Intrusion Alert Prioritisation and Attack Detection using Post-Correlation Analysis

Riyanat Shittu[a,], Alex Healing[b], Robert Ghanea-Hercock[b], Robin Bloomfield[a], Rajarajan Muttukrishnan[a]

[a]*School of Engineering and Mathematical Science, City University London, Northampton Square EC1V 0HB, UK*
[b]*Security Futures Practice, Research & Innovation, British Telecom, Ipswich, IP5 3RE, UK*

Abstract

Event Correlation used to be a widely used technique for interpreting alert logs and discovering network attacks. However, due to the scale and complexity of today's networks and attacks, alert logs produced by these modern networks are much larger in volume and difficult to analyse. In this research we show that adding post-correlation methods can be used alongside correlation to significantly improve the analysis of alert logs.

We proposed a new framework titled *A Comprehensive System for Analysing Intrusion Alerts (ACSAnIA)*. The post-correlation methods include a new prioritisation metric based on anomaly detection and a novel approach to clustering events using correlation knowledge. One of the key benefits of the framework is that it significantly reduces false-positive alerts and it adds contextual information to true-positive alerts.

We evaluated the post-correlation methods of ACSAnIA using data from a 2012 cyber range experiment carried out by industrial partners of the British Telecom SATURN programme. In one scenario, our results show that false-positives were successfully reduced by 97% and in another scenario, 16%. It also showed that clustering correlated alerts aided in attack detection.

The proposed framework is also being developed and integrated into a pre-existing Visual Analytic tool developed by the British Telecom SATURN Research Team for the analysis of cyber security data.

1. Introduction

A 2013 study showed that 84% of attacked organisations had evidence of the attack in their event log files (Verizon, 2013). This demonstrates that the task

Email addresses: riyanat.shittu.1@city.ac.uk (Riyanat Shittu), alex.healing@bt.com (Alex Healing), robert.ghanea-hercock@bt.com (Robert Ghanea-Hercock), reb@csr.city.ac.uk (Robin Bloomfield), rajarajan.muttukrishnan@city.ac.uk (Rajarajan Muttukrishnan)

of successfully analysing event logs for the purpose of attack detection is non-trivial. Event Correlation is a process used to detect attacks by finding multiple network events and activities with similar properties. Take for instance, in intrusion detection, an alert indicating the exploit of a known vulnerability on a host can be correlated with the host's list of vulnerabilities. This may prove useful in validating the intrusion alert as threatening or non-threatening.

Most widely used security event analysis tools such as IBM's Intelligent Operation Center (IBM Corporation, 2013) and AlienVault's USM (Alienvault, 2013) apply event correlation.

In our proposed framework - ACSAnIA, event correlation is the primary component and it is targeted at the analysis of Intrusion Detection System (IDS) alerts. The reason is that IDSs - particularly signature based IDSs - are known to generate a vast amount of low-level alerts with a high percentage being false positives. Using event correlation, IDS alerts can be grouped into high-level structures called meta-alerts. Meta-alerts make it easier to identify alerts that are interesting. Furthermore, meta-alerts significantly reduces the amount of data required to be assessed by a security analyst.

Post-correlation processes such as *"Prioritisation Analysis"* are used to analyse alerts after correlation has been performed (Salah et al., 2013). Prioritisation assigns a level of importance to each meta-alert. This aids a response system in determining the order IDS alerts should be addressed. In general, our observation from literature is that a small amount of research has focussed on improving post-correlation methods.

1.1. Contributions

In ACSAnIA, we focus on improving attack detection and enhancing the output of the correlation by performing post-analysis. The main contributions presented in this paper are:

1. A new metric for prioritising alerts based on anomalous behaviour.
2. A new method for applying clustering on correlated alerts.
3. An improved data structure for representing robust attack patterns.

The first contribution is the prioritisation metric. Our hypothesis is that a set of correlated alerts should be prioritised if it represents anomalous network behaviour at a given time. The prioritisation metric is based on our previous work (Shittu et al., 2014). In our new work we have improved the correlation phase. It uses the LOF (Local Outlier Factor) algorithm (Breunig et al., 2000) to assign an outlier value between 1 and 4 to each group of correlated alerts. In this research, each group of correlated alerts is referred to as a meta-alert. Meta-alerts with higher outlier values indicate higher anomalies thus higher prioritisations.

The second contribution is that this research is the first to apply clustering to meta-alerts. In prior art, cluster analysis is often applied to low-level IDS alerts. One of the drawbacks of this is that low-level IDS alert clustering only discovers alerts with similar properties and is not suitable for discovering progressive attacks which involve many alerts. In our work, a meta-alert can be seen as an overview of an attack with potentially multiple steps. Clustering similar meta-alerts into groups allow the discovery of similar attack patterns. In our work, we apply a density based clustering method, DBSCAN (Ester et al., 1996) which finds arbitrary sized and shaped clusters within a given density scope.

The final contribution is that through investigating methodologies for understanding attacks, we propose a new approach to extracting attack patterns from meta-alerts. Attack patterns are properties observed accross two or more meta-alerts that capture part of an attackers intention. Frequent pattern mining is applied to meta-alerts for pattern extraction using a graph mining algorithm, GSPAN (Yan, 2002).

To the authors' knowledge, the contributions listed are novel and timely. In Contribution 1, others have proposed prioritisation metrics but not based on anomaly analysis. Our results show that anomaly behaviour is a suitable approach for prioritisation IF outlier alerts exist in the dataset. In one of the datasets which had distinct outliers we significantly reduced the false positives by upto 99.7%. In another scenario with lesser distinct outliers the false positives were reduced by 16%. For Contribution 2, it is acknowledged that Patel (2009) first proposed clustering meta-alerts however we improve on this by using more robust data structures and clustering approaches. To achieve this we adapt data structures from graph based analysis. For Contribution 3, to our knowledge, is an entirely new area of investigation used to detect new types of attack patterns based on meta-alerts.

2. Related Work

The related work in this section is taken from three key areas - alert prioritisation, correlation and attack pattern extraction.

To the knowledge of the authors, very little work has been done on defining IDS alert prioritisation metrics. Porras et al. (2002) first proposed an alert ranking framework, M-Correlator, with a prioritisation component that consisted of two security metrics: relevance and the priority scoring. Relevance scoring measured the validity of an alert while priority scoring measured the severity of an alert given the targeted asset's value. The priority score also combined an interest score which measured the degree to which an analyst expressed interest in the attack category the alert belonged. Using a Bayesian model they determine the overall priority of an alert based on the acquired evidence. A limitation in their approach is that knowledge from alert correlation is not taken into account during the prioritisation despite their framework consisting of a similarity based correlation component. Since it is solely based on user and network knowledge the framework is limited to discovering known incidents while novel attack incidents remain unprioritised.

Noel and Jajodia (2007) proposed an alert prioritising framework which used a different metric. The metric calculated the proximity of an alert to a critical asset. Thus, alerts targeted at assets closer to critical assets had a higher priority over those further away. Similarly, in Porras et al's framework, it only used network knowledge and no alert or correlation context was taken into consideration. A more robust alert prioritisation system is proposed by Alsubhi et al. (2008, 2012) who define 7 metrics for prioritising alerts. One of the metrics, called *alert relationship metric* is relevant to our work. The alert relationship metric measures the degree to which the alert correlates with successive alerts. Using this prioritisation metric a high value could indicate the alert is potentially a causal alert. Our proposed outlier metric differs in two main ways. Firstly, we assign prioritisation at the meta-alert level rather than the alert level. Secondly, we not only measure the causality of alerts but furthermore, the unusualness of the causality.

Zomlot et al. (2011) also proposed a prioritisation model for the alert correlation system they had previously presented (Sundaramurthy et al., 2011). In their work on prioritisation, they use Dempster-Shafer to assign a degree of belief to each meta-alert (generated by the correlation system) which indicated the likelihood of true positivity given the quality of the IDS sensor which raised the alerts. Unlike alert prioritisation, more effort has been focussed on alert correlation techniques. Using Salah et al's correlation model taxonomy, these can be classified into case-based, similarity-based and sequential-based methods. Case-based methods involve a rule language that uses expert domain knowledge to define alert types that may occur in a given attack scenario.Cuppens and Ortalo (2000); Cheung et al. (2003); Steven Eckmann (2002); Cédric Michel (2001) proposed LAMBDA, CAML, STATL, and ADELE respectively. More recent work in case-based models include work by Zali et al. (2013) and Alireza Sadighian (2013). Although these provide high-quality correlations capturing known attacks, their limitation is that they are difficult to implement and maintain on large-scale complex networks. In similarity methods, the correlation is based on feature similarity. Valdes and Skinner (2001) as well as Dain and Cunningham (2001) first introduced this approach. Although simpler to implement, such methods do not capture complex nor hidden correlations. More Recently, Hofmann and Sick (2011) and Chen et al. (2014) both proposed improved similarity techniques using feature selection and probablistic and real-time clustering techniques. Sequential-based correlation is more suitable for capturing causally correlated alerts with little or no apriori knowledge. Sequential-based methods include those proposed by Ning et al. (2001) and Debar and Wespi (2001). Both used rule-like pre-requisites and consequences for correlating alerts. Qin (2005) also used abstract pre-requisites and consequences combined with statistical evaluation for correlating alerts. Sequential-based alert correlation models which use little to no a priori knowledge are based on Bayesian inference. Examples include work by Ahmadinejad and Jalili (2009); Ren et al. (2010); Marchetti et al. (2011) and Benferhat et al. (2013).

Each of these alert correlation models output a set of correlated alerts represented in a graph like structure known as an *Alert Correlation Graph (ACG)*. Ning and Xu (2003) addresses how ACGs can be made useful to an analyst by simplifying the graphs using node and edge reduction techniques. However, few have focussed on how to prioritise alert correlation graphs in the event where many ACGs are produced. Based on our experiments with alert correlation models this case is typical in real environments. In environments where a vast amount of alerts are produced, it is likely to generate a similarly vast amount of alert correlation graphs. Our work aims to address this challenge by introducing post-correlation components to make analysing the graphs a more tractable problem for the human analyst.

In the study of attack pattern extraction and recognition many apply frequent pattern mining algorithms. Khan et al. (2010) developed two probabilistic models to integrate with a prior existing pattern mining algorithm, Prefix-Span for mining frequent structures in Subgraphs. In their work they test the algorithm on intrusion alert traffic and discover meaningful attack patterns. Sadoddin and Ghorbani (2009) also applied FP-Growth, a frequent pattern mining technique for discovering graph attack patterns. Other work using pattern mining algorithms include work by Lagzian (2012).

One of the main challenges with applying frequent pattern mining on ACGs is that ACGs are multi-attributed weighted graphs therefore most of the methods cannot be used to mine such complex structures. In this research we investigate attack pattern structures and how frequent pattern mining can be applied on graph structures without the need for over-simplifying the data structures which may cause information loss.

3. Background on Alert Analysis

3.1. Low-level Alert

Figure 1 corresponds to a log entry of a single intrusion alert triggered by a Snort IDS. *Line 1* identifies the *type*, *classification* and the *default priority* of the intrusion alert. *Line 2* logs the *time* the intrusion was detected and the *IP addresses* and *ports* of the sender and recipient of the packet which triggered the alert. In addition, *lines 3 & 4* log details of the packet such as the packet's Time to Live(TTL), network protocol, Type of Service(TOS), length and other packet header details.

```
ICMP-INFO PING -> Misc activity -> Priority 3
03/07-15:45:37.137344 -> 172.16.113.84 -> 135.13.216.191
ICMP TTL:255 TOS:0x0 ID:1332 ipLen:20 DgmLen:38
Type:8 Code:0 ID:2049 Seq:5632 ECHO
```

Figure 1: Snort IDS Alert Example

The ACSAnIA system uses six of the intrusion alert's attributes which are logged. Each ACSAnIA alert is represented as a 6-tuple $(\alpha_1, \alpha_2, ..., \alpha_6)$ where the elements of the tuple are the attribute values of the alert's timestamp, source IP, source port, destination IP, destination Port, and intrusion type respectively. These attributes are either text, IP address or of numeric data types. An alert of type T_a is an alert instance which has T_a as the value for α_6.

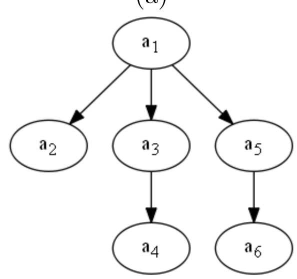
3.2. *Meta-alert and Alert Correlation Graph*

A meta-alert is a higher-level alert which contains one or more low-level alerts (e.g. Snort alerts) grouped together by an aggregation or correlation system. Figure 2(a) shows a set of intrusion alert tuples labelled a_1 to a_6. After some form of correlation, a logical relation such as Figure 2(b) is established and is referred to as an "Alert Correlation Graph".

In prior research, a meta-alert has been referred to as a hyper-alert, hyper-alert graph, alert correlation graph and attack graph. To ensure clarity, only the terms meta-alert and alert correlation graph are used in this paper. In general, the term meta-alert is used when describing correlated alerts. As a more technical definition, particularly when referencing the "graph" data structure of the correlation alerts depicted in Figure 2(b), the term "Alert Correlation Graph" is used.

a_1: (03/07-16:28, 1.1.1.40,26582, 5.5.5.3, 25, Policy attempted download of a pdf)
a_2: (03/07-16:28, 5.5.5.3, 3727, 10.0.0.3, 25, Policy attempted download of a pdf)
a_3: (03/07-17:30, 1.1.1.43,48097, 5.5.5.3, 25, Policy attempted download of a pdf)
a_4: (03/07-17:30, 5.5.5.3, 3730, 10.0.0.3, 25, Policy attempted download of a pdf)
a_5: (03/07-17:35, 1.1.1.48,53514, 5.5.5.3, 25, Policy attempted download of a pdf)
a_6: (03/07-17:35, 5.5.5.3, 3727, 10.0.0.3, 25, Policy attempted download of a pdf)

(a)



(b)

Figure 2: (a)A set of alert tuples -(timestamp, source IP, source port, destination IP, destination Port, and intrusion type) before correlation and (b) A meta-alert/alert correlation graph

An alert correlation graph is a weighted directed acyclic connected graph G = (V, E) where V represents a set of nodes and each node v $\in$ V represents an 6-tuple low-level alert. Each edge, $e_{v_i,v_j} \in$ E is a connection between two nodes v_i, v_j which indicate that 1) v_i and v_j are correlated and 2) v_i represents an alert that occurred before v_j. The weight of the edge depicts the correlation strength between both nodes.

4. Architecture of the Proposed System

The ACSAnIA framework consists of seven components: (1) Offline Correlation (2)Online Correlation (3) Meta-alert Comparison (4) Meta-alert Prioritisation (5) Meta-alert Clustering (6) Attack Pattern Discovery and (7) Reporting

System. A knowledge repository and database is also used to persist alert information and discovered knowledge. Figure 3 shows the architecture of ACSAnIA.

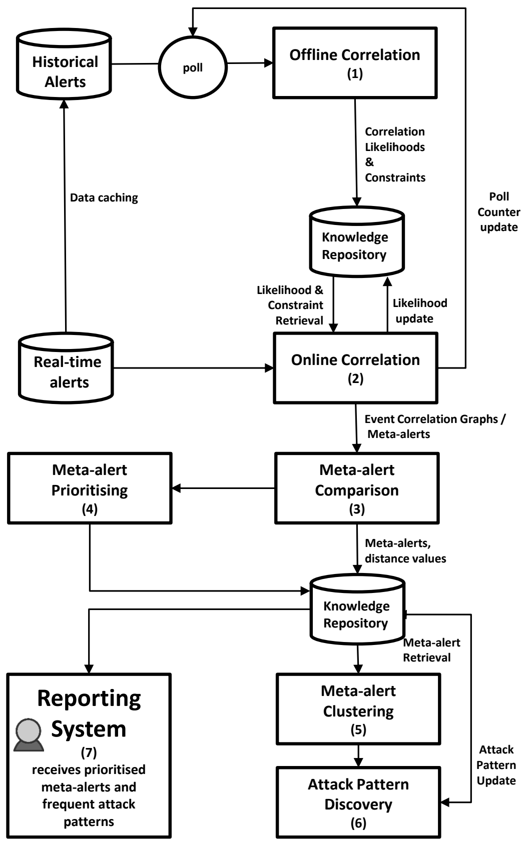


Figure 3: Architecture of the ACSAnIA Framework

4.1. Offline Correlation

ACSAnIA uses a set of historic alerts to build a correlation model. The correlation model is built by the offline correlation component and is periodically used and updated by the online correlation component. The correlation model consists of two knowledge tables: (i) Correlation Likelihood Table and (ii) Correlation Constraint Table. A correlation likelihood represents the strength between two alert types T_a and T_b. More specifically, it refers to the *likelihood* of an alert of type T_b occurring after an alert of type T_a. For any two alert types, T_a and T_b, the correlation likelihood $L(T_a, T_b)$ is conditional to a constraint being true. The relationship between correlation likelihood and constraint is represented in Eq.1.

$$L(T_a, T_b) = P(T_a \rightarrow T_b | C) \tag{1}$$

A constraint, C, is a rule which captures the conditions under which two alerts are correlated. For example a constraint such as $\text{time}_{a,b} \leq 20\text{secs}$ indicates that both alert types T_a and T_b are correlated when they occur within 20 seconds of each other. Another illustration of a constraint is $\{\text{destIP}_{a,b} = 1\}$, this indicates that both alert type T_a and T_b are correlated when they share the same destination IP (In other words, the difference between their destination IP addresses is zero). Table 1 illustrates further examples of constraints encountered.

Constraints	Descriptions
$\{\text{DestPort}_{a,b} =1 \}$	The destination port of alert of T_a and an alert of T_b must be identical
$\{\text{DestIP}_{a,b} \geq=0.5\}$	The destination IP of alert of T_a and an alert of T_b must be common up to at least the 2nd Octet.
$\{\text{DestIP}_{a,b} \geq 0.5$, $\text{SourceIP}_{a,b} \geq 0.25$, $\text{DestPort}_{a,b} = 1 \}$	The destination IP of alert of T_a and an alert of T_b must be common up to at least the 2nd Octet, their source IPs must be common up to at least the 1st Octet and their destination port must be identical.

Table 1: Examples of Constraints between Alert Types

When two alert types have $n : n > 1$ constraints between them, the correlation likelihood between the two alert types is the minimum likelihood of the n constraints:

$$L(T_a, T_b) = \min\{P(T_a \rightarrow T_b|C_i)\}_{i=1}^{n} \tag{2}$$

where the probability of $T_a \rightarrow T_b$ occurring given C_i is defined as:

$$P(T_a \rightarrow T_b|C_i) = \frac{P(C_i) * P(C_i|T_a \rightarrow T_b)}{P(T_a \rightarrow T_b)} \tag{3}$$

In Eq.3, given a set of historical alerts, $P(T_a \rightarrow T_b)$ refers to the number of times T_b occurs after T_a in the same time window W with respect to the number of times T_a occurs in that same window. $P(C_i)$ refers to the number of times an alert of Type T_b occurs after an alert of T_a where both types satisfy constraint C_i with respect to the number or times T_a occurs in the total historial alert dataset H. Finally, $P(C_i|T_a \rightarrow T_b)$ is the probability of C_i given both Type T_a and T_b occur within the same time window.

Algorithm 1 shows how the offline correlation component computes all correlation likelihoods and constraints.

Algorithm 1 Offline Correlation Process

1: **function** OFFLINE PROCESS
2: A = All alert attributes
3: H = Historic Alerts
4: T = All alert types in H
5: T' = All pairs of types in T
6: **for all** $T_a, T_b \in T'$ **do**
7: $C(T_a, T_b) =$
8: GETCONSTRAINTS(A, T_a, T_b)
9: $L(T_a, T_b) =$
10: $\min\{$
11: $P(T_a \rightarrow T_b | C(T_a, T_b)_i)$
12: $\}_{i=1}^{n}$
13: **end for**
14: **end function**

1: **function** GETCONSTRAINTS(C, T_a, T_b)
2: $k = 1$
3: $C(T_a, T_b) \leftarrow \emptyset$
4: Get first order feature set
5: **for all** $c_i \in C$ **do**
6: **if** $P(T_a \rightarrow T_b | c_i) > \theta$ **then**
7: $C(T_a, T_b) \leftarrow C(T_a, T_b)_k \cup c_i$
8: **end if**
9: **end for**
10: Gets k relevant feature sets
11: $k \leftarrow 2$
12: $C \leftarrow \emptyset$
13: **while** $(C(T_a, T_b)_{k-1} \neq \emptyset)$ **do**
14: $C \leftarrow$ All k combinations from
15: $C(T_a, T_b)_{k-1}$
16: **for all** $c_i \in C$ **do**
17: **if** $P(T_a \rightarrow T_b | c_i) > \theta$ **then**
18: $C(T_a, T_b)_k \leftarrow C(T_a, T_b)_k \cup c_i$
19: **end if**
20: **end for**
21: $k + 1$
22: **end while**
23: **return** $C(T_a, T_b)$
24: **end function**

Lines 1 to 5 describe the dataset required to initialise the offline process. A is the set of alert fields used by ACSANIA i.e. *(timestamp, source IP, source port, destination IP, destination Port, and intrusion type)*, H is a set of historical alerts used to train the model which are retrieved from log files or an alert database, T is a finite set of all values possible for field *type* and T' is a set containing all 2-permutations of the set T where T'_i represents the i^{th} T_a, T_b pair.

For each pair, the function GETCONSTRAINTS generates a set of constraints by computing all possible k-combinations of the attributes in A using an step-wise apriori approach. Firstly, we start with the $k-$combination where $k = 1$. This means we generate constraints of length 1, where each constraint, C only contains one attribute $a \in A$. For each constraint, we measure the probability that T_a will occur before T_b given they have the constraint C in common. If the probability does not exceed a given threshold θ, it is pruned and considered as non-relevant to the pair T_a, T_b. At the end of each incremental stage, non-pruned constraints are used to generate $K + 1$ combinations : $k <= |A|$. This is how the Constraint 3 in Table 1 is generated.

4.2. Online Correlation

Each incoming alert a_j, received in real-time is analysed against a set of alerts $S = \{a_1, a_2, ..., a_n\}$ that had occurred within the last T_θ seconds before alert a_j. To determine if a_j and an alert in S are correlated, their types are extracted and used to find the relevant correlation likelihood and constraints (stored in the knowledge repository by the offline component). Two alerts are correlated if:

1. The correlation likelihood of Type a_i and Type a_j is greater than or equal to a threshold, C_θ

2. At least one of the respective constraints of Type a_i and Type a_j holds true for a_i and a_j.

Each analysed historic alert is stored as a node in the database. If the incoming alert a_j is correlated with an existing alert a_i, a_j is added to the meta-alert which a_i belongs to. Consequently, an edge is added to the meta-alert to depict the correlation.

4.3. Meta-alert Comparison

The correlation process typically produces multiple meta-alerts. Under intense traffic analysis, hundreds of meta-alerts may be produced within a short time frame. In order to make sense of the generated meta-alerts, a quantitative approach is used to measure the differences between each meta-alert. This is subsequently used by the Meta-alert Prioritising and Meta-alert Clustering Components.

The importance of such analysis is explained in this real example that was observed in the analysis of one of our networks. Two arbitrary meta-alerts, m_1, m_2 where pulled out from a large set of generated meta-alerts. m_1 consists of two connected nodes $(v_{1_{m_1}}, v_{2_{m_1}})$ where $v_{1_{m_1}}$ represented a suspicious ping from a mail server to a web server and the $v_{1_{m_1}}$ represented the response from the web server to the mail server (the reply also triggered an intrusion). m_2 also consisted of two connected nodes however different context. $v_{1_{m_2}}$ represented an intrusion triggered on a packet from an external address routed to our networks DMZ mail server, the DMZ server then routed the packet to our internal mail server. This also triggered an alert which is represented by $v_{2_{m_2}}$. Despite the similarity in structure, visual analysis enables a security analyst to understand the attack patterns and their context (and that they are not the same type of intrusion). In addition, it was observed that many of the generated meta-alerts were similar to either but not both. Given thousands of meta-alerts detecting such attack patterns may prove infeasible to perform visually. Clustering analysis however could potentially solve this challenge.

As previously described in Section 3, the alert correlation graph (ACG) is the data structure of a meta-alert. Graph Edit Distance (GED) is used as the distance metric to compute the quantitative differences between alert correlation graphs (Shapiro and Haralick, 1981).

For any two alert correlation graphs, acg_1 and acg_2, the difference between them is the GED which is calculated by counting the minimal number of actions required to transform acg_1 into acg_2 by manipulating acg_1 using a number of operations such as node deletion/insertion, edge deletion/insertion and node/edge substitution. Algorithm 2 details the Meta-alert Comparison for any two meta-alerts. The distance values between all pairs of meta-alerts is computed and persisted in the knowledge repository as illustrated in Figure 3.

Algorithm 2 Edit Distance between meta-alerts (Tekhov, 2009)

```
function EditDistance('acg1, 'acg2)
    L ← maximum cost allowed
    Q ← ∅                        ▷ A queue sorted by
        minimum path cost
    vi ← random vertex from 'acg1

    for vj in 'acg2 do
        s = new substitutePath(vi,vj)
        Q ← Q ∪ s
    end for
    d ← new deletePath(vi)
    Q ← Q ∪ d
    while true do
        e = Q.firstPath()
        if e.isComplete() then
            return e
        end if
        if e.cost() > L then       ▷ The maximum
cost has been exceeded
            return L
        end if
        Extend(e, acg1, 'acg2, Q)
    end while
end function
```

```
function Extend(e, acg1, acg2, Q)
    if acg1(V) ⊆ e then
        vi = next vertex in acg1 : vi ∉ e
        for vj ∈ acg2 do
            s ← e ∪ new substitutePath(vi,vj)
            Q ← Q ∪ s;
        end for
        d ← e ∪ new deletePath(vi)
        Q ← Q ∪ d
    else
        for vi ∈ acg2 do
            if vi ∉ e then
                i ← e ∪ new insertPath(vi)
            end if
        end for
    end if
end function
```

In Algorithm 2, there are three key operations - *substitute path, delete path* and *insert path.* Each operation requires a cost function which is used to sort them in the queue. Using domain knowledge, a set of cost functions suitable for meta-alert comparison are defined.

Insert and Delete Path $C(N_I), C(N_D)$. When comparing any two ACGs, if one graph has more nodes (i.e. low-level alerts) than the other, a set of node insertions or deletions may be used to transform one graph into the other. For each intrusion type, we define a weight $w_{N(T)}$, In our experiments, all weights were defaulted to 1.

For edges, the cost of edge insertion/deletion is equivalent to the weight of the edge.

Substitute Path. The substitution of a node replaces an alert for another. Therefore the more similar the alerts, the less the cost of substitution. We measure the similarity between the alerts using the Euclidean metric in eq.4.

$$C(N_S) = \mathrm{d}(\mathrm{v_i}, \mathrm{v_j}) = \sqrt{\sum_{k=1}^{n} (v_i(\alpha_k) - v_j(\alpha_k))^2} \tag{4}$$

Note that $v_i(\alpha_k)$ is the k^{th} attribute of alert v_i and n is the total number of attributes. For categorical attributes, we use string edit distance and for IP attributes we use a common prefix metho. An example of the difference between two IP addresses is shown in Table 2.

In our work we also define a cost for edge substitution. The cost of edge substitution is the absolute value of the differences between the weights of both edges eq. 5.

Table 2: IP Similarity

172.16.113.20	10101100 . 00010000 . 01110001 . 11001111
172.16.115.20	10101100 . 00010000 . 01110011 . 00010100
Common Mask	11111111 . 11111111 . 11111100 . 00000000
	22/32 = 0.68

$$C(E_D) = |W_{E_s} - W_{E_d}| \tag{5}$$

4.4. Meta-alert Prioritisation

The meta-alert prioritisation component assigns a priority level to each meta-alert based on its dissimilarity to a set of other meta-alerts. There are four priority levels, meta-alerts which are highly similar to others are typically associated with a Priority 1 or 2 while highly dissimilar meta-alerts are assigned level 3 or 4.

Each meta-alert is mapped to a prioritising value based on the degree to which it is an outlier. The degree to which a meta-alert is an outlier is calculated using the Local Outlier Factor (LOF), of a point Breunig et al. (2000). The mapping between priority values and the LOF of a meta-alert is illustrated:

$$p(g) = \begin{cases} 1 & 0.00 \leq \text{nLOF(g)} \leq 0.25 \\ 2 & 0.25 < \text{nLOF(g)} \leq 0.50 \\ 3 & 0.50 < \text{nLOF(g)} \leq 0.75 \\ 4 & 0.75 < \text{nLOF(g)} \leq 1.00 \end{cases} \tag{6}$$

In equation 6, g is a meta-alert and nLOF is the weighted LOF value. nLOF is calculated over five steps using a point's (meta-alert's) neighbourhood, reachability distance and local density.

1. k-neighbourhood & k-distance: A k-neighbourhood of a meta-alert g_i, denoted as $N_k(g_i)$, is a set of other meta-alerts in which the difference between any of the other meta-alerts and g_i is less than or equal to the k-distance. The k-distance of a g_i is the distance between g_i and the k^{th} nearest meta-alert. k is a configurable parameter provided for the algorithm's computation.
2. reachability distance: This is the maximum between the distance between two meta-alerts and the latter meta-alert's k-distance.

$$rd_k(\text{g,g}_j) = \max\{D(g, g_j), k\text{-}distance(g_j)\} \tag{7}$$

3. Local reachability density: A meta-alert's local reachability density is the inverse of the average reachability distance between it and its k-

neighbourhood.

$$\mathrm{lrd}_{g_i} := \left(\frac{\sum\limits_{g_j \in N_k(g_i)} rd_k(g_i, g_j)}{|N(g_i)|} \right)^{-1} \tag{8}$$

4. Local Outlier Factor: For each meta-alert, g, its LOF degree is calculated:

$$LOF_{\mathrm{g}_i} = \frac{\sum\limits_{g_j \in N_k(g_i)} \frac{lrd_{g_i}}{lrd_{g_j}}}{|N_{g_i}|} \tag{9}$$

5. LOF Priority: Since the value LOF could range between 0 and ∞, we use a weighted technique to map the LOF to a value between 0 and 1.

$$nLOF(g) = \frac{LOF(g)}{\max\{LOF(g_i)\}_{i=0}^{|G|}} \tag{10}$$

Following the computation of the prioritisation value of each meta-alert, the meta-alert prioritisation component uses a filtering subcomponent which forwards all alerts with a threshold greater than P_θ to a reporting system for further investigation.

4.5. Meta-alert Clustering

The meta-alert clustering component receives the set of meta-alerts, G and groups them into clusters if $|G|$ is greater than two. A given meta-alert, g is considered to belong to cluster C_i if it is density reachable by an inlier member of cluster C_i. An inlier of a cluster is any meta-alert, $g_i \in C_i$ that has at least k other meta-alerts which is similar to it(i.e. the difference between g_i and g_j: $g_j \in C_i$ should be less than a threshold referred to as ϵ). The clustering process uses the DBSCAN algorithm and is illustrated in Algorithm 4.

4.6. Attack Pattern Discovery

The attack pattern discovery component receives the clusters of meta-alerts and attempts to extract a set of representative features for each cluster using frequent pattern mining. Prior to pattern mining, the attack pattern discovery component represents each meta-alert as a less complex graph structure. This graph structure is referred to as a pattern graph. An attack pattern graph is a graph representation of a meta-alert where each node either represents an alert type or attribute of an alert type and each edge either represents the correlation between two alert types or the association of an attribute to an alert type.

Figure 4 illustrates the mapping from a meta-alert graph to a graph pattern. The node labelled "HTTP IE Security..." in Figure 4(a) represents an alert. Each alert is represented as multi-dimensional vector. Because graph pattern

Algorithm 4 DBSCAN algorithmic procedure (Ester et al., 1996)

```
function ClusterGraphs(G, k, ϵ)
    i = 0 //the ith cluster
    C = ∅ //A set of all clusters
    for all g ∈ G do
        if (state(g) == unvisited) then
            state(g) ← visited
            N(g) = getNeighbors(g, ϵ)
            if (sizeOf(N(g) < k) then
                category(g) ← NOISE
            else
                C_i ← ∅
                growCluster(g, N(g), C_i, ϵ, k)
            end if
        end if
    end for
end function
function growCluster(g, N(g), C_i, ϵ, k)
    C_i ←C_i∪ g
    for all g' ∈ N(g) do
        if (state(g') is unvisited) then
            (g') ← visited
            N(g') ← getNeighbors(g', ϵ)
        end if
        if (sizeOf(N(g')) geq k) then
            N(g) ← N(g) ∪ neighbors(g')
        end if
        if (category(g') is NULL) then
            Category(g') ← i
            C_i ← C_i∪ g'
        end if
    end for
end function
```

mining is not suitable for multi-attribute nodes, each node is flattened into a single attributed labelled node. This is done by representing each attribute of an alert as a new node. In order to maintain all attributes and by associating each attribute node to the alert type node using an edge as illustrated in Figure 4 (b). Attribute nodes are distinguished from alert type nodes by using different shapes. A graph frequent pattern mining algorithm, GSPAN is used to extract

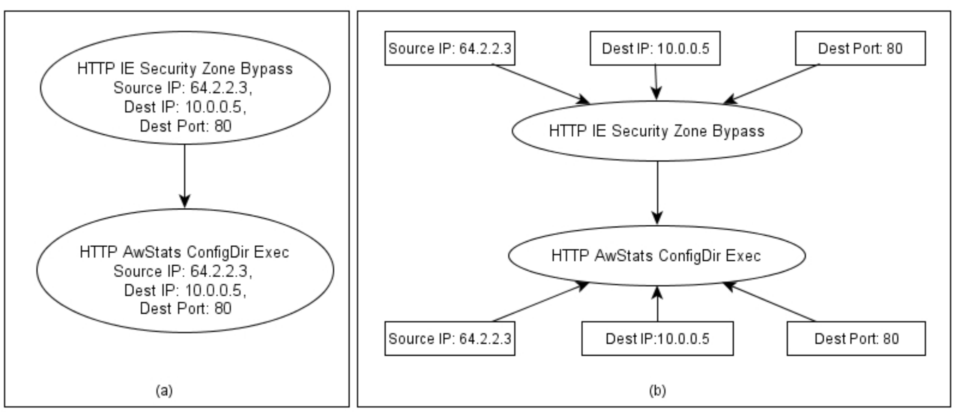


Figure 4: (a) An Alert Correlation Graph (b) Attack Pattern Graph of the Alert Correlation Graph

frequent patterns from each cluster. Given a set of graphs and a minimum support threshold value S_θ, a set of frequent patterns is extracted. Each frequent pattern extracted from any cluster C_i is a subgraph common to at least S_θ meta-alerts in the cluster. The GSPAN algorithm is too long and detailed to describe in this paper. We refer interested readers to the original paper (Yan, 2002).

4.7. Reporting System

The reporting system is a visual analytic web application and is integrated into the “Saturn Assure Analytics” tool-kit developed at British Telecom’s Security Research Labs. This tool-kit is described in detail by (Rowlingson et al., 2013).

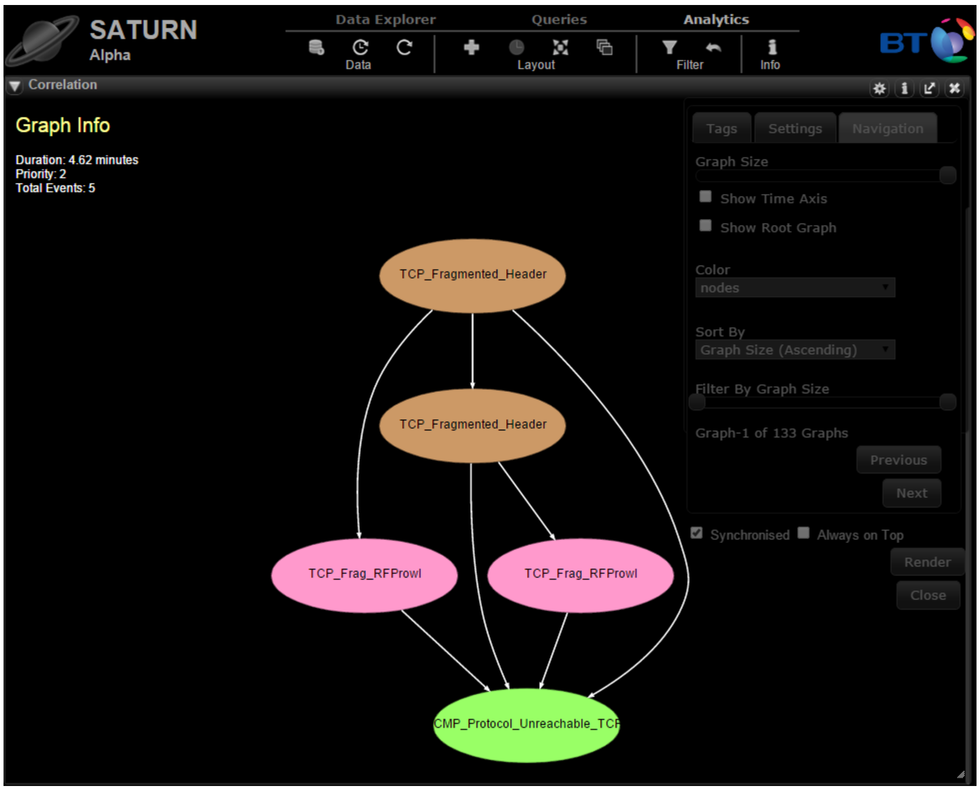


Figure 5: A snippet of the ACSANIA reporting system dashboard.

Figure 5 shows a snippet of the dashboard. The interface enables an analyst to explore each prioritised meta-alert extensively through a range of interactive actions. For example he or she may sort and filter meta-alerts based on their size, priority level, recency and more. The view displays a summary of the meta-alert (top-left) and to navigate between meta-alerts, the UI consists of a navigation tab (far right). In general, the key benefit of the reporting system is that it aids an analyst in understanding and detecting attacks from analysing alerts by 1) significantly reducing the content to be investigated and 2) adding contextual information to each low-level alert.

5. Evaluation Metrics

The ACSAnIA system is evaluated by using quantitative measures to measure the quality of the prioritisation and clustering components.

5.1. Alert Prioritisation Quality

The quality of the prioritised meta-alerts is measured using the sensitivity metric typically known as the true positive rate (TPR). The TPR evaluates the ability of the system to correctly prioritise the right meta-alerts. It is defined as follows:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\text{\# of correctly prioritised alerts}}{\text{\# of true positive alerts}} \tag{11}$$

The false positive rate (FPR) is also measured as the system's ability to ensure that unimportant meta-alerts are not prioritised.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\text{\# of incorrectly prioritised alerts}}{\text{\# of prioritised alerts}} \tag{12}$$

5.2. Cluster Quality

Intuitively a set of well clustered points are those where there is a high intra-similarity between the members of each cluster but a low inter-similarity between the various clusters. To measure this quality, the silhouette coefficient of each cluster is measured. The silhouette coefficient is defined as:

$$SC(C_i) = \left(\frac{\sum_{k=1}^{|C_i|} b_k - a_k}{\max\{a_k, b_k\}} \right) / |C_i| \tag{13}$$

Such that a_k and b_k are the mean intra-cluster similarity and inter-cluster similarity of the k^{th} member of Cluster C_i respectively. For each member, g of a cluster C_i, a and b are defined as follows:

$$a = \frac{\sum_{g_k \in C_i} D(g, g_k)}{|C_i|} \text{ and } b = \frac{\sum_{g_k \in C_i^{\complement}} D(g, g_k)}{|C_i^{\complement}|}$$

Note that $C_i^{\complement}$ is the set of graphs which are members of all other clusters.

6. Experiments and Results

As part of the British Telecom's SATURN Research programme (Rowlingson et al., 2013), a cyber range experiment was carried out in 2012 by industrial partners (Winter, 2012). This experiment was used as a case study in evaluating ACSAnIA. The cyber range experiment models a simulated computer network which comprises of two sub-networks – a main network with approximately 200 workstation clients and a branch network comprising of 10 workstation clients. To model real network activities, the experiment utilised comprehensive scripts for simulating email sending, server activity and content download activities. Figure 6 shows the network architecture.

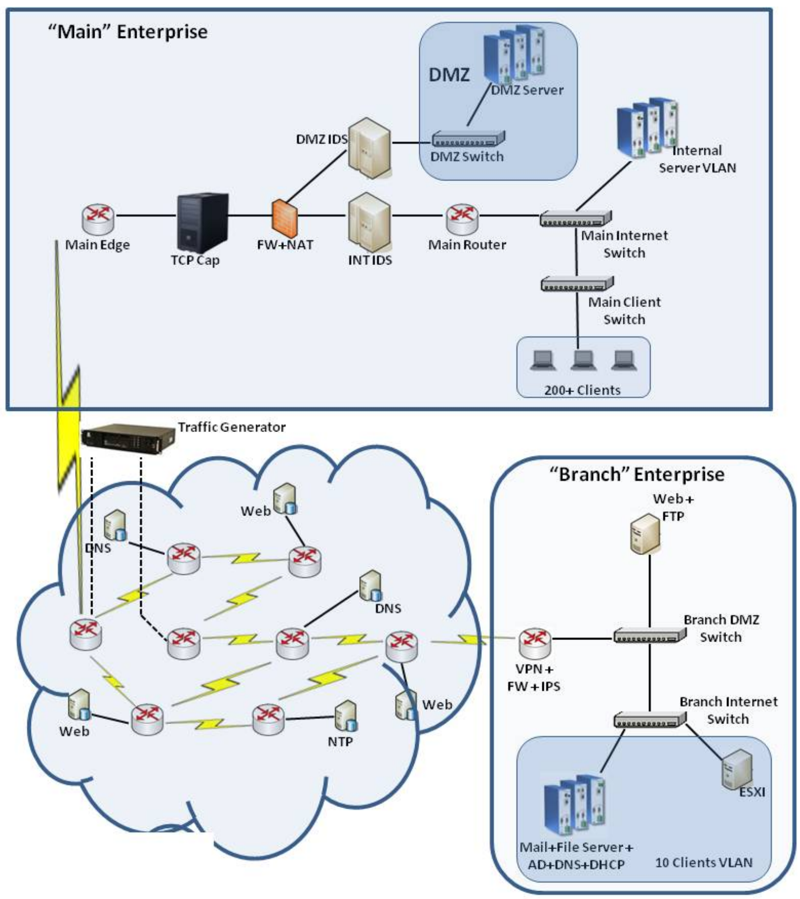


Figure 6: Network Architecture of Cyber Range Simulated Network

The cyber range experiment includes two simulated attacks on the modelled network. The alert logs generated from both the Delimitised Zone IDS and Internal IDS are analysed by ACSAnIA. (These are shown in Figure 6 as DMZ IDS and INT IDS).

During the experiments, ACSAnIA was deployed on a 64-bit Windows System with an Intel(R)Core i5 CPU processor at 2.40GHz, JVM 1.4.2 and 6GB for maximum heap memory.

6.1. Attack 1 – Main Network DMZ

A web server in the DMZ zone of Figure 6 is attacked by an offsite attacker. The attack comprised of 4 phases – DMZ Scanning (Casual & Intense), vulnerability assessment, exhaustive penetration and brute-force audit from the attacker on the web server. Normal network activity includes server activity such as email routing from a DMZ mail server to an internal mail server and network pinging between the DMZ mail and FTP servers.

The following configurations were applied to the ACSAnIA system for analysing the intrusion logs from Attack 1.

Components	**Correlation**		**Prioritising**		**Clustering**		**Pattern**
Parameters	C_θ	T_θ	k	P_θ	k	ϵ	$minSupp$
Values	0.7	30 mins	3	3	2	2	2

Table 3: System Configurations for Experiment on Attack 1

The Snort DMZ IDS (shown in Figure 6) captured 3225 alerts during the course of Attack 1. 649 alerts were false positives while 2577 were triggered as a result of the attack. Due to the low volume of alerts received, when ACSAnIA received the alerts for post processing, we allowed the *correlation component* to correlate alerts within a 30 minute time window. At the same time, in order to maintain high quality correlations, we set a correlation threshold of 0.7. The remaining parameters in Table 3 were set experimentally.

Our expected result was that the ACSAnIA prioritising component would filter out false positives and its clustering and attack pattern discovery components would aid in understanding the attack scenario.

6.1.1. Results

According to the default priorities assigned by a Snort DMZ IDS, 78% of the alerts were within medium priorities (i.e. priority 2 & 3). After ACSAnIA re-computed the priority levels, it was observed that alerts were regrouped as either very low (i.e. priority 1) or very high (i.e. priority 4) but not in-between. ACSAnIA successfully creates a margin between alerts by identifying alerts which are outliers as high priority and alerts which are common intrusion activity as low priority. This is shown clearly in Figures 7 and 8. In the attack 1 scenario, common intrusion activity corresponded to false positive alerts therefore low priority. With a priority threshold $P_\theta = 3$, ACSAnIA would only report alerts at Priority 4 and would achieve a TPR and FPR of 71% and 0.3% respectively.

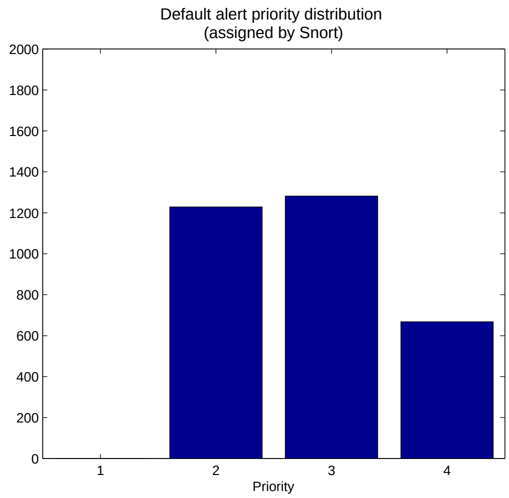


Figure 7: Attack 1: Default priority assignment (before ACSAnIA)

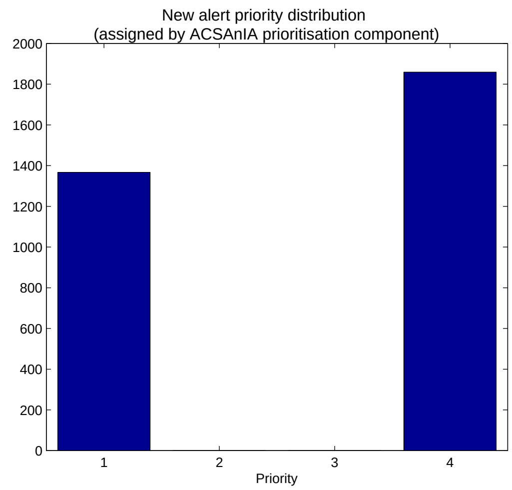


Figure 8: Attack 1: New Priority assignment (after ACSAnIA)

The Clustering component also successfully separates low level alerts (false positives) from high level alerts (true positives). Our correlation component generated 165 meta-alerts. In Figure 9, meta-alerts(alert correlaton graphs) within the cluster are common intrusion activity and therefore lower priority . The more unusual the intrusion activity, the further apart from the cluster and the higher the priority.To measure the quality of the cluster, the silhouette coefficient was applied and yielded 0.983. This implies good clustering.

The clustering analysis and attack pattern discovery reveals that most meta-alerts that form the dense area of the cluster in Figure 9 are smaller meta-alerts which capture small network intrusions, particularly benign activity. These meta-alerts include intrusion alerts raised as a result of the simulated network activity such as network pinging and email routing. Some of the automated email content routed across the network contained suspicious but not necessarily malicious PDF content. The outliers to the clusters however represent larger meta-alerts. The majority of the meta-alerts contained intrusion alerts that capture intense network scanning and exploit activity on the web server.

6.2. Attack 2 – Main Network Internal

A vulnerable host is situated in the main enterprise network amongst the 200 clients(illustrated in Figure 6). It is compromised by an off-site attacker and by an internal attacker. The internal attacker had been previously compromised by the off-site attacker. The internal attacker is supposedly situated in the branch sub-network of the computer network. The attack consists of the following phases:

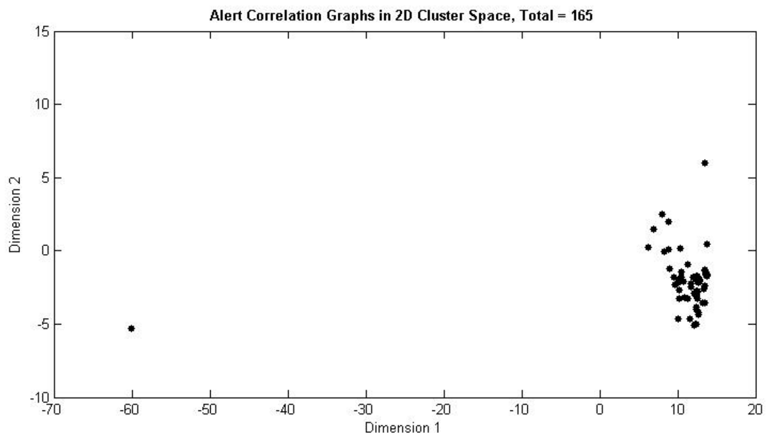


Figure 9: Clustering distribution of Attack 1 meta-alerts

1. Network scanning i.e. reconnaissance from the internal attacker to the main network
2. Failed exploit attempts on branch client workstations from internal attacker using shellcode embedded in email content
3. Successful exploit on internal vulnerable machine from internal attacker using shellcode embedded in email content
4. Gain of full control and admin access of vulnerable machine from offsite attacker
5. Offsite attacker uses vulnerable machine to explore internal network

The system configurations for this scenario are altered to effectively analyse the sheer volume of alerts in this attack scenario in comparison to Attack 1. Table 4 shows the adjusted system configurations.

Components	**Correlation**		**Prioritising**		**Clustering**		**Pattern**
Parameters	θ	T_θ	k	P_θ	k	ϵ	$minSupp$
Values	0.8	5 mins	12	2	48	2.4	5

Table 4: System Configurations for Experiment on Attack 1

The Snort INT IDS (shown in Figure 6) captured approximately 125,000 alerts during the course of Attack 2. After filtering the alerts, only 34,697 alerts were analysed by ACSAnIA. It is to be noted that the labelling of this dataset is still in progress. To our knowledge, only 88 alerts in the alert log have been identified as true positive alerts. Under these circumstances the worst case FPR rate of the IDS is approximately 99%. In addition, it was discovered that the IDS did not trigger alerts of some phases of the attack, particularly the Phase 3. Due to the sheer volume of alerts received, the ACSAnIA correlation component was set to a smaller time window, 5 mins. We maintain high quality correlations by using a strict correlation threshold 0.8. The remaining values were set experimentally.

6.2.1. Results

In Attack 2, both the Snort and ACSAnIA assignment of the alert priorities follow a similar pattern as Attack 1. Figure 10 shows that according to Snort's priority assignment, it is observed that majority of the alerts are within medium priorities. This scenario is not as straight forward as Attack 1 i.e. not all inliers correspond to false positives and low priority. From Figure 11, It is observed that ACSAnIA regroups common intrusion activity into Priority 1 and Priority 2(low to medium priority) and still succesfully prioritises unusual activity as higher priority.

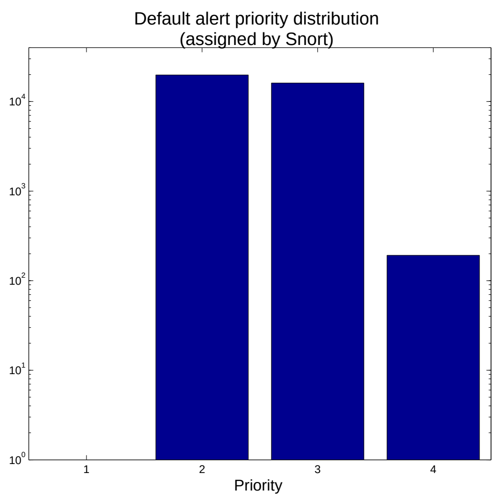


Figure 10: Attack 2: Default priority assignment (before ACSAnIA)

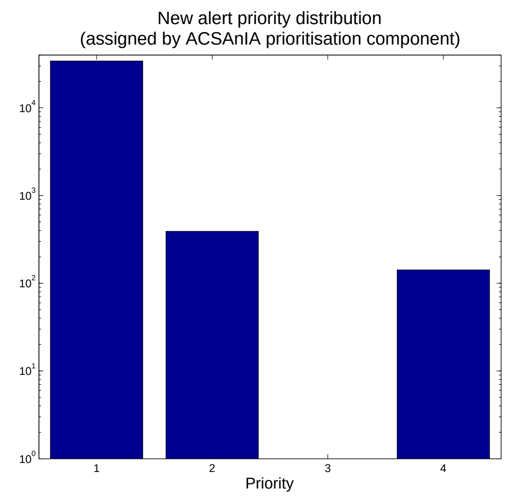


Figure 11: Attack 2: New Priority assignment (after ACSAnIA)

By ACSAnIA filtering all alerts with priority less than 2, only 533 were prioritised. Of these alerts, 88 were true positive alerts and were contained in a single meta-alert. Thus we can say our true positive rate is 100% (note that this is relative to the 88 alerts identified as true positives by the IDS). Although the false positive rate remains high at 83.4%, it is 16% less than the initial FPR of the IDS which raised the alerts. The correlation component generated 2,369 meta-alerts. The clustering component identified one large cluster and 73 outliers with a silhouette coefficient of 0.989. In Figure 12, the clusters are represented. meta-alerts part of the "Cluster" correspond to those in lower priorities, meta-alerts further apart from the cluster correspond to unusual activity and therefore are assigned higher priority.

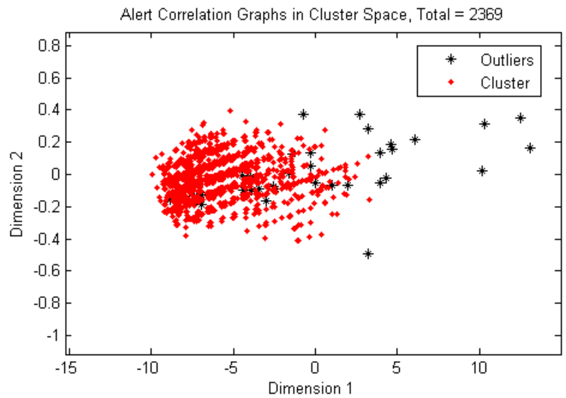


Figure 12: Clustering distribution of Attack 2 meta-alerts

6.2.2. Conclusion

A key observation in the results in both attack scenarios 1 & 2 is that ACSAnIA assigns higher priorities to the alerts which the previous Snort IDSs also prioritised highly. Hence, there is a consistency between both systems. This could be as a result of the following: in both scenarios, highly prioritised alerts have the least frequencies. In a scenario where the vast amount of alerts are supposed to be high priorities, ACSAnIA is unlikely to perform well. This is particularly because ACSAnIA strongly correlates unusual and infrequent activity with higher priorities.

In general, ACSAnIA significantly reduces the volume of alerts a Security Analyst needs to inspect first through correlating alerts into higher level abstract alerts called meta-alerts and then by filtering out low priority meta-alerts. Using the reporting system, a security analyst can explore the clusters and patterns of each meta-alert. Thus, ACSAnIA provides a platform for understanding attack patterns.

7. Summary

In this paper a system for analysing intrusion alerts has been presented. The usefulness of the post-correlation analysis has been demonstrated by evaluating the quality of core post correlation component's output. From evaluating the prioritisation component, we have shown that in one scenario we can reduce false positives by 97% and in another by 16%. The significant reduction in the first scenario is likely due to the small scale of the attack. Nonetheless, both are significant reductions. It is also to be noted that the prioritisation phase is mainly based on the likelihood that unusual intrusion activity corresponds to high alert priorities. For example, ACSAnIA would prioritise unusual intrusion actitivity that corresponded to noise as high priorities rather than low priorities.

The evaluation of the cluster component showed that some alert correlation graphs may share similar characteristics and thus alert correlation graphs can be grouped into clusters. We evaluated the selected clustering parameters by using the silhouette coefficient which averaged to around 0.9 showing good clustering quality. To the author's best knowledge, this research is the first to investigate and successfully carry out alert correlation graph based clustering using traditional clustering and distance metrics. Finally, we discussed how attack patterns can be extracted from alert correlation graphs. In future, we intend to perform more experiments on real-time attack pattern extraction.

The ACSAnIA System is currently being integrated into the British Telecom's Security Assure Analytics tool-kit for analysing Cyber security data.

8. FutureWork

The future work of this research mainly focusses on improving the clustering component to perform real-time clustering.

Our approach to prioritisation and clustering is based on batch analysis, i.e. at every set interval, a set of recent meta-alerts are extracted from the database and their prioritisation values are calculated. Consequently, they are clustered. In today's computer networks where intrusion activity is highly dynamic, and data volumes are high a batch approach may not be feasible. To address this, real-time prioritisation and clustering methods are being investigated. In particular we are investigating Incremental local outlier detection Pokrajac and Hartford (2007) and real-time graph clustering Aggarwal et al. (2010).

9. Acknowledgements

This work is being supported and funded by the British Telecommunications Security Future Practice Group, the Centre for Cyber Security Sciences(CCySS) at City University London and the Engineering and Physical Sciences Research Council (EPSRC).

Our appreciation also extends to Northrop Grumman who provided the cyberange data for the experiment.

References

Aggarwal, C. C., Zhao, Y., Yu, P. S., 2010. On Clustering Graph Streams. Proceedings of the 2010 SIAM International Conference on Data Mining, 478–489.

Ahmadinejad, S. H., Jalili, S., 2009. Alert Correlation Using Correlation Probability Estimation and Time Windows. 2009 International Conference on Computer Technology and Development (1), 170–175.

Alienvault, 2013. AlienVault Unified Security Management.
URL `http://www.alienvault.com/solutions/siem-event-correlation`

Alireza Sadighian, J. M. F., 2013. ONTIDS: A highly flexible context-aware and ontology-based alert correlation framework.

Alsubhi, K., Aib, I., Boutaba, R., 2012. FuzMet : a fuzzy-logic based alert prioritization engine for intrusion detection systems. International Journal of Network Management 22 (4), 263–284.

Alsubhi, K., Al-Shaer, E., Boutaba, R., 2008. Alert prioritization in Intrusion Detection Systems. NOMS 2008 - 2008 IEEE Network Operations and Management Symposium, 33–40.

Benferhat, S., Boudjelida, A., Tabia, K., Drias, H., 2013. An intrusion detection and alert correlation approach based on revising probabilistic classifiers using expert knowledge. Applied intelligence 38 (4), 520–540.

Breunig, M. M., Kriegel, H.-p., Ng, R. T., Sander, J., 2000. LOF : Identifying Density-Based Local Outliers. Proceedings Of The 2000 Acm Sigmod International Conference On Management Of Data, 1–12.

Cédric Michel, L. M., 2001. Adele: An Attack Description Language For Knowledge-Based Intrusion Detection. Trusted Information, 353–368.

Chen, S., Leung, H., Dondo, M., May 2014. Characterization of computer network events through simultaneous feature selection and clustering of intrusion alerts. In: Braun, J. J. (Ed.), SPIE Sensing Technology + Applications. International Society for Optics and Photonics, p. 912107.

Cheung, S., Fong, M. W., Ave, R., Park, M., 2003. Modeling Multistep Cyber Attacks for Scenario Recognition. In DARPA Information Survivability Conference and Exposition (DISCEX III) (DISCEX III), 284–292.

Cuppens, F., Ortalo, R., Oct. 2000. LAMBDA: A Language to Model a Database for Detection of Attacks. Recent advances in intrusion detection. Springer Berlin Heidelberg, 197–216.

Dain, O., Cunningham, R. K., 2001. Fusing a Heterogeneous Alert Stream into Scenarios. In Proceedings of the 2001 ACM workshop on Data Mining for Security Applications, 1–13.

Debar, H., Wespi, A., 2001. Aggregation and Correlation of Intrusion-Detection Alerts. Recent Advances in Intrusion Detection., 85–103.

Ester, M., Kriegel, H.-p., Xu, X., Miinchen, D., 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. KDD 96.

Hofmann, A., Sick, B., 2011. Online Intrusion Alert Aggregation with Generative Data Stream Modeling. IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING 8 (2), 282–294.

IBM Corporation, Dec. 2013. IBM Intelligent Operations Center - Demo - Software.
URL `www-01.ibm.com/software/industry/intelligent-oper-center`

Khan, A., Yan, X., Wu, K.-L., 2010. Towards proximity pattern mining in large graphs. Proceedings of the 2010 international conference on Management of data - SIGMOD '10, 867.

Lagzian, S., 2012. Frequent Item set mining-based Alert Correlation for Extracting multi-stage Attack Scenarios. IEEE Telecommunications (IST), 2012 Sixth International Symposium, 1010–1014.

Marchetti, M., Colajanni, M., Manganiello, F., 2011. Identification of correlated network intrusion alerts Pseudo-Bayesian. Cyberspace Safety and Security (CSS), 2011 Third International Workshop on, 15–20.

Ning, P., Reeves, D. S., Cui, Y., 2001. Correlating Alerts Using Prerequisites of Intrusions. Tech. rep., North Carolina State University, Raleigh NC,.

Ning, P., Xu, D., 2003. Learning attack strategies from intrusion alerts. Proceedings of the 10th ACM conference on Computer and communication security - CCS '03, 200.

Noel, S., Jajodia, S., 2007. Attack Graphs for Sensor Placement , Alert Prioritization , and Attack Response. Cyberspace Research Workshop, 1–8.

Patel, H., 2009. Intrusion Alerts Analysis Using Attack Graphs and Clustering. Masters, San Jose State University.

Pokrajac, D., Hartford, E., 2007. Incremental Local Outlier Detection for Data Streams. Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on (April).

Porras, P. A., Fong, M. W., Valdes, A., 2002. A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. Recent Advances in Intrusion Detection (Springer Berlin Heidelberg), 95–114.

Qin, X., 2005. A Probabilistic-Based Framework for INFOSEC Alert Correlation. Ph.D. thesis, Georgia Institute of Technology.

Ren, H., Stakhanova, N., Ghorbani, A. A., 2010. An Online Adaptive Approach to Alert Correlation. Proceedings of the 7th international conference on Detection of Intrusions and malware, and vulnerability assessment (DIMVA), 153–172.

Rowlingson, R., Healing, A., Shittu, R., Matthews, S. G., Ghanea-Hercock, R., 2013. Visual Analytics in the Cyber Security Operations Centre. Proceedings of The Information Systems Technology Panel Symposium on Visual Analytics.

Sadoddin, R., Ghorbani, A. a., May 2009. An incremental frequent structure mining framework for real-time alert correlation. Computers & Security 28 (3-4), 153–173.

Salah, S., Maciá-Fernández, G., Díaz-Verdejo, J. E., Jan. 2013. A model-based survey of alert correlation techniques. Computer Networks.

Shapiro, L. G., Haralick, R. M., May 1981. Structural descriptions and inexact matching. IEEE transactions on pattern analysis and machine intelligence 3 (5), 504–19.

Shittu, R., Healing, A., Ghanea-hercock, R., Bloomfield, R., 2014. OutMet : A New Metric for Prioritising Intrusion Alerts using Correlation and Outlier Analysis. 19th IEEE Conference on Local Computer Networks.

Steven Eckmann , G. V., 2002. STATL: An Attack Language for State-based Intrusion Detection. Journal of Computer Security 10 (1), 71–103.

Sundaramurthy, S. C., Zomlot, L., Ou, X., 2011. Practical IDS alert correlation in the face of dynamic threats. International Conference on Security and Management (SAM'11).

Tekhov, R., 2009. Graph Edit Distance Project. Tech. rep.

Valdes, A., Skinner, K., 2001. Probabilistic Alert Correlation. In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, 54–68.

Verizon, 2013. Data Breach Investigations Report.
URL `http://www.verizonenterprise.com/DBIR/2013/`

Winter, H., 2012. System security assessment using a cyber range. 7th IET International Conference on System Safety, incorporating the Cyber Security Conference 2012, 41–41.

Yan, X., 2002. gSpan: graph-based substructure pattern mining. 2002 IEEE International Conference on Data Mining, 2002. Proceedings. (d), 721–724.

Zali, Z., Hashemi, M. R., Saidi, H., Aug. 2013. Real-Time Intrusion Detection Alert Correlation and Attack Scenario Extraction Based on the Prerequisite-Consequence Approach.

Zomlot, L., Sundaramurthy, S. C., Luo, K., Ou, X., Rajagopalan, S. R., 2011. Prioritizing intrusion analysis using Dempster-Shafer theory. Proceedings of the 4th ACM workshop on Security and artificial intelligence - AISec '11, 59.