# A Framework of Event-Driven Detection System for Intricate Network Threats

**Rongmao Chen[1], Linbo Qiao[1], Bofeng Zhang[1], Zhenghu Gong[1]**

[1]Department of Computer National University of Defense Technology, Hunan Province, China

{chromao, qiao.linbo, bfzhang, gzh}@nudt.edu.cn

**Abstract -**As the network threats nowadays turn to be more intricate and diversiform, traditional intrusion detection methods are facing with the challenges of lacking flexibility because that they are just code-actual. This paper summarizes the common correlating features exhibited by the network events from the perspective of the detector, and proposes a detection framework which can be used to detect various network threats.After having a static scanning of the threats pattern library, it loads and initials the data structure of threat behaviors, and then utilizes the scheme of event driven to deal with the network event streams. Finally, it logs and calls the related function to query the threat behavior states. The formalization analysis shows that this framework has high flexibility and expansibility to adapt to the evolvement of network threat behaviors.

Index Terms –Intricate network threat, Detection Framework, Event-driven, Event aggregation

## 1. Introduction

With the rapid development of information technology, it is no longer uncommon to see that many aspects of current society have become more related to the computer network, which makes network attacks of more benefit. The motivation of network attack is also becoming benefit-driven, and the execution methods have developed from simple attack to distributed, cooperative and intricate attack modes, such as DDOS, worms, Botnet and so on[1,2,3], most of which have issued in heavy loss for the whole society. As the attack methods become integrated and steps-cooperative, the coming network threats turn to be more diversified, intricate and difficult to be detected. Therefore, to ensure the security of information system and healthy development of the society, it is of much significance for us to study deeply on the characteristic of intricate network threats, and propose a uniform detection framework of intricate network threats to enhance the accuracy of detection and automaticity of response.

Due to the intricacy and diversify of network threats, traditional detection systems are facing with the challenges from the intricate network threats. Most of them lack the flexibility in detecting framework, unification in threat modeling, scalability in detection methods and degree of automation in detection pattern generation. Based on the multi-stage cooperative characteristic of the intricate network threats behavior in the Internet environment, this paper proposes to summarize the common correlating features exhibited by the network events from the perspective of the detector, and establish a new uniform detection framework which would be able to load various monitoring strategies and identification methods according to the model of multi-level network events aggregation.

The reminder of this paper is organized as follows: the related work on detection system framework of network threats in Section 2. Section 3 presents the methods proposed in this paper in detail, overview of system, formation of event sequence, and formalization of event aggregation. Section 4 presents the formalization of the framework. Section 5 concludes this paper and discusses some points for future research.

## 2. Related Work

As network threat behaviors turn to be more and more intricate, the framework of detection system has developed from simple payload-matched IDS which is delegated by Snort [6] to distributed detection system with the ability of distributed monitoring and integrated identification.

Snort has been a classic tool for network threat detection for a long time. It detects the threat behaviors through matching the character string with the known abnormal signatures, which means that it cannot meet the detection requirement of intricate network threats any more. However, the good expansibility of snort rules makes it the criterion of many network events collection system, and the basic network events produced by Snort used to be the important elements of analyzing intricate network threats.

Based on the events collected by Snort, Gu et al, from TAMU in USA proposed a detection system for hosts in Botnet based on the behavior characteristic of Botnet, which is also called BotHunter[7]. It utilizes the detection framework based on correlation of certain events from Snort. Through using the Snort rules to load event monitoring algorithms, it identifies the bot host by collecting and correlating alerts from Snort, which means that it has a strong pertinence in event aggregation.

Another well-known intrusion detection system is called Bro[8],which is proposed by U.C.Berkeley. It also has the ability of collecting events which can be from either its analyzer or Snort and other applications. Bro is divided into an event engine that reduces a kernel-filtered network traffic stream into a series of higher-level events, and a policy script interpreter that interprets event handlers written in a specialized language used to express a site's security policy. Event handlers can update state information, synthesize new events, record information to disk, and generate real-time notification via system log.

Lee, from Georgia Institute of Technology, cooperated with some famous companies such as Google and IBM, proposing to build a cross-layers analysis system of network activities for internet security, which is called CLEANSE[9].It is proposed to analyze and restrain those network activities such as Botnet which does much harm to the Internet. CLEANSE has a more wide monitoring over the whole WAN. Apart from collecting those common events from various network exports, some basic events such as DNS activities and route updating are also collected for identification of threats.

In spite of the fact that there are many detection methods for various intricate network threats, most of them are coding-actualize and just concentrate on certain network attacks. They just cannot adapt to the evolutional network threats. It is urgent for us to propose a uniform detection framework, which can be easily adjusted to detection of different threats. There are some important facts that should be taken into account to propose a uniform detection framework.

(1) We need a uniform method that can be used to describe various threats and a new uniform detection framework which would be able to load various monitoring strategies and identification methods according to the certain threats.

(2) The detection framework should be flexible enough to adapt to multifarious updating network threats. As what was mentioned above, most of detection methods nowadays are just called 'method' because they are only for certain threats, and become disable when the threats evolve. We need a framework rather than a coding-actualize method.

During the detection of network threats, event is an important concept, which is the abstract of network activities related to the threat patterns. It is the pivotal element of a IDS, and the efficiency of events analysis makes a significant influence on the performance of IDS. Therefore, it is a pivotal problem that how to deal with network events and mining the network behaviors accurately that are contained in the event streams.

## 3. Framework of Event-Driven Detection System

This paper proposes a framework of network threats detection system. Unlike those coding-actualize ones, the framework purposes to carry out a uniform system that can be used to detect various network threats. It only needs some adjustment to adapt to the evolvement of network threat behaviors.

### *A. Overview of System*

The proposed detection system takes five steps to identify the network threat. After having a static scanning of the threats pattern library, it loads and initials the data structure of threat behaviors, and then utilizes the scheme of event driven to deal with the event stream. Finally, it logs and calls the related function to query the threat behavior states.

To be more clearly, as shown in Fig. 1, the whole process is as below,

**Step1:**Loading and parsing threat behavior patterns.

**Step2:**Capturing network traffic and delivering to events collection unit.

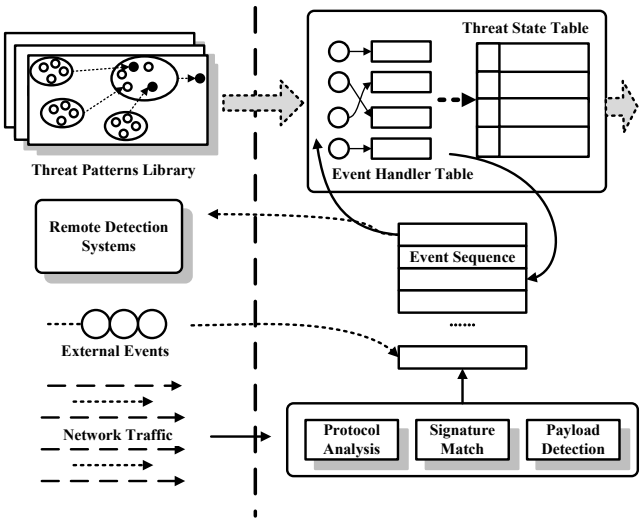


Figure.1 Framework of System

**Step3:**Analyzing traffic, generating events and delivering to the event sequence in classification.

**Step4:**Triggering handler process of events, updating event sequences and logging threat state.

**Step5:**Querying threat behavior states and triggering alarms and threat reports.

Network event uses several attributes to depict its data characteristics, which reflect some aspects of network activities. A representative network event has some main attributes including *Time Stamp*, *Duration of Online*, *Source Address*, *Destination Address*, *Protocol Type*(server port), *State Symbol* (SYN, FIN,PUSH and so on), *Sequence Number of TCP Packet*, *Available Size of Receiver Buffer* and so on. In this paper we define the event as a 6-tuple, each attribute of which has its signification as shown in Table I.

This paper divides events into two types, ***Atom Event*** and ***Advanced Event***. ***Atom Event*** refers to those events produced by protocol analysis or signature matching and some other predefine rules like that. The event generated through the aggregation of events (either ***Atom Event*** or ***Advanced Event***) is called ***Advanced Event***, which is proposed to describe high level network behaviour. In the rest of this paper, we use *AtEvent* and *AdEvent* to refer to the two types of event respectively.

Before starting to detect network threats, the system will load and parse threat behavior patterns, have a static scanning of the threat behavior library, initial the relation table of events and their handler functions as well as the global data structure which represents the threat states. The aim of this step is mainly to form the aggregation process, providing the rules for the event analysis in the further steps. There are many methods related to threat patterns description and also much work related to mining threat pattern from network traffic. One author of this paper proposed a mining method based on two-stage cluster[10],which can identify the attack serials from large-scale network traffic in high accuracy and efficiency. We can make use of this method to enrich the threat pattern library.

TABLE I Event Attributes and Corresponding Significations

| Attribute | Signification |
|---|---|
| *ID* | *Event ID* |
| *Time* | *Generate time of event* |
| *SrcIP* | *Source IP address* |
| *DstIP* | *Destination IP address* |
| *DPort* | *Destination port* |
| *SContent* | *Suspicious content* |

### *B. Formation of Event Sequence*

The function of event collection unit is to transfer the traffic to events and deliver them to the event sequences for the further analysis. There are many sources of events, one of which are mainly from the protocol analysis and signature matching. Protocol analyzer involves almost whole the protocols from network layer, transport layer and application layer as well as some uncommon protocol types. Signature matching aims at identifying those known attack, such as spacious requests. In order to realize cooperated detection, the system can also receive events from other systems located in other places of the same network. Apart from this, the source of event sequence can also be the advanced events generated from the handler of former events.

Most intricate network threats feature in cooperation as shown in Table II. In order to collect the events belong to the same threat together as much as possible, we need to take full consideration of the logic relationships between time, type and space of network events corresponded to the intricate threat behaviors, and then generate event sequences, each of which may correspond to a certain threat scenario. This will make the further analysis more precious.

This paper define the measure method of similarity between event $\boldsymbol{e_i}$ and $\boldsymbol{e_j}$,

$$Sim_{total}(e_i, e_j) = S_{time}(e_i, e_j) + S_{ip}(e_i, e_j) + S_{port}(e_i, e_j). \quad (1)$$

As it was shown above, the total similarity of events is the summary of *Time Similarity*, *IP Address Similarity* and *Port Similarity*. System will generate event sequences after receiving events form the bottom units. The generation algorithm of event is as shown in Fig. 2.

TABLE II Threat Feature and Corresponding Relation Feature

| Attack feature | Relation feature |
|---|---|
| *Order of attack steps* | *Time locally* |
| *Space locally* | *Address comparability* |
| *Protocol consistency* | *Port comparability* |

**Input:** Existing event sequences set $\boldsymbol{ES_{old}} = \{\boldsymbol{E_1}, \boldsymbol{E_2}, \boldsymbol{E_3}, \ldots \boldsymbol{E_k}\}$, a new event $\boldsymbol{e}$
**Output:** A new set of event sequences $\boldsymbol{ES_{new}}$

1: initial $\boldsymbol{i} = 0$,
2: **repeat**
3: initial $\boldsymbol{j} = 0$;
4: **repeat**
5: Compute the similarity between $\boldsymbol{e}$ and the $\boldsymbol{j}$ *th* event of $\mathbf{E}_i$
6: $\boldsymbol{j} = \boldsymbol{j} + \boldsymbol{1}$;
7: **until**( $\boldsymbol{j}$==| $\mathbf{E}_i$ | )
8: Record the similarity $\boldsymbol{S_i}$ which is of the most significant value;
9: $\boldsymbol{i} = \boldsymbol{i} + 1$;
10: **until** ($\boldsymbol{i}$ == | $\mathbf{ES_{old}}$ |)
11: Pick out the $\boldsymbol{E_n}$ which has the most significant similarity with $\boldsymbol{e}$;
12: **if** ( $\boldsymbol{S_n} > \varepsilon$ )
Add $\boldsymbol{e}$ into $\boldsymbol{E_n}$;
Update $\boldsymbol{ES_{old}}$
$\boldsymbol{ES_{new}} = \boldsymbol{ES_{old}}$;
13: **else** Generate a new $\boldsymbol{E_{k+1}}$ with $\boldsymbol{e}$; $\boldsymbol{ES_{new}} = \{ \boldsymbol{E_1}, \boldsymbol{E_2}, \boldsymbol{E_3}, \ldots \boldsymbol{E_k}, \boldsymbol{E_{k+1}}\}$;
14: Return $\boldsymbol{ES_{new}}$;

Figure.2 Algorithm of event generation

### *C. Formalization of Event Aggregation*

For the sake of describing threat behaviors accurately, several related events will be abstracted to a more advanced event when some conditions are met. We call this process as ***Event Aggregation***, and the aggregation of events define another new event, which represents a set of related events. As shown in Fig. 3, a set of related events can be aggregated to a new event, which can be also abstracted to a more advanced event with other related events. The nature of ***Event Aggregation*** is a process of abstracting network traffic to advanced events stage by stage. During the process, the original events come from the traffic, and then most of them are aggregated to advanced events according to the *Event Handler Table* loaded by the threat patterns library which is also a source of the event sequences. The final goal is to form an event that represents a whole threat scenario.

In order to describe the ***Event Aggregation*** in formalization, the system define the aggregation process as a 3-tuple,

$$< E, e_a, P > \quad (2)$$

in which,

(1) $E = \{e_1, \ldots e_k\}, \forall i \in [1,k], e_i \in (AtEvent \vee AdEvent)$

(2) $e_a \in AdEvent$

(3) $P = cond_1 \wedge cond_2 \wedge \ldots\ldots \wedge cond_j$

This 3-tuple means that $E$ will be aggregated into $e_a$, if the set of conditions $P$ is met.

The original threat patterns will not be able to describe the threat accurately due to the evolvement of threat behaviors. Therefore, the system needs to take into account the flexibility and support the expansibility of description mechanism of threat behaviors.

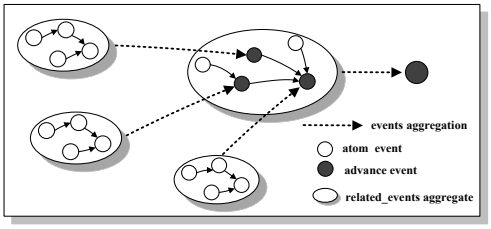


Figure.3 Aggregation of Events

Threat behaviors generally evolve by adding new threat features into the original one, which means that new events can be used to describe this change. As shown in Fig.4, based on the original events, the system can add some events to aggregate with the original events and finally represent the new threat. Through this way, the system can be flexible enough to adapt to multifarious updating network threats. This feature is also an improvement beside the existing methods.

As shown in Fig. 5, after getting an event from the sequence, the system will query the *Event Handler Table* and handle the event according to the defined process. For example, the system can count the total number of some event arising and generate another advanced event representing this phenomenon. We can count the event *http_request* and generate event *scan* if the number exceeds the threshold, which is a common event in many threat scenario. Every time the handle of events completes, the system will update the *Threat State Table*, which can trigger a threat report or an alarm. After the *Threat State Table* is updated, the system will query the current state of threats. It will log the threat report and generate an alarm when the condition of triggering is met.

## 4. Formalization Analysis

*A. Formalization Examples*

This part serves some threat detection cases to expatiate the work mechanism of the system framework. Using formalization analysis, four examples will give out below to have a more detailed validation.

***Scan Event*** Generally, attackers need to get the information about the targeted network through scanning before launch a attack. It usually achieves that goal by send a number of connection request, which is called event *ConnRq*. *ConnRq* can be several types of events, such as *HTTP Request, ICMP Ping, TCP ACK Ping, TCP SYN* and so on. Therefore, the aggregation process of *ConnRq* can be defined as shown in Fig. 6 (*Scan_In*) and Fig. 7(*Scan_Out*).

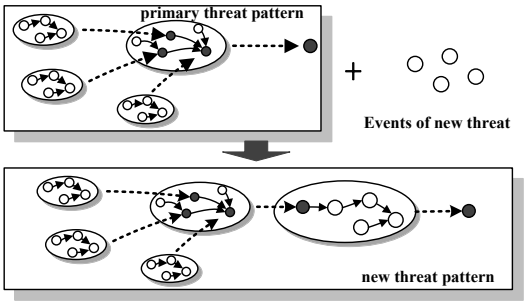


Figure.4 Expansibility of Events Aggregation

**Input:** A existing threat behaviors state $TS_{old}=\{T_1,T_2,T_3,\ldots T_k\}$, event $e$ from event sequence $E_k$
**Output:** A new state set $TS_{new}$

1: Call the handler of event $e$, generate advance event $e_a$
2: Add $e_a$ into $E_k$;
3: initial $i = 1$;
4: **repeat**
5: **if** ($e_a$ belong to $T_i$)
Update $T_i$;
Return $TS_{new}$ = Update $\{TS_{old}\}$;
6: **else** $i = i + 1$;
7: **until** ($i == |\ TS_{old}\ |$)
8: Record a new threat state $T_{k+1}$;
9: $TS_{new} = \{T_1,T_2,T_3,\ldots T_k,T_{k+1}\}$
10: Return $TS_{new}$;

Figure.5 Handle algorithm of event sequences

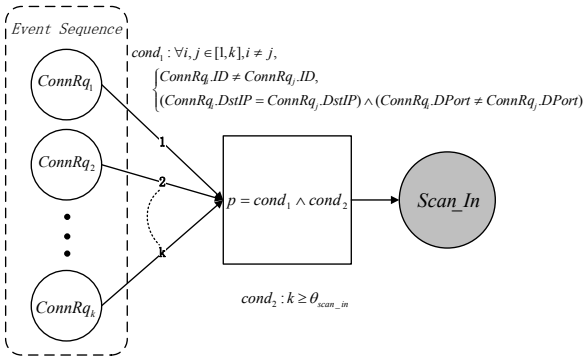


Figure.6 Formalization Example (1-1): Scan_In Event

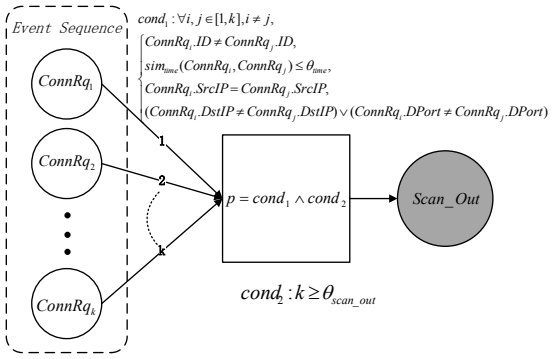


Figure.7 Formalization Example (1-2):Scan_Out Event

***Worm Event*** Worm is some malicious code which can spread through the network independently. It can infect other hosts by scanning and self-copy. As shown in Fig. 8, This event can be detected through signature matching.

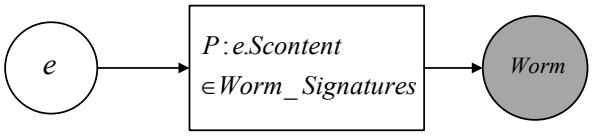


Figure.8 Formalization Example (2): Worm Event

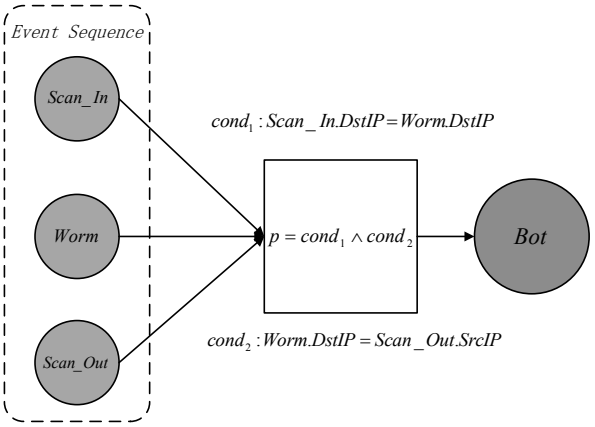


Figure.9 Formalization Example (3): Bot Event

***Bot Event*** Bot is referred to the puppet host controlled by attackers. It can receive commands, call the module and execute instructions to launch the attack. It can also try to expand Botnet by scanning out. Therefore, based on the aggregation process above, we can evolve the detection process of Bot,as shown in Fig. 9.

### *B. Analysis Conclusion*

Based on the analysis above, we can see that this method can be easily expanded to detect more intricate network threats. Some simple aggregation rule of network events can be combined to detect more intricate ones. This is the ultimate improvement than those code-actual ones. Therefore, this framework can be able to load various monitoring strategies and identification methods according to the network threat behaviors.

## 5. Discussion

This paper proposes a framework of detection system for intricate network threat. Comparing to the traditional methods, the framework has high flexibility and can be adjusted for new threats more easily. It concentrates in the policy adjustment rather than code-actualize.

But there is still considerable work to do. The framework is just an idea about the event-driven detection method. Therefore, it does not take other issues into consideration such as the aggregation algorithm efficiency, storage of event sequences and so on. In the future, the framework needs to be improved on these sides.

## Acknowledgment

The work was partially supported by the National High Technology Research and Development Program of China (863 Program) under Grant No.2011AA01A103, Aid Program of NUDT for Innovation Project of Excellent Graduate Student under Grant No.S120601, National Science Foundation of China (NSFC) under Grant No. 61003303, Program for Chang Jiang Scholar and Innovative Research Team in University (PCSIRT, No. IRT1012), Aid Program for Science and Innovative Research Team in Higher Education Institutions of Hunan Province: “Network Technology” and Hunan Provincial Natural Science Foundation of China under Grant No. 11JJ7003.

## References

[1] Symantec Inc. Symantec global Internet security threat report trends for 2009 volume XV. 2010.http://eval.symantec.com/mktginfo/enterprise/white_papers/bwhitepaper_internet_security_threat_report_xv04-2010.en-us.pdf.

[2] Georgia Tech Information Security Center. Emerging cyber threats report for 2013. October 7, 2012. http://www.gtisc.gatech.edu/pdf/cyberThreatReport 2013.pdf.

[3] Georgia Tech Information Security Center. Emerging cyber threats report for 2009. October 15, 2008. http://www.gtisc.gatech.edu/pdf/CyberThreatsReport 2009.pdf.

[4] Yang C, Robert H, Zhang J, Shin S, Gu G. Analyzing spammers' social networks for fun and profit -- A case study of cyber criminal ecosystem on Twitter. Proceedings of the 21st International World Wide Web Conference (WWW'12). April, 2012.

[5] Caballero J, Grier C, Kreibich C, Paxson V. Measuring Pay-Per-Install: The commoditization of malware distribution. Proc. USENIX Security Symposium, 2011.

[6] Snort homepage, 2013. http://www.snort.org/.

[7] Gu G, Porras P, Yegneswaran V, Fong M, and Lee W. BotHunter: Detecting malware infection through IDS-Driven dialog correlation. In Proc of USENIX Security, 2007.

[8] Paxson V. Bro: A system for detecting network intruders in real-time. Computer Network, 1999.

[9] J.Zhang, R.Perdisci, W.Lee , et al. Detecting stealthy P2P Botnets using statistical traffic fingerprints. In Proceedings of the 41st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Hong Kong, China, June 2011.

[10] Linbo Qiao, Bofeng Zhang, Zhiquan Lai, et al. Mining of Attack Models in IDS Alerts from Network Backbone by a Two-stage Clustering Method. IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum.Shanghai,China,2012:1257-1263.