

Methods for Detecting Important Events and Knowledge From Data Security Logs

Risto Vaarandi
CCD COE, Tallinn, Estonia
risto.vaarandi@ccdcoe.org

Abstract: In modern computer networks and IT systems, event logging is commonly used for collecting system health information, in order to ease the system management process. For example, many sites are collecting events and network flow records from their applications, servers, and network devices over protocols like syslog, SNMP and Netflow, and analyze these data at central monitoring server(s). Among collected data, many events and records provide information about security incidents. Unfortunately, during the last decade security logs have grown rapidly in size, making the manual analysis extremely labor intensive task. This task is further complicated by the large number of irrelevant records and false positive alerts in security logs. For this reason, the development of methods for detecting important events and knowledge from security logs has become a key research issue during the recent years. In our paper, we propose some methods for tackling this issue in the context of IDS and Netflow logs from an organizational network. The first contribution of this paper is the study of important properties of IDS and Netflow logs. We have conducted our analysis on a number of production system logs obtained from a large financial institution, and some of our findings are supported by results from other researchers. The second contribution of the paper is the proposal of several data mining based and heuristic methods for event and knowledge detection from security logs. Our data mining methods are based on frequent itemset mining for identifying regularities in IDS alert sets and network traffic. These regularities are then used for finding unexpected IDS alert patterns and prominent network traffic flows. In this paper, we also discuss the implementations of the proposed methods in a production environment, and provide performance estimates for our implementations. We conclude the paper with a short discussion on some promising directions for further research.

Keywords: data mining, security log analysis

1. Introduction

In modern computer networks and IT systems, one of the key security management techniques is network monitoring for detecting unwanted, malicious or anomalous traffic. Two widely employed methods for network monitoring are the use of network intrusion detection system (IDS) and the collection of network traffic information with protocols like Netflow or IPFIX. A network IDS sensor performs *deep packet inspection* (DPI) for a network segment – for every packet that traverses the segment, the sensor analyzes both the packet headers and its payload. Most network IDSs use signature based approach for DPI – human experts write packet matching conditions (*signatures*), in order to recognize known bad traffic (e.g., a signature could be a regular expression for matching the packet payload). When network traffic matches a signature, IDS triggers an alert which is typically sent to the central network management server. Unfortunately, IDSs are known to generate large volumes of alerts – for example, a single IDS sensor can emit hundreds of thousands of alerts per day (Vaarandi and Podiņš 2010; Viinikka, Debar, Mé, Lehtikainen and Tarvainen 2009). Furthermore, usually the majority of these alerts are false positives or irrelevant (Julisch 2001; Long, Schwartz and Stoecklin 2006; Vaarandi and Podiņš 2010). Therefore, the manual review of IDS logs is often impossible.

In contrast, when network traffic information is collected from routers, switches or dedicated network probes (with a protocol like Netflow), only data from packet headers are considered. For example, a Netflow record contains a transport protocol ID (e.g., 6 for TCP), source and destination IP addresses, source and destination ports (if supported by transport protocol), and a few other fields. A Netflow record is created when the network device first observes relevant traffic flow (e.g., a TCP connection is established from the workstation 10.2.1.13 port 21892 to the web server 10.1.1.1 port 80). Typically, the network device sends the record to the central network management host when activity or inactivity timer expires for the flow (e.g., no packets have travelled from source to destination during 15 seconds), when the flow table becomes full, or when the flow ends (e.g., the corresponding TCP connection is terminated). Since collecting network traffic information does not involve the packet payload analysis, it requires much less computing resources than DPI. However, since in larger networks many millions of flow records can be created within a short amount of time (Wagner 2008), processing and storing these records is expensive in terms of CPU time and disk space. In order to reduce these costs, *packet sampling* is usually employed in very large networks – traffic information is only extracted from a fraction of packets (e.g., 0.1%). Nevertheless, packet sampling is often not used in the context of network security

monitoring, since this allows for recording all network packet flows between peers and thus for the detection of unusual traffic patterns.

In this paper, we will focus on IDS and Netflow log analysis for organizational networks. We will first study important properties of IDS and Netflow logs and will show that these data sets are prone to contain strong patterns. We will then propose several heuristic and data mining based algorithms for analyzing these logs. The remainder of this paper is organized as follows – section 2 describes related work, section 3 focuses on properties of IDS and Netflow logs, section 4 describes log analysis algorithms which harness these properties, and section 5 concludes the paper.

2. Related work

Since IDS and Netflow logs contain large volumes of data and it is highly impractical to review these logs manually, their analysis has attracted a considerable amount of attention in the research community. A number of methods have been proposed during the last decade, including machine learning (Pietraszek 2004), time series analysis (Viinikka, Debar, Mé and Séguier 2006; Viinikka, Debar, Mé, Lehtikainen and Tarvainen 2009), the application of EWMA control charts (Viinikka and Debar 2004), visualization (Taylor, Paterson, Glanfield, Gates, Brooks and McHugh 2009), the use of locality paradigm (McHugh and Gates 2003) and chronicles formalism (Morin and Debar 2003), the application of game theory (Wagner, Wagener, State, Engel and Dulaunoy, 2010), graph based methods (Ning, Cui and Reeves 2002), etc.

Among recently proposed methods, data mining algorithms have been often suggested for IDS alert logs. With these methods, IDS alert logs are mined for previously unknown regularities and irregularities. This knowledge is then used by human experts for writing event correlation rules which highlight important alerts and filter out large volumes of false positives and other irrelevant alerts. Long, Schwartz and Stoecklin have developed a supervised clustering algorithm for distinguishing Snort IDS true alerts from false positives (Long, Schwartz and Stoecklin 2006). Treinen and Thurimella have investigated the application of association rule mining, in order to detect knowledge for writing event correlation rules for novel attack types (Treinen and Thurimella 2006). Clifton and Gengo have suggested a similar approach for creating IDS alert filters (Clifton and Gengo 2000). Julisch and Dacier have proposed a conceptual clustering technique for IDS alert logs (Julisch 2001; Julisch and Dacier 2002; Julisch 2003). With this approach, detected clusters correspond to alert descriptions, and the human expert can use them for developing filtering and correlation rules for future IDS alerts. Al-Mamory, Zhang and Abbas have proposed clustering algorithms for finding generalized alarms which help the human analyst to build filters (Al-Mamory, Zhang and Abbas 2008; Al-Mamory and Zhang 2009). Vaarandi and Podiņš have developed a novel data mining based method for IDS alert classification (Vaarandi and Podiņš 2010). The method fully automates the knowledge interpretation process which has been traditionally carried out by human experts, and derives alert classification rules without a human intervention. These rules are used for distinguishing important alerts from irrelevant ones.

Various data mining methods have also been proposed for the analysis of Netflow logs. Wagner has applied entropy measurement techniques to Netflow data, in order to detect worms in fast IP networks (Wagner 2008). Paredes-Oliva et al. have employed a frequent itemset mining algorithm for identifying traffic flows that are root-causes of network security anomalies (Paredes-Oliva, Dimitritopoulos, Molina, Barlet-Ros and Brauckhoff 2010). Li and Deng have proposed several frequent pattern mining algorithms, in order to detect network anomalies (Li and Deng 2010). Also, Vaarandi has proposed frequent itemset mining for automated close-to-real-time identification of strong traffic patterns from Netflow logs (Vaarandi 2008).

3. Properties of IDS and Netflow logs

During our experiments, we have discovered several important properties of IDS and Netflow logs which are confirmed by findings of other researchers. When investigating the properties of IDS alert log data, we reviewed the yearly logs of three IDS sensors from a large financial institution. Sensors had more than 15,000 signatures and were deployed in different locations (both in intranet and public Internet). Two logs contained more than 50 million alerts and one log more than 2 million alerts.

Firstly, we found that majority of the alerts were triggered only by a few signatures – 10 most verbose signatures created more than 95% of alerts for two sensors and more than 80% of alerts for one sensor. Other researchers have reported similar findings – in (Viinikka, Debar, Mé and Séguier 2006) it was found that 5 signatures produced 68% of alerts, while in (Viinikka, Debar, Mé, Lehtikainen and Tarvainen 2009) the authors discovered that 7 signatures produced 78% of alerts. Secondly, we found that prolific

signatures usually trigger large volumes of alerts over longer periods of time. For three aforementioned sensors, less than 25 signatures triggered alerts for more than 300 days during 1 year period, and these alerts constituted 70-90% of entries in the logs. Finally, vast majority of these verbose signatures trigger false positives or irrelevant alerts. In our experimental environment, we found that they are mostly related either to well-known threats (such as MS Slammer Sapphire worm) or legitimate network traffic (like SNMP queries from network management servers). Similar findings have also been reported in (Viinikka, Debar, Mé, Lehikoinen and Tarvainen 2009). Therefore, IDS alert logs contain strong patterns in many environments, and these patterns describe commonly occurring irrelevant alerts.

For investigating the properties of organizational Netflow logs, we studied the log of a Netflow probe that was deployed in a backbone network of a large financial institution. The probe collected information about network traffic for hundreds of workstations, tens of servers and various other devices without packet sampling. The log of the probe covered the period of 14 days and contained 104,142,530 Netflow records. In order to detect changes in network usage patterns over time, we divided the 14 day (336 hour) log into 336 non-overlapping time frames, with each frame covering 1 hour. In the remainder of the paper, *destination address* denotes the following tuple: (*transport protocol ID*, *destination IP address*, *destination port*). Note that Netflow records for portless transport protocols (like ICMP) might use the destination port field for specifying the type of the packet.

Firstly, we noticed that the number of distinct destination addresses is quite large for each time frame – each frame contained an average of 309,948 records and an average of 103,335 destination addresses. However, the majority of destination addresses (90-98%, an average of 92.6% per frame) were associated with only one source IP address. Also, most such destination addresses appeared only in a few records during short period of time. Furthermore, only 47-121 destination addresses had 20 or more source IP addresses in a frame, but 35-46% of Netflow records represented the network traffic to these few destinations. When we investigated these destination addresses more closely, we found that they correspond to widely used network services (for example, corporate mail and web servers). Due to the large volume of traffic going to these services, strong patterns that reflect this traffic show up in Netflow logs.

Secondly, when analyzing the network traffic of workstations, we discovered that a typical workstation communicates with a limited number of IP addresses within 1 hour time frame – the average number of peer addresses per workstation ranged from 9.8 to 24.7 in 336 frames. Thirdly, we also found that many workstations often communicate only with well-known network services which are simultaneously used by several other network nodes. Inspecting 1 hour time frames revealed that 68-94% of workstations (an average of 88.3% per frame) did only interact with network services used by at least 4 other nodes during the same time frame. Other researchers have observed similar regularities in workstation network traffic (McHugh and Gates 2003).

These properties of Netflow log data for organizational networks clearly indicate that such logs contain strong patterns. Furthermore, these patterns often reflect the use of well-known network services (by workstations and other legitimate clients). For these reasons, the emergence of new and unusual patterns might be a symptom of an anomalous (and possibly malicious) network activity. In addition, the discovery of network services from Netflow logs facilitates the identification of illegal services. In the following section we will present several algorithms for addressing these issues.

4. Anomaly detection algorithms for Netflow and IDS logs

4.1 Frequent pattern mining from Netflow logs

In order to mine patterns from Netflow logs, we propose a frequent itemset mining based approach. Although various frequent itemset mining algorithms have been often suggested for various log types (see (Vaarandi 2004) for references), their application for Netflow data sets is fairly novel and we are aware of only a few recent works (Vaarandi 2008; Paredes-Oliva, Dimitritopoulos, Molina, Barlet-Ros and Brauckhoff 2010; Li and Deng 2010).

Let $I = \{i_1, \dots, i_n\}$ be a set of items. If $X \subseteq I$, X is called an *itemset*. A *transaction* is a tuple (tid, X) , where tid is a transaction identifier and X is an itemset. A *transaction database* D is a set of transactions, and the *support* of an itemset X is the number of transactions that contain X : $supp(X) = |\{tid \mid (tid, Y) \in D, X \subseteq Y\}|$. If s is a support threshold and $supp(X) \geq s$, X is called a *frequent itemset*. Note that if the support threshold is specified as a percentage $p\%$, then $s = |D| * p / 100$. If itemset X does not have any proper

supersets with the same support, X is called a *closed itemset*. In this paper, we focus on mining frequent closed itemsets, since they are a compact and lossless representation of all frequent itemsets.

For mining patterns from Netflow logs, we are using LogHound data mining tool which has been developed for efficient mining of very large logs (Vaarandi 2004). If a Netflow record reflects a flow of m network packets between some source and destination transport address, we view this record as a set of m transactions with identical itemset $\{(sourceIP,1), (sourcePort,2), (destinationIP,3), (destinationPort,4), (protocol,5)\}$. In other words, we order the five relevant flow record attributes, in order to distinguish identical values of different attributes during the mining. With this representation, each itemset describes a traffic pattern, and the support of the itemset equals to the number of packets for this pattern. In the rest of the paper, we use the terms pattern and itemset interchangeably. Figure 1 depicts some frequent traffic patterns detected with LogHound.

```
** 10.16.23.3 162 17
Support: 161657

10.12.47.1 993 * * 6
Support: 166959

10.13.25.14 80 10.11.48.44 1915 6
Support: 1211532
```

Figure 1: Sample traffic patterns detected with LogHound

The first pattern reveals that 161,657 UDP packets have been sent from various sources to SNMP trap collector (port 162/udp) at 10.16.23.3, while the second pattern reflects 166,959 TCP packets sent to various destinations from secure IMAP server (port 993/tcp) at 10.12.47.1. The third pattern indicates that port 1915/tcp at the node 10.11.48.44 has received 1,211,532 TCP packets from the web server (port 80/tcp) at 10.13.25.14.

For mining traffic patterns from Netflow data, we propose the following framework. After every W second time interval, frequent closed patterns are detected from the Netflow data of last W seconds and stored to disk. The content of the file can be viewed over the web by the security administrators for getting a quick overview of most prominent recent network traffic patterns. In addition, for each detected pattern the last N pattern files are scanned, in order to detect in how many files the pattern is present. If the pattern has occurred in less than K files, the pattern is highlighted as potentially anomalous.

We have implemented this framework for analyzing data from a Netflow probe in a backbone network of a large financial institution (see section 3 for the probe deployment details). We measured the algorithm performance during 21 days, with the support threshold set to 1%, W set to 3600 seconds, N set to 96 and K to 12. In other words, the algorithm mined patterns once in every hour, and highlighted each pattern which had occurred in less than twelve 1-hour windows during the last 4 days. During the experiment, 56-261 patterns were detected (an average of 182.5 per window), and 3-140 patterns were highlighted (an average of 54.3 per window). All highlighted patterns corresponded to system and network management activity which does not occur routinely on everyday basis. Thus the algorithm is able to identify unusual strong network traffic patterns.

4.2 Network service detection from Netflow logs

Identification of network services in organizational networks is an important task. Firstly, new legitimate services are discovered which eases the configuration management process. Secondly, unexpected or illegal services might be found that violate security policies or have been created with malicious intentions (e.g., for leaking data to Internet). Today, network services are often detected with dedicated network/host scanning tools like Nmap. However, scanning larger networks is time-consuming and requires a lot of network bandwidth. In addition, scanning is an intrusive technique which might alert the illegal service provider. Furthermore, scanning could trigger many alarms in the security monitoring system of the organization itself (e.g., host firewalls might report all their ports that were scanned). Finally, the illegal service might be protected with a firewall, denying access for known security monitoring hosts.

The approach proposed in section 4.1 is able to identify actively used network services which receive or send large amounts of network packets. However, in many cases the amount of data sent and received by services is modest. For example, during our experiments described in section 4.1 we discovered that many services exchanged less packets with clients than the support threshold, and thus remained undetected. Unfortunately, lowering the support threshold will substantially increase the number of patterns, thus making it hard for the human to spot patterns that correspond to services. Furthermore, mining large data sets with very low support thresholds will also increase the CPU and memory consumption of the algorithm.

In this section, we propose a non-intrusive algorithm for real-time service detection from Netflow logs. The algorithm processes Netflow records immediately after their arrival to the network monitoring server, and employs the following heuristic – if the destination address is employed for providing an actively used service, this address is likely to show up in Netflow logs repeatedly during longer periods of time. In contrast, as discussed in section 3, most destination addresses appear only in few records during short time.

The algorithm employs memory based lists L_0, \dots, L_n for destination address analysis, where each list is allocated for destination addresses with a certain number of associated source IP addresses. For each list L_i , W_i specifies the size of the analysis window in seconds and T_i the threshold for number of sources (T_n is set to infinity; $T_i < T_{i+1}$ and $W_i \leq W_{i+1}$, $0 \leq i < n$). These lists allow for treating more widely used destination addresses differently during the analysis, and are also useful for the grouping purposes during reporting.

For each incoming Netflow record, the algorithm applies the following steps:

- 1) extracts the source IP address S and destination address D from the Netflow record,
- 2) if D belongs to list L_i , S is appended to the peer list P_D ; if during the last W_i seconds T_i distinct entries were appended to P_D , D is moved to list L_{i+1} ,
- 3) if D is not present in lists L_0, \dots, L_n , D is inserted into list L_0 and S is appended to P_D .

After short time intervals (e.g., once in a second), the algorithm checks all destination addresses. If the destination address D belongs to L_i , entries appended to P_D more than W_i seconds ago are removed for memory saving purposes. Also, if the destination address D belongs to list L_i ($0 < i \leq n$) and during the last W_i seconds less than T_{i-1} distinct entries were appended to P_D , D is moved to list L_{i-1} . If the destination address D belongs to L_0 and during the last W_0 seconds no entries were added to P_D , D is removed from L_0 .

It is easy to see that if the destination address is actively used by larger number of sources, it will be promoted to higher level lists, while if the number of active peers decreases, the address will be moved back to lower levels. If the address in L_0 has been without peers for W_0 seconds, it will be dropped from memory. Otherwise it will stay in one of the lists and have a chance for promotion if its peer activity increases. If T_0 is set to 2, L_0 will contain destination addresses with only one associated source during the last W_0 seconds. Since the majority of destination addresses do not correspond to network services and appear briefly in a few records with one source only (see section 3), they will only stay in L_0 , being dropped shortly after W_0 seconds. Therefore, the algorithm will not consume large amounts of memory.

During our experiments, we have used the value of 3600 seconds for W_0 which represents a good tradeoff between low memory consumption and service detection precision. We have also set n to 3, W_1 to 7200 seconds, both W_2 and W_3 to 1440 seconds, T_0 to 2, T_1 to 5, and T_2 to 20. In other words, we have used four lists for destination addresses with 1 source during 1 hour, with 2-4 sources during 2 hours, with 5-19 sources during 4 hours, and with 20 or more sources during 4 hours.

We have configured the algorithm to produce output in several ways:

- A web report is created once in 5 minutes from destinations in lists L_1, \dots, L_n ,
- When a new destination is created in L_1 or an entry has stayed in L_0 for more than K seconds, a syslog message is produced about the appearance of new service (we have set K to 86400, in order to detect services which have been consistently used by one peer during 1 day).

During the experiment of 14 days, the memory consumption of the algorithm was low as we had expected. The L_0 , L_1 , L_2 , and L_3 lists remained limited in size and contained 1054-3845, 95-445, 7-75, and 45-133 entries, respectively. Also, 6381 syslog messages about the appearance of new services were logged. However, 3267 (59%) of them were repeated messages about 656 well-known services (in most cases, services were rediscovered after nightly peer inactivity). Among remaining 3114 messages, some were false positives generated by a few network management hosts – since these nodes poll network intensively over SNMP, UDP ports are sometimes reused for creating client sockets, thus these ports enter the L_1 list and are reported. We believe that if service syslog messages are correlated further, their number could be reduced several times and false positives could be eliminated.

4.3 Frequent pattern mining from IDS logs

As discussed in section 2, most data mining based algorithms for IDS log analysis have been developed for distinguishing important events from false positives and other background noise. However, in their recent works Viinikka, Debar, Mé et al. have argued that it is equally important to detect unanticipated changes in alarm flows (Viinikka, Debar, Mé and Séguier 2006; Viinikka, Debar, Mé, Lehikoinen and Tarvainen 2009). Since the algorithm presented in section 4.1 detects unexpected strong patterns from Netflow logs, we also propose this algorithm for IDS log analysis. In this section, we will briefly describe the experiment results for IDS logs.

Similarly with Netflow logs, after every W seconds the algorithm mines frequent closed patterns from the IDS log data of last W seconds. Patterns are both stored to file and used for creating a web report. Also, patterns which appear in less than K of last N pattern files are highlighted.

We have applied the algorithm for an IDS sensor of a large financial institution, with the sensor being deployed at the outer network perimeter. We measured the algorithm performance during 32 days, with the support threshold set to 10, W set to 3600 seconds, N set to 96 and K to 12. During the experiment, 5-187 patterns were detected (an average of 22.2 per window), and 0-175 patterns were highlighted (an average of 6.4 per window). Figure 2 presents some highlighted alert patterns (for the reasons of privacy, IP addresses have been obfuscated).

```
1:2009414 TCP 10.175.178.182 * 10.1.1.1 80
1:2001219 TCP 10.55.173.56 * * 22
1:474 ICMP 10.37.237.66 – 10.1.1.1 –
```

Figure 2: Sample highlighted IDS alert patterns

The first pattern reflects the Nkiller2 DOS attack from 10.175.178.182 against the company web server, while the second pattern indicates a horizontal SSH scan from 10.55.173.56. The third pattern corresponds to an ICMP echo scan flood from 10.37.237.66 against the company web server. During the experiments, we found that the algorithm is able to highlight many strong and unexpected attack patterns, and also provide a concise overview of latest attack trends for the security administrator.

5. Conclusion

In this paper, we have presented a study of important properties of IDS and Netflow data sets. We have also proposed several algorithms for IDS and Netflow log analysis.

For future work, we plan to employ statistical algorithms for measuring unexpected changes in supports of commonly occurring frequent alert and network traffic patterns. We also intend to elaborate the service detection algorithm and augment it with event correlation methods. In particular, we are considering the creation of Simple Event Correlator (Vaarandi 2006) rules for suppressing repeated service messages and for verifying with specifically crafted test packets if destination addresses are responding connection attempts. Finally, our research agenda includes work on workstation traffic anomaly detection, employing some of the methods described in this paper.

References

Al-Mamory, S.O., Zhang, H. and Abbas, A.R. (2008) "IDS Alarms Reduction Using Data Mining", *Proceedings of 2008 IEEE World Congress on Computational Intelligence*, pp. 3564-3570.

- Al-Mamory, S.O and Zhang, H. (2009) "Intrusion Detection Alarms Reduction Using Root Cause Analysis and Clustering", *Computer Communications*, vol. 32(2), pp. 419-430.
- Clifton, C. and Gengo G. (2000) "Developing Custom Intrusion Detection Filters Using Data Mining", *Proceedings of 2000 MILCOM Symposium*, pp. 440-443.
- Julisch, K. (2001) "Mining Alarm Clusters to Improve Alarm Handling Efficiency", *Proceedings of 2001 Annual Computer Security Applications Conference*, pp. 12-21.
- Julisch, K. (2003) "Clustering Intrusion Detection Alarms to Support Root Cause Analysis", *ACM Transactions on Information and System Security*, vol. 6(4), pp. 443-471.
- Julisch, K. and Dacier, M. (2002) "Mining intrusion detection alarms for actionable knowledge", *Proceedings of 2002 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 366-375.
- Li, X. and Deng, Z.-H. (2010) "Mining Frequent Patterns from Network Flows for Monitoring Network", *Expert Systems with Applications*, vol. 37(10), pp. 8850-8860.
- Long, J., Schwartz, D. and Stoecklin, S. (2006) "Distinguishing False from True Alerts in Snort by Data Mining Patterns of Alerts", *Proceedings of 2006 SPIE Defense and Security Symposium*, pp. 62410B-1--62410B-10.
- McHugh, J. and Gates, C. (2003) "Locality: A New Paradigm for Thinking About Normal Behavior and Outsider Threat", *Proceedings of 2003 New Security Paradigms Workshop*, pp. 3-10.
- Morin, B. and Debar, H. (2003) "Correlation of Intrusion Symptoms: an Application of Chronicles", *Proceedings of 2003 RAID Symposium*, pp. 94-112.
- Ning, P., Cui, Y. and Reeves, D. S. (2002) "Analyzing Intensive Intrusion Alerts via Correlation", *Proceedings of 2002 RAID Symposium*, pp. 74-94.
- Paredes-Oliva, I., Dimitritopoulos, X., Molina, M., Barlet-Ros, P. and Brauckhoff, D. (2010) "Automating Root-Cause Analysis of Network Anomalies using Frequent Itemset Mining", *Proceedings of 2010 SIGCOMM Conference*, pp. 467-468.
- Pietraszek, T. (2004) "Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection", *Proceedings of 2004 RAID Symposium*, pp. 102-124.
- Taylor, T., Paterson, D., Glanfield, J., Gates, C., Brooks, S. and McHugh, J. (2009) "FloVis: Flow Visualization System", *Proceedings of 2009 Cybersecurity Applications and Technology Conference for Homeland Security*, pp. 186-198.
- Treinen, J.J. and Thurimella, R. (2006) "A Framework for the Application of Association Rule Mining in Large Intrusion Detection Infrastructures", *Proceedings of 2006 RAID Symposium*, pp. 1-18.
- Vaarandi, R. (2004) "A Breadth-First Algorithm for Mining Frequent Patterns from Event Logs", *Proceedings of 2004 IFIP International Conference on Intelligence in Communication Systems*, pp. 293-308.
- Vaarandi, R. (2006) "Simple Event Correlator for real-time security log monitoring", *Hakin9 Magazine*, vol. 1/2006 (6), pp. 28-39.
- Vaarandi, R. (2008) "Mining Event Logs with SLCT and LogHound", *Proceedings of 2008 IEEE/IFIP Network Operations and Management Symposium*, pp. 1071-1074.
- Vaarandi, R. and Podiņš, K. (2010) "Network IDS Alert Classification with Frequent Itemset Mining and Data Clustering", *Proceedings of 2010 IEEE Conference on Network and Service Management*, pp. 451-456.
- Viinikka, J. and Debar, H. (2004) "Monitoring IDS Background Noise Using EWMA Control Charts and Alert Information", *Proceedings of 2004 RAID Symposium*, pp. 166-187.
- Viinikka, J., Debar, H, Mé, L., Lehtikoinen, A., and Tarvainen, M. (2009) "Processing intrusion detection alert aggregates with time series modeling", *Information Fusion Journal*, vol. 10(4), pp. 312-324.
- Viinikka, J., Debar, H., Mé, L., and Séguier, R. (2006) "Time Series Modeling for IDS Alert Management", *Proceedings of 2006 ACM Symposium on Information, Computer and Communications Security*, pp. 102-113.
- Wagner, A. (2008) *Entropy-Based Worm Detection for Fast IP Networks*, PhD Thesis, Swiss Federal Institute of Technology.
- Wagner, C., Wagener, G., State, R., Engel, T. and Dulaunoy, A. (2010) "Game Theory driven monitoring of spatial-aggregated IP-Flow records", *Proceedings of 2010 IEEE Conference on Network and Service Management*, pp. 463-468.