{M_1, \cdots M_n\}$
Ensure: a new M-attacks set M'=$\{M_1$,.... $M_k\}$
  procedure CORRELATION ALGORITHMII($h_1$,$M$)
      Find a minimal and ordered subset $M^k$ of set $M$ such that h1.requires is
      satisfied by capabilities in M-attacks of set $M^k$ using algorithm 11
      if $M^k \neq \phi$ then Make new $M_{new}$ as
          $M_{new}.capset = C_M \cup h_1.provide$ where $C_M = \bigcup_i M_i^k.capset$ and $M_i^k \in M^k$,
          $M_{new}.haset = h_M \cup h_1$ where $h_M = \bigcup_i M_i^k.haset$
          and $M_i^k \in M^k$ and replace all M-attacks in $M^k$ by $M_{new}$
      else Make new $M_{new}$ as $M_{new}.haset = h_1.provide$ and $M_{new}.haset = h_1$
      end if
      for all pair of capabilities ($C_i$,$C_j$) in $M_{new}.capset$ do $C_k$=join($C_i$,$C_j$)
          if $C_k \neq NULL$ then replace $C_i$ and $C_j$ by $C_k$ in $M_{new}.capset$
          end if
      end for
      $M_{new}$.timestamp equal to the timestamp of newly correlated alert.
  end procedure
```

6.1 Alternate Method 1

In this method after the correlation, algorithm 12 joins capabilities within each M-attack i.e. within each M-attack if two or more capabilities can be joined then they are joined to minimize the number of capabilities in capset and removes the redundancy if it is there. The minimal set search algorithm is same as algorithm 11.

It may be noted that the method minimizes the number of comparisons while searching for the minimal set of M-attacks because of lesser number of capabilities in each M-attack's capset.

However join operation is a costlier operation. For example in a M-attack's capset if there are n capabilities then join operation is called for every pair of subset of capabilities which is exponential because the join operation need to be called recursively until no more joins are possible.

6.2 Alternate Method 2

In this method capabilities in new h-alert's require set are split into minimal granularity based on their composite attributes.

In this case, we do not use join operation for correlation as it is costly. By using split operation, the granularity of each attribute of every capability will become one. Consequently, this will make the comparisons easier. Also we do not need the subtract operation as all capabilities are in their minimal reduced form. Minimal set search algorithm is same as algorithm 11 except the subtract operation in steps 8,9 and 10.

Algorithm 13 (Alternate Method 2) Correlate a new h-alert which is an abstract form of recently came alert with M-attacks set

```
Require: h-alert $h_1$ and set of M-attacks $M$=$\{M_1, \cdots M_n\}$
Ensure: a new M-attacks set M'=$\{M_1$,.... $M_k\}$
  Let $S = \{cred, action, \{service, property\}, interval\}$
  procedure CORRELATION ALGORITHMIII($h_1$,$M$)
      for all capabilites $C_i \in h_1.require$ do
          for all attributes $A \in S$ do
              if A is composite then split $C_i$ into minimal granularity based on A
              end if
          end for
      end for
      Find a minimal and ordered subset $M^k$ of set $M$ such that h1.requires is
      satisfied by capabilities in M-attacks of set $M^k$ using algorithm 11
      if $M^k \neq \phi$ then Make new $M_{new}$ as
          $M_{new}.capset = C_M \cup h_1.provide$ where $C_M = \bigcup_i M_i^k.capset$ and $M_i^k \in M^k$ ,
          $M_{new}.haset = h_M \cup h_1$ where $h_M = \bigcup_i M_i^k.haset$ and $M_i^k \in M^k$
          and replace all M-attacks in $M^k$ by $M_{new}$
      else Make new $M_{new}$ as $M_{new}.capset = h_1.provide$ and $M_{new}.haset = h_1$
      end if
      $M_{new}$.timestamp equal to the timestamp of newly correlated alert.
  end procedure
```

However in some cases we may end up in split where it may not be required. For example capability containing action RW has been split into two capabilities with action R and W in M-attack's capset. A new required capability with same RW action comes, then we split it into R and W, which require two comparisons. Indirectly we may be increasing the comparisons unintentionally as number of capabilities in the capset of M-attack have increased in some cases.

6.3 Alternate Method 3

This method is a combination of Alternate Method 1 and Alternate Method 2 which splits the capabilities of h-alert's require set into minimal granules and after correlation, joins the capabilities in the newly formed M-attacks's capset which can be joined.

The method wipes out pitfall of pervious alternate methods as split has been used initially to simplify the comparisons and later on join has been used in each M-attack's capset to minimize the number of capabilities which consequently minimizes the number of comparisons . However this method is more costly than previous in time complexity.

7 Conclusion

In this work we have defined time parameter and shown its impact in reducing false correlation. We have also identified and defined relations between ca-

Algorithm 14 (Alternate Method 3) Correlate a new h-alert which is an abstract form of recently came alert with M-attacks set

```
Require: h-alert $h_1$ and set of M-attacks $M$=$\{M_1, \cdots M_n\}$
Ensure: a new M-attacks set M'=$\{M_1$,.... $M_k\}$
  Let $S = \{cred, action, \{service, property\}, interval\}$
  procedure CORRELATION ALGORITHMIV($h_1$,$M$)
      for all capabilites $C_i \in h_1.require$ do
          for all attribute $A \in S$ do
              if A is composite then split $C_i$ into maximum granularity based on A
              end if
          end for
      end for
      Find a minimal and ordered subset $M^k$ of set $M$ such that h1.requires is
      satisfied by capabilities in M-attacks of set $M^k$ using algorithm 11
      if $M^k \neq \phi$ then Make new $M_{new}$ as
          $M_{new}.capset = C_M \cup h_1.provide$ where $C_M = \bigcup_i M_i^k.capset$ and $M_i^k \in M^k$,
          $M_{new}.haset = h_M \cup h_1$ where $h_M = \bigcup_i M_i^k.haset$ and $M_i^k \in M^k$
          and replace all M-attacks in $M^k$ by $M_{new}$
      else Make new $M_{new}$ as $M_{new}.capset = h_1.provide$ and $M_{new}.haset = h_1$
      end if
      for all pair of capabilities ($C_i$,$C_j$) in $M_{new}.capset$ do $C_k$=join($C_i$,$C_j$)
          if $C_k \neq NULL$ then replace $C_i$ and $C_j$ by $C_k$ in $M_{new}.capset$
          end if
      end for
      $M_{new}$.timestamp equal to the timestamp of newly correlated alert.
  end procedure
```

pabilities, operations on capability and derived Inference rules along with their semantic that have been used in correlation process. The framework is made systematic, consistent and defined properly with algorithms. Comparison between the previous model and the proposed model is exhibited by demonstrating cases where the correlated alerts were not captured by the old model, but are taken care in our proposed model.

By making the correlation process modular we have simplified the whole correlation process. This makes system more understandable for even non security expert. This approach helps in facilitating the process flexibility and easy enhancement. With this systematic model, the system can be automated and adaptive to optimizations.

Part of the future work will be to optimize algorithms and to achieve better performance. One possibility would be to optimize the algorithm of join operation and to use that in given alternate correlation algorithm (in section 6). This would help in making whole system real time with low false rate.

Another future work will be to model the defence capability of security personnel. This defence capability will help the administrator in identifying his position against the attacker's capability. There is also scope in the future work to develop language for whole framework.

8 Acknowledgment

This project was supported by a grant from Dept of Information Technology, Govt of India. We thank the database and security group members at IIT Delhi for their reviews and valuable comments to improve this paper.

References

1. Dawkins, J., Hale, J.: A systematic approach to multi-stage network attack analysis. Information Assurance Workshop, 2004. Proceedings. Second IEEE International (8-9 April 2004) 48–56
2. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. **13**(2) (1987) 222–232
3. Gosh, A.K., Wanken, J., Charron, F.: Detecting anomalous and unknown intrusions against programs. In: ACSAC '98: Proceedings of the 14th Annual Computer Security Applications Conference, Washington, DC, USA, IEEE Computer Society (1998) 259
4. Javits, Valdes: The NIDES statistical component: Description and justification. http://www.csl.sri.com/papers/statreport. (Mar 1993)
5. Ko, C., Ruschitzka, M., Levitt, K.: Execution monitoring of security-critical programs in distributed systems: a specification-based approach. In: SP '97: Proceedings of the 1997 IEEE Symposium on Security and Privacy, Washington, DC, USA, IEEE Computer Society (1997) 175
6. Kruegel, C., Vigna, G.: Anomaly detection of web-based attacks. In: CCS '03: Proceedings of the 10th ACM conference on Computer and communications security, New York, NY, USA, ACM (2003) 251–261
7. Warrender, C., Forrest, S., Pearlmutter, B.A.: Detecting intrusions using system calls: alternative data models. Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on Security and Privacy (1999) 133–145
8. Paxson, V.: Bro: a system for detecting network intruders in real-time. Computer Networks **31**(23-24) (1999) 2435–2463
9. Neumann, P.G., Porras, P.A.: Experience with emerald to date. In: Proceedings of the Workshop on Intrusion Detection and Network Monitoring, Berkeley, CA, USA, USENIX Association (1999) 73–80
10. Roesch, M.: Snort - lightweight intrusion detection for networks. In: LISA '99: Proceedings of the 13th USENIX conference on System administration, Berkeley, CA, USA, USENIX Association (1999) 229–238
11. Vigna, G., Kemmerer, R.A.: Netstat: a network-based intrusion detection system. J. Comput. Secur. **7**(1) (1999) 37–71
12. Eckmann, S.T., Vigna, G., Kemmerer, R.A.: Statl: an attack language for state-based intrusion detection. J. Comput. Secur. **10**(1-2) (2002) 71–103
13. Xu, D., Ning, P.: Alert correlation through triggering events and common resources. In: ACSAC '04: Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04), Washington, DC, USA, IEEE Computer Society (2004) 360–369
14. Templeton, S.J., Levitt, K.: A requires/provides model for computer attacks. In: NSPW '00: Proceedings of the 2000 workshop on New security paradigms, New York, NY, USA, ACM (2000) 31–38

15. Zhou, J., Heckman, M., Reynolds, B., Carlson, A., Bishop, M.: Modeling network intrusion detection alerts for correlation. ACM Trans. Inf. System Secur. **10**(1) (2007) 4
16. Pouget, Fabien, Dacier, M.: Alert correlation: Review of the state of the art. Technical Report EURECOM+1271, Institut Eurecom, France (Dec 2003)
17. Manganaris, S., Christensen, M., Zerkle, D., Hermiz, K.: A data mining analysis of rtid alarms. Computer Networks **34**(4) (2000) 571–577
18. Michel, C., Mé, L.: Adele: an attack description language for knowledge-based intrustion detection. In: Sec '01: Proceedings of the 16th international conference on Information security: Trusted information. (2001) 353–368
19. Cuppens, F., Miège, A.: Alert correlation in a cooperative intrusion detection framework. In: SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy, Washington, DC, USA, IEEE Computer Society (2002) 202
20. Siraj, A., Vaughn, R.B.: Alert correlation with abstract incident modeling in a multi-sensor environment. IJCSNS International Journal of Computer Science and Network Security **7**(8) (August 2007) 8–19
21. Morin, B., Mé, L., Debar, H.: Correlation of intrusion symptoms: An application of chronicles. In: RAID'03: Proc. of the 6th Int. Symp. on Recent Advances in Intrusion Detection, Springer Berlin / Heidelberg (Sep 2003) 94–112
22. Vigna, G., Valeur, F., Kemmerer, R.A.: Designing and implementing a family of intrusion detection systems. In: ESEC/FSE-11: Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering, New York, NY, USA, ACM (2003) 88–97
23. Yang, D., Chen, G., Wang, H., Liao, X.: Learning vector quantization neural network method for network intrusion detection. Wuhan University Journal of Natural Sciences **12**(1) (Jan 2007) 147–150
24. Mehdi, M., Zair, S., Anou, A., Bensebti, M.: A bayesian networks in intrusion detection systems. Journal of Computer Science **3**(5) (May 2007) 259–265
25. Morin, B., Mé, L., Debar, H., Ducassé, M.: M2d2: A formal data model for ids alert correlation. In: RAID'02: Proc. of the 5th Int. Symp. on Recent Advances in Intrusion Detection, Springer Berlin / Heidelberg (Oct 2002) 115–137
26. Ning, P., Cui, Y., Reeves, D.S., Xu, D.: Techniques and tools for analyzing intrusion alerts. ACM Trans. Inf. Syst. Secur. **7**(2) (2004) 274–318
27. Li, N., Wang, Q.: Beyond separation of duty: an algebra for specifying high-level security policies. In: CCS '06: Proceedings of the 13th ACM conference on Computer and communications security, New York, NY, USA, ACM (2006) 356–369
28. Wijesekera, D., Jajodia, S.: A propositional policy algebra for access control. ACM Trans. Inf. Syst. Secur. **6**(2) (2003) 286–325
29. Bonatti, P., di Vimercati, S.D.C., Samarati, P.: An algebra for composing access control policies. ACM Trans. Inf. Syst. Secur. **5**(1) (2002) 1–35