Intrusion Detection Based on Spatiotemporal Characterization of Cyberattacks

Jiyeon Kim 1* and Hyong S. Kim 2

1 Center for Software Educational Innovation, Seoul Women’s University, Seoul 01797, Korea
2 Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA; kim@ece.cmu.edu
* Correspondence: jykim07@swu.ac.kr; Tel.: +82-2-970-2835

Received: 3 December 2019; Accepted: 29 February 2020; Published: 9 March 2020

Abstract: As attack techniques become more sophisticated, detecting new and advanced cyberattacks with traditional intrusion detection techniques based on signature and anomaly is becoming challenging. In signature-based detection, not only do attackers bypass known signatures, but they also exploit unknown vulnerabilities. As the number of new signatures is increasing daily, it is also challenging to scale the detection mechanisms without impacting performance. For anomaly detection, defining normal behaviors is challenging due to today’s complex applications with dynamic features. These complex and dynamic characteristics cause much false positives with a simple outlier detection. In this work, we detect intrusion behaviors by looking at number of computing elements together in time and space, whereas most of existing intrusion detection systems focus on a single element. In order to define the spatiotemporal intrusion patterns, we look at fundamental behaviors of cyberattacks that should appear in any possible attacks. We define these individual behaviors as basic cyberattack action (BCA) and develop a stochastic graph model to represent combination of BCAs in time and space. In addition, we build an intrusion detection system to demonstrate the detection mechanism based on the graph model. We inject numerous known and possible unknown attacks comprising BCAs and show how the system detects these attacks and how to locate the root causes based on the spatiotemporal patterns. The characterization of attacks in spatiotemporal patterns with expected essential behaviors would present a new effective approach to the intrusion detection.

Keywords: intrusion detection; spatiotemporal pattern; cyberattacks; cybersecurity

1. Introduction

Cyberattacks are becoming increasingly more sophisticated. For example, zero-day attacks exploit undisclosed vulnerabilities and advanced persistent threats (APT) attacks consist of multiple phases of attacks for a long period of time. With traditional intrusion detection systems based on signature and anomaly, it is challenging to detect these sophisticated attacks.

Signature-based intrusion detection systems (S-IDS) depend on known signatures to detect cyberattacks. There are two issues with S-IDSs. First, new attacks cannot be detected because new signatures are only obtained through post-analysis of attack events [1,2]. Even variant attacks are hard to detect as attackers work around the known signatures. Second, as the number of signatures increases, it is challenging to scale the detection mechanisms without impacting performance [3].

Anomaly-based IDSs (A-IDS) detect cyberattacks by comparing the system behavior with pre-defined normal behavior [2,4]. A-IDS can be effective for unknown attacks as it does not rely on known signatures. The major issue with A-IDS is the large number of false positives generated [5]. In simple applications, it is easy to define a normal behavior of the system. However, it is challenging to define a normal behavior in today’s complex applications running in an N-tier architecture with...
dynamic features [6]. These applications obfuscate normal behaviors and thus create much false positives with anomaly detection based on a simple outlier detection.

Machine learning (ML) techniques are being actively employed in anomaly detection as an alternative to these issues. ML-based anomaly detection trains historical datasets to define normal behaviors and detect outlier events as attacks [5, 7]. Although processing of massive datasets would help to set a flexible threshold of detection, there are still issues with false positives due to overfitting and unoptimized hyperparameters [8,9].

Most of existing S-IDS and A-IDS, including ML-based A-IDS, focus on a single computing or network element, whereas we focus on multiple elements. We use the terms element or host interchangeably to denote the computing or network element. Focusing on multiple elements in time and space rather than that of a single element would provide further evidence of an attack. Furthermore, this approach contributes to locate root causes by tracking the spatiotemporal behaviors.

In order to define the spatiotemporal attack patterns, we develop fundamental and essential behaviors that should appear in any attacks. We carefully study intrusion datasets as well as attack classifications including CAPEC [10] and characterize system and network features caused by intrusions. We define these behaviors of a single element as Basic Cyberattack Actions (BCAs).

BCAs allow detection of novel and complex cyberattacks as long as the attacks show any combination of BCA patterns. Future attacks could also consist of many combinations of BCAs. We propose to look at number of computing and network elements together in space (i.e., networked groups of hosts) and time rather than relying on individual BCA of a single element. Combination of BCAs describe the spatiotemporal characterization of an attack and would provide further insight into the attack. We also develop a stochastic graph model to represent the combination of BCAs.

In order to demonstrate our detection idea based on the spatiotemporal patterns, we develop an IDS in our production datacenter. We inject known and possible unknown attacks comprising BCAs and illustrate how the system detects these attacks and locates the root causes by tracking BCAs in time and space. The performance evaluation with extensive attacks comprising complex BCAs is not the focus of this paper and will be addressed in the forthcoming paper.

The remainder of this paper is organized as follows: We review related works in Section 2. Section 3 defines BCAs based on existing attack classifications. Section 4 defines a stochastic graph model to describe the behavior of BCA in time and space. In Section 5, we describe our BCA detection system. In Section 6, we evaluate our system with numerous attacks. Finally, the conclusions are presented in Section 7.

2. Related Work

S-IDSs detect signatures of known attacks. Kumar and Spafford [11] propose a pattern matching model for S-IDS based on Colored Petri Nets. Honeycomb [12] automatically generates attack signatures using a honeypot system and detects these signatures using pattern matching techniques. Josue et al. [13] propose a pattern matching algorithm to filter out the audit trail. Koral et al. [14] define a set of state transition signatures and detects an attack sequence of the transition. Zhengbing et al. [15] employ data mining techniques to develop more accurate signatures. These systems use known signatures and they are focused on improving the search and pattern matching speed. They do not consider unknown attacks without matching signatures.

A-IDSs define normal behaviors and detect outlier events as attacks. Although A-IDS is able to detect unknown attacks, it suffers from large numbers of false positives. Collaborative detection mechanisms are proposed to reduce false positives [16–20]. They aggregate and correlate a number of alerts generated by different IDSs. IDES [17] first proposes the IDS collaboration and EMERALD [18] refines IDES. Cuppens and Miege [16] used an expert system to develop an aggregation and correlation module. Valdes and Skinner [19] employed a probability-based approach for a similarity recognition. Yu et al. [20] develop a knowledge-based alert aggregation system. They collect a number of false alerts and process them based on correlation rules.
Numerous studies employ ML to identify legitimate behaviors. They define normal behavior patterns based on historical data from numerous system metrics. Bayesian network, decision Tree, and SVM (Support Vector Machine) are widely used in intrusion detection systems based on ML techniques. Kruegel et al. [21] propose an event classification scheme based on Bayesian network to mitigate false alarms. Bilge et al. [22,23] detect malicious domains by employing a passive DNS analysis based on a decision tree. Feng et al. [24], Kuang et al. [25], Thaseen and Kumar [26] apply SVM for better performance in intrusion detection. There are numerous studies that employ deep learning that belongs to ML. Khan et al. [27], Li et al. [28], Liu et al. [29] and Kim et al. [30] transform intrusion datasets to images and then detect attacks based on a convolutional neural network (CNN). Bonnetps et al. [31], Staudenmeyer and Omlin [32] suggest an IDS model based on a long short-term memory recurrent neural network (LSTM) using the KDD dataset [33]. There are further IDS studies that perform binary and multiclass classifications based on a recurrent neural network [30,34–36].

In addition, Dokas et al. [37], Hu and Panda [38] employ data mining techniques. Stephenson [39] combines forensics with the intrusion detection and response. Ren and Jin [40] develop a framework for the real-time intrusion forensic system.

Although numerous studies on S-IDS and A-IDS have been addressed, most of the studies focus on a single element. Our focus is on behaviors of multiple elements in time and space rather than that of a single element. As an existing study considering the concept of time and space, Chen et al. [41] identify spatiotemporal patterns of cyberattacks by analyzing victims’ IP addresses collected by HoneyPots. The biggest difference from our work is that they define every packet arriving at HoneyPots as attacks and analyze characteristics of attack traffic in order to predict cyberattacks, whereas our focus is on defining a novel method of detecting cyberattacks based on fundamental attack behaviors in time and space. They focus on the macroscopic characteristics of attack traffic and identify deterministic and stochastic patterns among a wide range of consecutive IP addresses. In addition, they only use IP addresses observed from the victim side, whereas we monitor not only the states of both attackers and victims but their spatiotemporal relationships.

3. Basic Cyberattack Action (BCA)

In order to detect an attack by looking at the number of computing and network elements together, we carefully study existing attack classifications as well as intrusion datasets. We focus on system and network characteristics by intrusions. We finally define BCAs, fundamental behaviors of attacks. BCAs observed from multiple elements naturally lend themselves to be described in space and time.

CAPEC [10] organizes more than 500 attack patterns employed to exploit vulnerabilities. CAPEC contains a comprehensive list with detailed information about each pattern. By analyzing CAPEC, we find that all attack patterns can be described with 10 essential methods of attack (MA) as shown in Table 1. Every attack pattern in CAPEC consists of some combination of MAs. We define five types of BCAs associated with relevant MA. In this work, we do not include MA10 as it depends on the human trust behavior during an attack. For example, CAPEC-98 (phishing attacks) trick people into offering access to their sensitive information. It deals with the human trust issue and it does not manifest in a particular system behavior that can be attributed to particular BCAs.

Table 1 also shows how each MA maps to BCAs. We analyze two types of intrusion dataset and as well as CAPEC to find out the mapping. Table 1 lists possible attacks corresponding to the mapping. The first intrusion dataset is KDD, the most widely-used dataset in intrusion detection. KDD classifies attacks into denial of service (DoS), remote-to-local (R2L), user to root (U2R) and Probing for IDS evaluation in the 1998 DARPA project. Numerous attacks belonging to the four classifications has been injected for dataset generation. The other one is CSE-CIC-IDS 2018 [42] that has been actively used in recent intrusion studies. CSE-CIC-IDS 2018 was generated by injecting 6 types of attack, such as brute force, DoS and botnet.

There are many proposed methods to detect MAs. In this work, we are mainly interested in BCAs and we could use any of these methods for MA detection. Focusing on common and fundamental features of cyberattacks rather than specific characteristics of each attack would become increasingly necessary to detect new and variant attacks.
| ID  | MA               | CAPEC (ID)                        | KDD            | CSE-CIC-IDS 2018                  | BCA               |
|-----|------------------|-----------------------------------|----------------|-----------------------------------|-------------------|
| MA1 | Flooding         | XML Ping of the death (147)       | DoS            | DoS, DoS, Botnet                 | BCA-1, BCA-4      |
| MA2 | Protocol         | HTTP attacks (33, 34, 105)         |                | DoS-Slowloris,                    | BCA-1, BCA-4      |
|     | manipulation     |                                   |                | SlowHTTPTest, Hulk               |                   |
| MA3 | Time and state   | Forced deadlock (25)               |                |                                   | BCA-1             |
|     |                  | Race condition (27, 29)            |                |                                   |                   |
| MA4 | API abuse        | Inducing account Lockout (2)       | U2R            |                                   | BCA-1             |
|     |                  | Buffer overflow (10, 24, 42, 46,47, 67) |                |                                   |                   |
| MA5 | Injection        | Command injection (136)            |                |                                   | BCA-1             |
|     |                  | SQL injection (108, 109)           |                |                                   |                   |
| MA6 | Analysis         | Port scanning (300-308)            | Probing        |                                   | BCA-3             |
|     |                  | Resource location spoofing (38, 132) | DoS-smurf      |                                   | BCA-5             |
| MA7 | Spoofing         |                                   |                |                                   |                   |
| MA8 | Brute force      | Password attack (16, 55, 70)       | R2L            | BruteForce-SSH, FTP, XSS         | BCA-2             |
| MA9 | Modification of  | Leverage alternate encoding (71)   |                |                                   | BCA-1             |
|     | resources        | Block access to libraries (96)    | U2R            |                                   |                   |
|     |                  |                                   |                |                                   |                   |
| MA10| Social engineering|                                    |                |                                   |                   |

The five types of BCA are characterized as follows:

- **BCA-1. Sudden performance degradation**
  
  Most attacks target to disrupt services offered by computing and network elements [43–45]. A sudden performance degradation would describe an essential behavior of affected hosts. This behavior can also occur in hardware and software failures. MA1 (Flooding), MA2 (Protocol manipulation), MA3 (Time and state), MA4 (API abuse), MA5 (Injection) and MA9 (Modification of resources) would result in sudden performance degradation. According to CAPEC, these MAs cause resource consumption, instability, crash, unexpected state, or execution logic change. Known attacks such as DoS [43,46–48], malware injection [43,46,49], buffer overflow [43,48], race condition [48], symbolic link [48], and a single point of failure [46] belong to BCA-1. Detecting attacks based on a single MA may lead to many false positives. However, BCA detection that combines several essential attack behaviors would decrease false positives significantly.

- **BCA-2. Iterative behavior**
  
  Many cyberattacks begin by obtaining an access to a target element. Most common method to obtain an access is through the brute force method of login trials with different passwords [43,46,48]. The resulting behavior is iterative access requests and corresponding responses. MA8 (Brute force) is based on a repetitive trial-and-error method. Known attacks such as login attempts [43] and authentication attacks [46,48] belong to BCA-2. This pattern manifests distinctively from common application requests and responses. Normal transactions in client-server systems do not exhibit this iterative behavior. Therefore, these iterative actions result in an essential attack behavior of a computing element.

- **BCA-3. Propagating behavior**
  
  Many attacks do not remain in a single target element. They tend to propagate to increase the number of infected hosts [43,50]. Attackers initially search for a vulnerable target. Once the target is infected by an attacker, the target becomes an attacker and starts propagating its search and infect tasks. This behavior is quite distinct from common application behavior. The resulting behavior is the increasing number of infected hosts as the time increases and such behavior translates to a spatiotemporal behavior of increasing infected elements. MA6 (Analysis) corresponds to the initial search such as probing [47] and scanning [43]. Known attacks such as worm [43,48] and port scanning [10] belong to BCA-3.
• **BCA-4. Sudden increase or decrease in ingress and egress traffic**

In addition to the performance degradation, the resulting behavior of attacks can be observed in either sudden increase or decrease of both ingress and egress traffic at the same time [51–54]. Usually the performance degradation would decrease the egress traffic corresponding to the responses of a server but the ingress traffic corresponding to the requests would remain the same. Decrease or increase in ingress and egress traffic usually result from malicious operation in computing or network elements. DDoS attacks [43,48] and flooding attacks [49] belong to BCA-4. In addition, BCA-4 could occur in combination with BCA-1 because this type of attacks could decrease the server performance.

• **BCA-5. Uncorrelated ingress and egress traffic**

We observe in any servers that the ingress traffic is highly correlated to the egress traffic. As the number of requests increases, we expect the number of responses to increase. This behavior is true when the server is working in desired operational range. As long as the server is capable of responding all requests immediately, we expect the number of responses to closely track the number of requests. When the server is congested or malfunctioning, the ingress and egress traffic are not correlated. Many attacks manifest in this uncorrelated ingress and egress traffic behavior. For example, when attackers spoof their identities during an attack, they do not receive any responses while it sends large numbers of requests [55]. Uncorrelated ingress and egress traffic would describe an essential behavior of a computing element with forged identity. In the existing works, masquerade [48] belongs to BCA-5. Figure 1 illustrates each BCA with the key behavioral features described above.

| BCA   | Description                                                                 | BCA   | Description                                                                 | Notation                                                                 |
|-------|-----------------------------------------------------------------------------|-------|-----------------------------------------------------------------------------|--------------------------------------------------------------------------|
| BCA-1 | ![Diagram showing performance degradation](image)                           | BCA-2 | ![Diagram showing attack behavior](image)                                  | $t_0$, initial time (normal)                                              |
|       | ![Diagram showing uncorrelated traffic](image)                             |       | $t_0$, attack-triggered time (abnormal), where $k \geq i$                    | $t_0$                                                                    |
| BCA-3 | ![Diagram showing request and response](image)                              | BCA-4 | ![Diagram showing ingress and egress traffic](image)                       | normal element                                                          |
|       | ![Diagram showing traffic volume](image)                                    |       | ![Diagram showing increasing traffic volume](image)                        | MA-detected element                                                      |
|       | ![Diagram showing traffic volume](image)                                    |       | ![Diagram showing decreasing traffic volume](image)                        | request response                                                        |
| BCA-5 | ![Diagram showing correlated and uncorrelated traffic](image)              |       | ![Diagram showing increasing traffic volume](image)                        | increasing traffic volume                                                |
|       | ![Diagram showing decreasing traffic volume](image)                         |       | ![Diagram showing decreasing traffic volume](image)                        | decreasing traffic volume                                                |

**Figure 1.** Key system behavior features of BCAs.
4. BCA Description and Composition

We now describe each BCA with associated MAs and spatiotemporal patterns. We use a stochastic graph model to describe the behavior of BCA in time and space. We define the stochastic graph model as follows. Table 2 shows the notation of the graph model.

Definition: A stochastic graph $G(t)$ represent the overall stochastic graph comprising of all elements $i$ at time $t$. $G(t)$ represents the stochastic graph only related to element $i$, where $i \in I$, thus a subset graph of $G(t) = |G(t)|$. $G(t)$ is modelled by 3-tuple of graph $G(t) = (V(t), E(t), \lambda(t))$. $V(t) = \{null, MA1, MA2, ..., MA9\}$ represent the state of the element $i$. If the element detects $MA3$, its state is $MA3$, for example. Null represent that no MA is detected and is operating normally. $E(t) = \{e_{i,j}\}$ where $i, j \in I$, is the set of edges that represent the communication between the node $i$ and $j$. $\lambda(t) = \{\lambda_{i,j}(t)\}$ is the set of traffic volume from node $i$ to all $j$. $\lambda_{i,j}(t)$, where $i, j \in I$. $\lambda_{i,j}(t)$ represent the stochastic random variable associated with $e_{i,j}$. $G(t)$ does not contain any vertices not connected to element $i$. BCAs can now be modelled using the stochastic graph $G$ as follows.

| Notation | Description |
|-----------|-------------|
| $G(t)$ | overall stochastic graph comprising of multiple elements at time $t$ |
| $G(t)$ | stochastic graph related to a single element $i$ at time $t$ |
| $V(t)$ | a set of states of a single element $i$ at time $t$ |
| $E(t)$ | a set of edges between a single element $i$ and other elements at time $t$ |
| $\lambda(t)$ | a set of traffic volume from a single element $i$ to other elements at time $t$ |
| $\lambda_{i,j}(t)$ | a stochastic random variable associated with an edge between a single element $i$ and $j$ |

- **BCA-1. Sudden performance degradation**

$$V(t) = \{MA1, MA2, MA3, MA4, MA5, MA9\}, |E(t)| \geq 0 \text{ and } \sum_{j} \lambda_{i,j}(t) \geq 0 \quad (1)$$

$MA1$ (Flooding), $MA3$ (Time and state), $MA4$ (API abuse), $MA5$ (Injection) and $MA9$ (Modification of resources) degrade performance of computing elements. $MA2$ (Protocol manipulation) would disrupt services offered by an application server. Host $i$ then has suddenly degraded performance. $E(t)$ and $\sum_{j} \lambda_{i,j}(t)$ could be zero or non-zero. BCA-1 is detected when host $i$ has sudden performance degradation.

- **BCA-2. Iterative behavior**

$$V(t) = \{MA8\}, |E(t)| \leq |E(t + \Delta)| \text{ and } \frac{\int_{t}^{t+w} \sum_{j} \lambda_{i,j}(t)}{w} \approx \frac{\int_{t+\Delta}^{t+\Delta+w} \sum_{j} \lambda_{i,j}(t + \Delta)}{w} \quad (2)$$

$MA8$ (Brute force) manifests in an iterative behavior. When host $i$ repeats the same behavior such as continuous login trials, the host would generate consistent traffic during the period of attack. $\Delta$ and $w$ denote the time period for the successive iteration and the window size for the traffic analysis respectively. During the password attack, iterative access requests and responses between the attacker and the target server generate consistent traffic volume. The password attack could target one or multiple servers. Brute force attacks may target different victims and the number of neighbors the stochastic graph would increase in time.

- **BCA-3. Propagating behavior**

$$V(t) = \{MA6\}, |E(t)| < |E(t + \Delta)| \text{ and } \sum_{j} \lambda_{i,j}(t) \leq \sum_{j} \lambda_{i,j}(t + \Delta) \quad (3)$$

In propagating attacks, an infected host $i$ becomes the attacker and starts infecting another host. Host $i$ would infect more and more hosts as time increases. Host $i$ keeps scanning other hosts $j$ to find vulnerable hosts. $MA6$ corresponds to the scanning behavior. The total volume does not play significant role here. Traffic from $i$ to all connected elements would usually increase but it is not necessary to show BCA-3 behavior.
• BCA-4. Sudden increase or decrease in ingress and egress traffic

\[ V(t) = \{MA1, MA2\}, |E_i(t)| \geq |E_i(t + \Delta)| \text{ and } \frac{d^2 \Sigma \lambda_{ij}(t)}{dt^2} > \alpha \text{ or } \frac{d^2 \Sigma \lambda_{ij}(t)}{dt^2} \leq \beta \quad (4) \]

In DDoS attacks, the traffic volume of both attackers and targets would suddenly increase exponentially. Host \( i \) under the DDoS attack results in \( \frac{d^2 \Sigma \lambda_{ij}(t)}{dt^2} > \alpha \). The traffic volume increases greater than the acceleration rate \( \alpha \). HTTP DoS attacks disrupt a web application server by depleting the web resources. Ingress and egress traffic of the server \( i \) would suddenly decrease exponentially. In this attack, \( \frac{d^2 \Sigma \lambda_{ij}(t)}{dt^2} \leq \beta \). The traffic volume decreases faster than rate \( \beta \). MA1 (Flooding) and MA2 (Protocol manipulation) are the essential methods for the DDoS and HTTP DoS attack, respectively. Usually multiple new hosts show up in the DDoS attack, but it is not necessarily required.

• BCA-5. Uncorrelated ingress and egress traffic

\[ V(t) = \{MA7\}, |E_i(t)| \geq |E_i(t + \Delta)| \text{ and } \lambda_i(t) = R_{ij}(\Sigma \lambda_{ij}(t), \sum \lambda_{ij}(t)) < \gamma \quad (5) \]

MA7 (spoofing) belongs to BCA-5. In IP spoofing attacks, the attacker does not receive any responses while it sends requests. This behavior results in uncorrelated ingress and egress traffic. When Host \( i \) spoofs its identity, \( \lambda_i(t) \) satisfies \( R_{ij}(\Sigma \lambda_{ij}(t), \sum \lambda_{ij}(t)) < \gamma \). \( R_{ij} \) is the cross correlation of ingress and egress traffic of \( i \). \( \gamma \) is a threshold coefficient of \( R_{ij} \) and \( 0 < \gamma < 1 \). Host \( i \) is hidden to other elements due to its spoofed IP address. As Host \( i \) is unknown to other elements during the attack, the number of its neighbor deceases.

5. BCA Detection System

Our system detects BCAs by monitoring spatiotemporal patterns according to the stochastic graph model. The spatiotemporal pattern describes the change of interactions among elements in time and space. There are many existing detection methods for MAs. We deploy any one of existing effective MA detection mechanisms. Periodically we generate a graph \( G_i \) for element \( i \). MAs are associated to \( G_i \) when they are detected for element \( i \). We match \( G_i \) against the stochastic graph models of BCAs to detect intrusions. We demonstrate the effectiveness of spatiotemporal patterns in detecting existing attacks as well as unknown attacks.

5.1. MA Detection

We apply existing mechanisms for MA detection in the host. Many of these mechanisms monitor system metrics and correlate metrics to detect a particular MA in a single computing or network element. We apply common MA detection mechanisms in the literature as shown in Table 3. \( S_i \) denotes the system metrics for MA detection mechanisms.

| Type                          | Metric            | MA                  | Reference |
|-------------------------------|-------------------|---------------------|-----------|
| System measurements           |                   |                     |           |
| \( S_1 \)                     | CPU usage         | MA1, MA3,          | [44,45]   |
| \( S_2 \)                     | Memory usage      | MA4, MA5,          |           |
| \( S_3 \)                     | Disk usage        | MA9                 |           |
| Network measurement          |                   |                     |           |
| \( S_4 \)                     | inbound traffic/sec| MA1                 | [51−54]   |
| \( S_5 \)                     | outbound traffic/sec |                   |           |
| \( S_6 \)                     | ratio of inbound and outbound traffic | MA7 |            |
| \( S_7 \)                     | Number of neighbors | MA6                | [50]      |
| Application measurement      |                   |                     |           |
| \( S_8 \)                     | Requests/sec      | MA2, MA8           | [53,56]   |
| \( S_9 \)                     | Responses/sec     | MA2                 |           |
| \( S_{10} \)                  | ratio of response over requests | MA2 |            |
| \( S_{11} \)                  | Response time     | MA2                 | [44,45,57,58] |
Our focus is not on performance of particular MA detection mechanisms but to demonstrate the advantage of BCA and their spatiotemporal patterns. Improvement in existing MA detection would improve our overall system. Again MA detection is limited to a single element and tends to have many false positives and false negatives.

5.2. BCA Detection

Individual host \( i \) monitors any change in MA, traffic volume, or temporal and spatial relationship among elements. Periodically its stochastic graph \( G_i \) is generated. The spatiotemporal pattern of \( G_i \) is then compared to BCA models. When there is a match between \( G_i \) and any of BCA models, we determine there is an intrusion and cyberattack to Host \( i \) and its associated elements.

Here is an example of BCA detection mechanism. Assume that host \( A \) generates \( G_A(t) \) as shown in Figure 2.

\[
\begin{align*}
V_A(t) &= \text{null} \\
E_A(t) &= \{e_{AB}, e_{AC}\} \\
A_A(t) &= \{\lambda_{AB}(t), \lambda_{AC}(t)\}
\end{align*}
\]

\[
\begin{align*}
V_A(t + \Delta) &= \text{MA} \\
E_A(t + \Delta) &= \{e_{AB}, e_{AC}, e_{AD}, e_{AE}\} \\
A_A(t + \Delta) &= \{\lambda_{AB}(t + \Delta), \lambda_{AC}(t + \Delta), \\
& \quad \lambda_{AD}(t + \Delta), \lambda_{AE}(t + \Delta)\}
\end{align*}
\]

Figure 2. Generation of \( G_A \)

We then proceed to match with BCA graphs. Assume that \( t_0 = t \) and \( t_1 = t + \Delta \). BCA-3 matches \( G_A(t_1) \) in the example as shown in Figure 3.

\[
\begin{align*}
\text{match} \quad & G_A(t_1) \\
V_A(t_1) &= \text{MA} \\
E_A(t_1) &= |E_A(t_0)| = 2, \quad |E_A(t_1)| = 4, \quad \text{and} \\
& |E_A(t_0)| < |E_A(t_1)| \\
A_A(t_1) &= \sum_{j \in \{B,C\}} \lambda_{A_j}(t_0) \leq \sum_{j \in \{B,C,D,E\}} \lambda_{A_j}(t_1)
\end{align*}
\]

Figure 3. BCA-3 detection of \( G_A \).
5.3. Combination of BCA Detections

The stochastic graph $G$ contains all elements with detected MAs. Each element carries out BCA detection through matching its own stochastic graph with BCA graphs. We then see all BCA detected elements collectively. If any of these graphs are connected, meaning that there is a connecting edge between these graphs, we consider the validity of given BCA detections by those elements. By considering multiple elements together, we reduce additional false positives by finding contradicting combination of BCAs. We also further reassure the accuracy of the detection by examining multiple elements. Here are examples to illustrate further reduction in false positives as well as improving detection accuracy. Figure 4 shows a worm attack.

Assume that hosts A-G detect BCA-3 at different times, $t_1$, $t_2$, and $t_3$. Each host generates $G_i(t)$, where $i \in \{A, B, \ldots, G\}$, according to BCA matching as shown in Figure 4. Once a host is infected by the worm attack, the host starts propagating the worm to other hosts continuously. Hosts A-G detect BCA-3 as the worm propagates in space and time. Assume that only one host detects BCA-3 and others do not detect any BCA patterns. There is no evidence of propagation and we determine that particular single BCA-3 detection has to be false positive. Combination of multiple $G_i(t)$ help us to reduce false positive. On the other hand, if there are multiple connected elements detecting BCA-3, then it confirms the propagating attack. Thus, $G(t)$ comprising of all $G_i(t)$ gives overall view of elements and helps to reduce false positives in many attack scenarios.

Figure 4. Combination of BCA-3.

Figure 5 shows another example of advantage of having more comprehensive $G(t)$. Host A guesses a B’s password using the brute force password attack. Host A sends login requests continuously until it finds out the correct password. During the attack, host B repeats the same
behavior to authenticate the passwords. In the password attack, the iterative behavior of either side of the host requires similar behavior from the other host. If only A or B detects BCA-2, we cannot definitely determine it as the brute force attack or a false positive. The combination of BCA-2 detected by A and B increases the confidence in detecting the attack.

5.4. Root Cause Analysis

Another advantage of using BCA graph is its ability to find possible root cause and location of the attack. The BCA graphs contain temporal and spatial relationship among elements. It is possible to trace the attack pattern to the originator using BCA graphs. Figure 6 shows an example of locating the root cause.

![Figure 6. Example of root cause analysis.](image)

At time $t_0$, host A, B, C, D and E are running normally in a multi-tier application. When C detects BCA-1 due to performance degradation, $E_C(t_1) - E_C(t_0) = \{e_{C,B}, e_{C,D}, e_{C,F}\}$. Only $e_{C,F}$ shows up at $t_1$ while other edges appeared at $t_0$. Host F would be the attacker who disrupts host C by injection.

6. Experimental Evaluation

We deploy several experiments in our datacenter with a controlled VM cluster. We evaluate our system’s performance in known attack and unknown attack detection. We also compare our system with those only relying on existing MA detection. We demonstrate how BCAs reduce false positives in several scenarios. We demonstrate that the spatiotemporal characterization of attack patterns helps in accuracy and reliability of intrusion and cyberattack detection. More extensive performance evaluation is not the focus of this paper and will be addressed in the forthcoming paper.

6.1. Experimental Setup

We implement our system and deploy in our production datacenter with a controlled VM cluster. We run a small agent in virtual machines (VMs). Each agent runs MA detection and BCA detection using its own stochastic graph. The agent creates its stochastic graph periodically. The agent then matches its graph to BCA graphs. When it finds the matching BCA, the agent sends an alarm along with its graph to the management server. The management server compiles graphs from all elements to generate and update overall stochastic graph, $G$. The management server then examines all
connected graphs $G_i$ to determine the attacks and possible root causes. The infrastructure for the experiments consists of the following components:

- Physical servers: Fedora 21, QEMU 1.6.2 hypervisor
- VM: Ubuntu 14.04 and Fedora 22, 1v CPU, 1024MB RAM
- Cloud web application: Rubbos application [59]

For the high reliability of the experimental evaluation, we deploy the Rubbos web application running in an N-Tier architecture. During attacks, web servers and database servers keep processing service requests from 100 clients on average per second.

6.2. Known Attacks

As shown in Table 4, we inject four known attacks selected by analyzing intrusion datasets as well as CAPEC as described in Section 3. We use released attack scripts as well as a penetration software for attack injection. Both Scenarios 1 and 4 detect multiple BCAs including BCA-4. Scenario 1 detects sudden decrease in traffic while scenario 4 detects sudden increase in traffic in BCA-4 detection.

| Scenario      | BCA 1 | BCA 2 | BCA 3 | BCA 4 | BCA 5 |
|---------------|-------|-------|-------|-------|-------|
| 1 Slowloris   | ✔     | ✔     |       | ✔     |       |
| 2 Password    |       | ✔     |       |       |       |
| 3 SSH worm    |       |       | ✔     |       |       |
| 4 Spoofed DDoS| ✔     |       | ✔     | ✔     | ✔     |

6.2.1. Scenario 1

Slowloris attack is a DoS attack targeting an application layer. The attacker modifies HTTP headers with wrong termination characters. The attacker then sends the packets to a web application server. This attack disrupts the web server due to a large number of incomplete open HTTP connections. The attacker consumes all connections on the server.

Existing HTTP DoS detection systems manually configure the web application parameters or set appropriate firewall rules to drop the suspicious packets [60]. Our system monitors the application metric $S_s$ (requests/s), $S_r$ (responses/s) and $S_{10}$ (ratio of requests and response). Here we have four hosts A, B, C and D as shown in Figure 7a. We deploy A (client), B (web server), and C (DB server) running a Rubbos application at t0. We inject the Slowloris attack into D using a released script [61]. Host D sends the modified HTTP requests (200 packets/s) to B.

- BCA detection
  $S_{10}$ (ratio of response over requests) indicates the performance of B for processing HTTP requests. When B is operating normally, the value of $S_{10}$ fluctuates from 1 to 4, as shown in Figure 7b. $S_{10}$ suddenly decreases when the attack is injected. $S_{10}$ decreases as $S_r$ (responses/s) suddenly decreases due to the performance degradation, as shown in Figure 7c. Host B detects MA2 (Protocol manipulation) based on $S_s$ and $S_{10}$. 

$S_{10}$
Host B detects BCA-1 and BCA-4 based on $G_B(t_1)$ as shown in Table 5. $B$ satisfies $E_B(t_1)$ and $A_B(t_1)$ as well as $V_B(t_1)$ for BCA-1 and BCA-4. Existing systems would also detect this attack by monitoring only MA2 using $S_8$, $S_9$ and $S_{10}$.

Table 5. Detection of BCA-1 and BCA-4 on host $B$ at $t_1$ in Scenario 1.

| $G_B(t_1)$ | Detection                        | BCA       |
|------------|----------------------------------|-----------|
| $V_B(t_1)$ | $V_B(t_1) = MA2$                 | BCA-1, BCA-4 |
| $E_B(t_0)$ | $E_B(t_0) = \{e_{B,A}, e_{B,C}\}$, $E_B(t_1) = \{e_{B,A}, e_{B,C}, e_{B,D}\}$, $|E_B(t_0)| = 2$, $|E_B(t_1)| = 3$ and $|E_B(t_3)|$ | BCA-1, BCA-2, BCA-3, BCA-4 |
| $A_B(t_1)$ | $\sum_{i\in\{A,C,D\}} A_{B,i}(t_1) \geq 0$ | BCA-1     |
| $\frac{d^2\sum_{i\in\{A,C,D\}} A_{B,i}(t_1)}{dt^2} > 0$ | BCA-4     |

Although our focus is not on detection methods of MAs, we analyze false positives in detecting MA2 for the validation. Because our attack scenario has 100 clients in the cloud application, we deploy 50 clients, 100 clients, and 200 clients without attacks. Table 6 shows the false positive rate (FPR) for $S_8$, $S_9$ and $S_{10}$, respectively.

Table 6. False positive rate of MA detection in Scenario 1.

| Metric | 50 Clients | 100 Clients | 150 Clients |
|--------|------------|-------------|-------------|
| $S_8$  | 8.13%      | 0.81%       | 0%          |
| $S_9$  | 7.31%      | 0.81%       | 0%          |
| $S_{10}$ | 0%        | 0%          | 0%          |

- **Root cause**

According to $G_B(t_1)$, only $e_{B,D}$ shows up at $t_1$ as $E_B(t_3) - E_B(t_0) = \{e_{B,D}\}$. The new element $D$ would be the attacker disrupting $B$’s service by protocol manipulation (MA2).
6.2.2. Scenario 2

We inject a password attack that tries guessing a victim’s password. In our experiment, we have two hosts, A and B as shown in Figure 8a. We inject the attack into A using Metasploit, a penetration software [62]. Metasploit is open-source software and allows us to inject a variety of attacks with our custom modules. Host A sends login requests more than 4000 times for 20 s guessing B’s password. Host B is a MySQL server.

- BCA detection

Our system monitors $S_b$ (requests/s) and $S_r$ (responses/s). From A’s perspective, $S_b$ shows the number of trials of guessing the password for one second. $S_r$ is the number of responses from the MySQL server, B. Host A and B detect MA8 (Brute force) according to large values of $S_b$ and $S_r$, as shown in Figure 8b.

![Figure 8. (a) Network structure of Scenario 2 and the BCA status of host A and B. (b) Experimental results of $S_b$ and $S_r$ on host A and B.](image)

Our system detects BCA-2 on both hosts from $G_\alpha(t_1)$ and $G_\beta(t_2)$. Host A and B have a new neighbor ($E_\alpha(t_1)$ and $E_\beta(t_2)$) and they generate very consistent traffic ($A_\alpha(t_1)$ and $A_\beta(t_2)$) during the attack. $G_\alpha(t_1)$ and $G_\beta(t_2)$ show that A and B satisfy all conditions for BCA-2 as shown in Table 7. Without MA detection ($V_\alpha(t_1)$ and $V_\beta(t_2)$), it could be either BCA-1 or BCA-2.

| $G_\alpha(t_1)$ | Detection | BCA |
|----------------|------------|-----|
| $V_\alpha(t_1) = MAB$ | | BCA-2 |

| $E_\alpha(t_1)$ | $E_\alpha(t_2) = \emptyset, \ |E_\alpha(t_2)| = 0, \ |E_\alpha(t_1)| = 1, \text{ and } |E_\alpha(t_0)| \leq |E_\alpha(t_1)|$ | BCA-1, BCA-2 |
|----------------|---------------------------------|----------------|
| $A_\alpha(t_1)$ | $\sum_{v_{ij}\in\{\}} A_{A,j}(t_2) \geq 0$ | BCA-1, BCA-3 |
| | $\frac{\sum_{v_{ij}\in\{\}} A_{A,j}(t_2)}{w} \approx \frac{t_{1}^{\alpha,w} A_{A,t_2}}{w}$ | BCA-2 |

| $G_\beta(t_2)$ | $V_\beta(t_2) = MAB$ | BCA-2 |

| $E_\beta(t_2)$ | $E_\beta(t_2) = \emptyset, \ |E_\beta(t_2)| = 0, \ |E_\beta(t_2)| = l, \text{ and } |E_\beta(t_0)| \leq |E_\beta(t_2)|$ | BCA-1, BCA-2 |
|----------------|-------------------------------------------------|----------------|
| $A_\beta(t_2)$ | $\sum_{v_{ij}\in\{\}} A_{B,j}(t_2) \geq 0$ | BCA-1, BCA-3 |
| | $\frac{\sum_{v_{ij}\in\{\}} A_{B,j}(t_2)}{w} \approx \frac{t_{2}^{\alpha,w} A_{B,t_2}}{w}$ | BCA-2 |

| $G_\beta(t_2)$ | | BCA-2 |
- **Associating MA improves detection capability of our system.** BCA-2 requires similar BCA-2 behavior from connected elements. The combination of BCA-2 detected by A and B in our system increases the confidence of correct detection. Existing systems that analyze elements independently could introduce many false positives.

- For MA detection using \( S_r \) and \( S_b \), we have no false positive found. This is because number of requests to the database server is less than 70 per second for all 50 clients, 100 clients, 150 clients in normal state. However, FPR could increase if the application has much more clients than 150 clients.

- **Root cause**

  According to \( G(t) \) comprising of \( G_A(t) \) and \( G_B(t) \), the new edge between \( A \) and \( B \) appears at \( t \) when \( A \) detects BCA-2. \( E_A(t_1) - E_A(t_0) = [e_{A,B}] \). \( A \) is more likely to be the attacker sending the login requests to \( B \), because \( A \) detects BCA-2 earlier than \( B \).

6.2.3. Scenario 3

We inject a worm spreading over a local network. An attacker infects a target via an SSH. The attacker usually uses known hosts file to collect target addresses and to bypass the authentication process. In our experiment, we deploy 10 hosts \( A \) to \( J \) which have all of other hosts’ credentials. We first inject the worm into \( A \) using Metasploit. Host \( A \) repeats infecting other hosts. Once a target host is infected by the worm, it becomes the attacker and starts infecting another host.

- **BCA detection**

  Our system monitors \( S_r \) (number of neighbors) to detect the worm. \( S_r \) refers to the number of trials to infect the worm via the SSH connection. Every host detects MA6 (Analyses) as \( S_r \)-increases as time increases as shown in Figure 9b. Our system detects MA6 when the number of neighbors \( (S_r) \) is greater than 5 (more than half of the entire hosts). Our system detects BCA-3 on every host based on \( G(t) \) where \( i \in \{ A, B, C, \ldots, J \} \). Table 8 shows an example of the BCA detection of \( G_A(t_1) \). Host \( A \) has new neighbors \( (E_A(t_1)) \). Traffic volume \( (A_A(t_1)) \), increases as the infection propagates through elements. \( G_A(t_1) \) matches all conditions of BCA-3. All hosts detect BCA-3 as the time increases. Overall \( G(t) \) consisting of multiple BCA-3 elements is consistent with the expected behavior of BCA-3 with propagating attacks. Again the overall view of all related hosts increases the confidence of correct detection in this scenario.

![Network Structure](image1.png)

*Figure 9. (a) Network structure of Scenario 3 and the status of BCA from host A to host J. (b) Experimental results of \( S_r \) on host A to host J.*
Table 8. BCA-3 detection on host A at t in Scenario 3.

| G_a(t) | Detection | BCA |
|--------|-----------|-----|
| V_a(t) | V_a(t1) = MA6 | BCA-3 |
| E_a(t1) | E_a(t2) = 0, E_a(t1) = [e_L, e_LH, e_LH, e_LH, e_LH], | BCA-1, BCA-2, BCA-3, BCA-4 |
|         | | BCA-3 |
| A_a(t)  | \(A_a(t) = \sum_{i} I_i(t_i) \cdot \Lambda_{a,i} \cdot R_{i,a} \geq 0\) | BCA-1, BCA-3 |
| G_a(t)  | | BCA-3 |

In order to analyze false positives in MA detection based on S_7, we monitor clients in our datacenter. Because the clients usually communicate with a web server, the number of neighbors is not proportional to the number of clients. In our experiment without attacks, the number of neighbors is less than 3 with a normal application running.

- **Root cause**

According to \(G(t)\), \(A\) and \(B\) first detect BCA-3 at \(t_1\), while other hosts detect BCA-3 at between \(t_2\) and \(t_3\). Either \(A\) or \(B\) could be the attacker that initiated the worm among the hosts.

6.2.4. Scenario 4

We inject a distributed SYN flooding attack with a spoofed IP address using Hping3 [63]. The attacker sends massive SYN packets to zombies with the victim’s IP address. The zombies then send SYN-ACK packets to the victim. The massive SYN-ACK packets deplete bandwidth of the victim. In this experiment, we deploy 6 hosts \((A-F)\) as shown in Figure 10a. \(A\) is the attacker. Host \(A\) keeps sending SYN packets to \(B-E\) with \(F\)’s IP address.

- **BCA detection**

Our system monitors \(S_4\) (inbound traffic/s), \(S_5\) (outbound traffic/s) and \(S_6\) (ratio of inbound and outbound traffic). In this experiment, \(S_4\) and \(S_6\) are used for MA1 (Flooding) detection. \(S_5\) is used to detect MA7 (Spoofing). Host \(A\) sends massive SYN packets but does not receive any responses during the attack. These SYN packets increase \(S_5\) as shown in Figure 10b and \(S_6\) increases accordingly. Host \(A\) detects MA1 and MA7 due to the high value of \(S_4\) and \(S_6\) respectively. Host \(F\) receives massive SYN-ACK packets from four hosts \((B-E)\). \(F\) detects MA1 due to the high value of \(S_4\), as shown in Figure 10c. In our experiment, the four hosts \((B-E)\) do not detect MA1 because each host does not meet the detection threshold (500 kb/s). The range of \(S_4\) and \(S_6\) are from 180 kb/s to 450 kb/s. The total amount of traffic going to \(F\) exceeds the threshold, thus host \(F\) detects MA1.

![Figure 10](image-url)

**Figure 10.** (a) Network structure of Scenario 4 and the BCA status of host \(A\) and \(F\).
(b, c) Experimental results of \(S_4\) and \(S_6\) on host \(A\) and \(F\).
Our system detects BCA-4 and BCA-5 as shown in Table 9. Host A detects BCA-5 as it has a low correlation between inbound and outbound traffic. Host A and F detect BCA-1 as their outbound and inbound traffic suddenly increase respectively. Both A and F match all conditions for BCA-4 and BCA-5. By combining BCA graphs, our system correctly detects not only the DDoS attack to F but also the spoofing attack from A.

Table 9. BCA detection in Scenario 4 (BCA-5 detection on host A at t1; detection of BCA-1 and BCA-4 on host A and F at t2).

| G(t) | Detection | BCA |
|------|-----------|-----|
| V_A(t1) | V_A(t1) = MA7 | BCA-5 |
| E_A(t1) | E_A(t2) = 0, | BCA-1, BCA-2, BCA-4, BCA-5 |
| A_A(t1) | A_J(t1) = 0. | BCA-5 |
| G_A(t1) | G_A(t1) = MA1 | BCA-1, BCA-4 |
| V_F(t2) | V_F(t2) = MA1 | BCA-1, BCA-4 |
| E_F(t2) | E_F(t2) = 0, | BCA-1, BCA-2, BCA-3, BCA-4 |
| A_F(t2) | A_J(t2) = 0. | BCA-1, BCA-4 |
| G_F(t2) | G_F(t2) = MA1 | BCA-1, BCA-4 |

For detection of MA1 and MA7, we have no false positive found until we deploy 150 clients with normal behaviors. In the application, the values of S1 and S: are less than 100 kb/s and 200 kb/s with 100 clients and 150 clients, respectively. In addition, S: has a value of at least 0.8 or higher with normal clients.

- Root cause
  After host A detects BCA-5 at t1, both A and F detect BCA-4 at t2. Based on G_A(t1) and G_F(t2), we find host A spoofs its identity and sends massive traffic. According to G_F(t2), F has new edges between F and the 4 hosts (B–E). We can infer that A initiated DDoS attack to F using B-E’s IP addresses.

6.3. Unknown Attack

We create an unknown attack based on the bait and switch method. It consists of a bait attack and the intended attack. The bait attack is designed to distract security managers’ attention away from the intended attack. The ultimate goal of this attack is to distribute malicious codes. We deploy 3 malicious hosts (A, B, C), 4 clients (D, E, F, G), two web servers (H, I), and one DB server (J). Figure 11 shows seven hosts (D–J) running normally in the multi-tier application at t0.
The unknown attack consists of three attacks as follows:

**Password attack (intended attack) at \( t_1 \):** This attack requires gaining access to the target server \( H \). The attacker employs a slow password attack to find host \( H \)’s password. The slow brute force attack is harder to detect using the existing brute force detection mechanisms. We inject the slow password attack into host \( B \) which is one of the malicious hosts. Host \( B \) repeatedly sends HTTP login requests to host \( H \) (web server) until it finds the correct password.

**Flooding attack (bait attack) at \( t_2 \):** The attacker employs a flooding attack to distract the security manager’s attention from the intended attack. We inject the flooding attack to malicious host \( A \). Host \( A \) starts sending large number of SYN packets to \( I \) in order to disrupt the server \( I \).

**Redirection attack (intended attack) at \( t_3 \):** After host \( B \) gains access to host \( H \) through slow password attack, host \( B \) controls host \( H \). Host \( B \) changes server \( H \)’s configuration to redirect all incoming requests to host \( C \) (malicious host) instead of intended DB server \( J \). When \( C \) receives requests from \( H \), \( C \) sends malicious codes as a response to all clients.

- **BCA detection**
  Figure 12 shows overall \( G(t) \) from our system when the unknown attack is injected.

**Password attack (intended attack) :** According to \( V_0(t_1) \), host \( B \) does not detect MA8 (Brute force) as shown in Table 10. Existing systems using metrics \( S_8 \) (requests/s) and \( S_9 \) (responses/s) would not detect the slow brute force attack. Our system detects the pattern of BCA-2 based on \( (E_0(t_1)) \) and \( (A_0(t_1)) \) at \( h \). Host \( B \) finds a new edge to the target \( H \) and generates consistent requests during the attack as shown in Figure 13. BCA-2 requires similar BCA-2 behavior in the related host in the traditional password attack detection. In Figure 12, host \( H \) does not detect BCA-2 unlike host \( B \). Host \( H \) fails to
detect a slow rate of login request attack embedded among normal application requests. Our system detects host B’s brute force behavior while existing systems fail to detect attacks on both B and H.

Figure 13. Iterative behavior of host B.

Table 10. BCA-2 detection on host B at t1 during a password attack.

| G(t) | Detection | BCA |
|------|-----------|-----|
| V_d(t1) V_p(t1) = null | | |
| E_d(t1) E_B(t0) = ∅, E_B(t1) = {e_B,H}, | [E_B(t1)] = 0, | [E_B(t1)] = 1 and | |
|                              | [E_B(t1)] ≤ | |
| A_d(t1) | \[\frac{1}{w} \sum_{t1}^{t2} t_{a_B}\] | BCA-2 |
| G_d(t1) | \[\frac{1}{w} \sum_{t1}^{t2} t_{a_B}\] | BCA-2 |

Flooding attack (bait attack): In the bait attack, host A and I detect MA1 (Flooding) due to high inbound (S1) and outbound (S5) traffic at t2. According to G_d(t2) and G_d(t2) in Table 11, these hosts have a new edge between them and have a sudden increase in traffic as shown in Figure 14. This flooding attack also results in the sudden decrease of traffic in host J as host I is disrupted by flooding attack. The security manager is distracted by host I being attacked by host A through the flooding.

Figure 14. Sudden increase and decrease of traffic by the flooding attack.
Furthermore, (MAs) contradicting essential on analyzing combination a removed to Redirection Electronics propagation and elements.

**Table 11. BCA-4 detection on host A, I and J at t2 during a flooding attack.**

| G(t)     | Detection | BCA       |
|----------|-----------|-----------|
| V_A(t2)  | V_A(t2) = MA1 | BCA-4     |
| E_d(t2)  | E_d(t0) = \emptyset, E_d(t2) = \{e_{IL}, \{E_d(t0)\} = 0, |E_d(t2)| = 1 and |E_d(t0)| \leq |E_d(t2)| | BCA-4     |
| A_d(t2)  | \(\frac{d^2 \sum_{t \in J} a_d(t2)}{dt^2} > 0\) | BCA-4     |
| G_d(t2)  | \(\frac{d^2 E_d(t2)}{dt^2}\) \(\text{BCA-4 (increase)}\) |          |
| V_J(t2)  | V_J(t2) = null | -         |
| E_J(t2)  | E_J(t0) = \{e_{IL, e_{IL, e_{IL}}}, E_J(t2) = \{e_{IL, e_{IL, e_{IL}}}, |E_J(t0)| = 2, |E_J(t2)| = 2 and |E_J(t0)| \leq |E_J(t2)| | BCA-4     |
| A_J(t2)  | \(\frac{d^2 \sum_{t \in E} a_J(t2)}{dt^2} < 0\) | BCA-4     |
| G_J(t2)  | \(\frac{d^2 G_J(t2)}{dt^2}\) \(\text{BCA-4 (decrease)}\) |          |

**Redirection attack (intended attack):** According to \(G(t)\) in Table 12, host J detects a removed edge between host H at t2. The removed edge triggers the detection of BCA-5. The removed edge belongs to application elements. In normal operation, we do not expect any application element to be removed without prior notification. Thus it further confirms the attack behavior. Host J also detects a low correlation between inbound and outbound traffic due to the flooding and redirection attacks.

**Table 12. BCA-5 detection on host J at t2 during a redirection attack.**

| G(t)     | Detection | BCA       |
|----------|-----------|-----------|
| V_J(t2)  | V_J(t2) = null | -         |
| E_J(t2)  | E_J(t0) = \{e_{IL, e_{IL}}\}, E_J(t2) = \{e_{IL, e_{IL}}\}, |E_J(t0)| = 2, |E_J(t2)| = 2 and |E_J(t0)| \geq |E_J(t2)| | BCA-5     |
| A_J(t2)  | R_J \{\sum_{t = 1}^{n} a_J(t2)\} < \sum_{t = 1}^{n} a_J(t2) < 0.1 | BCA-5     |
| G_J(t2)  | \(\frac{d^2 G_J(t2)}{dt^2}\) \(\text{BCA-5}\) |          |

Overall \(G(t)\) graph indicates high possibility of the redirection attack based on other connected BCA detections. Our system correctly detects not only the bait attack but the intended attack where existing systems fail to detect the intended attack.

**7. Conclusions**

We have presented a different perspective on ways to detect cyberattacks. Rather than relying on traditional signatures and anomaly patterns, we proposed an approach based on fundamental and essential behaviors of cyberattacks. We defined these behaviors as Basic Cyberattack Action (BCA) and proposed five types of BCA such as a sudden performance degradation, iterative behavior, propagation behavior, sudden increase or decrease in ingress and egress traffic, and uncorrelated ingress and egress traffic. Individual BCA is detected by monitoring not only Methods of Attack (MAs) and traffic volume of a single element, but also the spatiotemporal relationship among elements. In order to represent combination of BCAs, we developed a stochastic graph model. The combination of BCAs describes the change of interactions among elements in time and space. By considering multiple elements together, we can reduce additional false positives by finding contradicting combination of BCAs. We also implemented and deployed our spatiotemporal-based intrusion detection system in our datacenter for preliminary validation of our idea. We demonstrated the effectiveness of BCAs in numerous known and unknown attack scenarios. For known attacks, we injected a Slowloris attack, password attack, SSH worm attack, and Smurf attack selected by analyzing intrusion datasets and CAPEC. Our experimental results showed that our system accurately detects all the known attacks comprising BCAs and locates possible root cause as well. Furthermore, we built an example of unknown attack based on a bait-and-switch method.
combines three types of attacks such as a password attack, flooding attack, and redirection attack. The experimental results showed that such unknown attack is effectively detected by our system while existing detection mechanisms fail to detect the intended attack. Many existing systems may not be adequate for future unknown and advanced attacks. In addition, today’s complex applications may trigger a significant number of false positives. We believe that the characterization of attacks in spatiotemporal patterns with expected essential behaviors of any attack presents a new effective approach to the intrusion detection. The performance evaluation with not only extensive attacks comprising complex BCAs but a variety of applications will be addressed in the future.

**Author Contributions:** Conceptualization, J.Y. and H.S.; methodology, J.Y. and H.S.; implementation and experiments, J.Y.; writing—J.Y. and H.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1D1A1B07050543).

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**

1. Mell, P.M.; Lippmann, R.; Hu, C.T.; Haines, J.; Zissman, M. An Overview of Issues in Testing Intrusion Detection Systems; NIST Interagency/Internal Report (NISTIR)-7007; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2003.
2. McCarthy, J.; Powell, M.; Stouffer, K.; Tang, C.; Zimmerman, T.; Barker, W.; Ogunyale, T.; Wynne, D.; Wiltberger, J. Securing Manufacturing Industrial Control Systems: Behavioral Anomaly Detection; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2018.
3. Carter, E.; Hogue, J. Intrusion Prevention Fundamentals; Cisco Press: Indianapolis, IN, USA, 2006.
4. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. ACM Comput. Surv. (CSUR) 2009, 41, 1–58.
5. Garcia-Teodoro, P.; Díaz-Verdejo, J.; Maciá-Fernández, G.; Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. Comput. Secur. 2009, 28, 18–28.
6. Diffily, S. The Website Manager’s Handbook; Lulu: Morrisville, NC, USA, 2006.
7. Sommer, R.; Paxson, V. Outside the closed world: On using machine learning for network intrusion detection. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA, USA, 16–19 May 2010; pp. 305–316.
8. Trinh, V.-V.; Tran, K.P.; Huong, T.T. Data driven hyperparameter optimization of one-class support vector machines for anomaly detection in wireless sensor networks. In Proceedings of the 2017 International Conference on Advanced Technologies for Communications (ATC), Quy Nhon, Vietnam, 18–20 October 2017; pp. 6–10.
9. Ikram, S.T.; Cherukuri, A.K. Intrusion detection model using fusion of chi-square feature selection and multi class SVM. I. King Saud Univ. -Comput. Inf. Sci. 2017, 29, 462–472.
10. MITRE. Common Attack Pattern Enumeration and Classification. Available online: http://capec.mitre.org (accessed on 5 March 2020).
11. Kumar, S.; Spafford, E.H. A Pattern Matching Model for Misuse Intrusion Detection; Technical Report CSD-TR-94-071; Purdue University: West Lafayette, IN, USA, 1994.
12. Kreibich, C.; Crowcroft, J. Honeycomb: Creating intrusion detection signatures using honeypots. Acm Sigcomm Comput. Commun. Rev. 2004, 34, 51–56.
13. Kuri, J.; Navarro, G.; Mé, L.; Heye, L. A pattern matching based filter for audit reduction and fast detection of potential intrusions. In International Workshop on Recent Advances in Intrusion Detection; Springer: Berlin/Heidelberg, Germany, 2000; pp. 17–27.
14. Ilgun, K.; Kemmerer, R.A.; Porras, P.A. State transition analysis: A rule-based intrusion detection approach. IEEE Trans. Softw. Eng. 1995, 3, 181–199.
15. Hu, Z.; Li, Z.; Wu, J. A novel Network Intrusion Detection System (NIDS) based on signatures search of data mining. In Proceedings of the 1st international Conference on Forensic Applications and Techniques in Telecommunications, information, and Multimedia and Workshop, Adelaide, SA, Australia, 23–24 January 2008; pp. 45.
16. Cuppens, F.; Miege, A. Alert correlation in a cooperative intrusion detection framework. In Proceedings of the 2002 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 12–15 May 2002; pp. 202–215.

17. Lunt, T.F. IDES: An intelligent system for detecting intruders. In Proceedings of the Symposium: Computer Security, Threat and Countermeasures; Purdue University: West Lafayette, IN, USA, 1990; pp. 30–45.

18. Porras, P.A.; Neumann, P.G. EMERALD: Event monitoring enabling response to anomalous live disturbances. In Proceedings of the 20th National Information Systems Security Conference, Baltimore, MD, USA, 7–10 October 1997; pp. 353–365.

19. Valdes, A.; Skinner, K. Probabilistic alert correlation. In International Workshop on Recent Advances in Intrusion Detection; Springer: Berlin/Heidelberg, Germany, 2001; pp. 54–68.

20. Yu, J.; Reddy, Y.V.R.; Selliah, S.; Reddy, S.; Bharadwaj, V.; Kankanahalli, S. TRINETR: An architecture for collaborative intrusion detection and knowledge-based alert evaluation. Adv. Eng. Inform. 2005, 19, 93–101.

21. Kruegel, C.; Mutz, D.; Robertson, W.; Valeur, F. Bayesian event classification for intrusion detection. In Proceedings of the 19th Annual Computer Security Applications Conference, 2003. Proceedings, Las Vegas, NV, USA, 8–12 December 2003; pp. 14–23.

22. Leyla, B.; Engin, K.; Christopher, K.; Marco, B. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In Proceedings of the NDSS 2011, 18th Annual Network and Distributed System Security Symposium, San Diego, CA, USA, 6–9 February 2011; pp. 1–17.

23. Bilge, L.; Sen, S.; Balzarotti, D.; Kirda, E.; Kruegel, C. Exposure: A passive dns analysis service to detect and report malicious domains. ACM Trans. Inf. Syst. Secur. (TISSEC) 2014, 16, 14.

24. Feng, W.; Zhang, Q.; Hu, G.; Huang, J.X. Mining network data for intrusion detection through combining SVMs with ant colony networks. Future Gener. Comput. Syst. 2014, 37, 127–140.

25. Kuang, F.; Xu, W.; Zhang, S. A novel hybrid KPCA and SVM with GA model for intrusion detection. Appl. Soft Comput. 2014, 18, 178–184.

26. Thaseen, I.S.; Kumar, C.A. Intrusion detection model using fusion of PCA and optimized SVM. In Proceedings of the 2014 International Conference on Contemporary Computing and Informatics (ICSI), Mysore, India, 27–29 November 2014; pp. 879–884.

27. Khan, R.U.; Zhang, X.; Alazab, M.; Kumar, R. An Improved Convolutional Neural Network Model for Intrusion Detection in Networks. In Proceedings of the 2019 Cybersecurity and Cyberforensics Conference (CCC), Melbourne, Australia, 8–9 May 2019; pp. 74–77.

28. Li, Z.; Qin, Z.; Huang, K.; Yang, X.; Ye, S. Intrusion detection using convolutional neural networks for representation learning. In Proceedings of the International Conference on Neural Information Processing, Guangzhou, China, 14–18 November 2017; pp. 858–866.

29. Liu, Y.; Liu, S.; Zhao, X. Intrusion Detection Algorithm Based on Convolutional Neural Network. In Proceeding of the 4th International Conference on Engineering Technology and Application, Nagoya, Japan, 29–30 June 2017. pp. 9–13.

30. Kim, J.; Shin, Y.; Choi, E. An Intrusion Detection Model based on a Convolutional Neural Network. J. Multimed. Inf. Syst. 2019, 6, pp. 165–172.

31. Bontemps, L.; Cao, V.L.; Mcdermott, J.; Le-Khac, N.-A. Collective anomaly detection based on long short-term memory recurrent neural networks. In Proceedings of the International Conference on Future Data and Security Engineering, Can Tho City, Vietnam, 23–25 November 2016; pp. 141–152.

32. Staudemeyer, R.C.; Omlin, C.W. Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data. In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, East London, South Africa, 7–9 October 2013; pp. 218–224.

33. KDD CUP 1999 Data. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 7 February 2020).

34. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. IEEE Access 2017, 5, 21954–21961.

35. Kim, J.; Kim, H. An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization. In Proceeding of the 2017 IEEE International Conference on Platform Technology and Service (PlatCon), Busan, South Korea, 13–15 February 2017; pp. 1–6.
36. Almiani, Muder, et al. Deep recurrent neural network for IoT intrusion detection system. *Simul. Model. Pract. Theory* 2019, 102031, doi:10.1016/j.simpat.2019.102031.
37. Dokas, P.; Ertoz, L.; Kumar, V.; Lazarevic, A. Data mining for network intrusion detection. In Proceedings of the NSF Workshop on Next Generation Data Mining, Marriott, Inner Harbor, Baltimore, 1-3 November 2002; pp. 21–30.
38. Hu, Y.; Panda, B. A data mining approach for database intrusion detection. In Proceedings of the 2004 ACM Symposium on Applied Computing, Nicosia, Cyprus, 14–17 March 2004; pp. 711–716.
39. Stephenson, P. The application of intrusion detection systems in a forensic environment. In Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection (RAID), Toulouse, France, 2–4 October 2000.
40. Ren, W.; Jin, H. Distributed agent-based real time network intrusion forensics system architecture design. In Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA’05), Taipei, Taiwan, 28–30 March 2005; pp. 177–182.
41. Chen, Y.-Z.; Huang, Z.-G.; Xu, S.; Lai, Y.-C. Spatiotemporal patterns and predictability of cyberattacks. *PLoS ONE* 2015, 10, e0131501.
42. CSE-CIC-IDS2018 on AWS. Available online: https://www.unb.ca/cic/datasets/ids-2018.html (accessed on 7 February 2020).
43. eCSIRT. WP4 Clearinghouse Policy—Release 1.2. Available online: http://www.eccsirt.net/cec/service/documents/wp4-clearinghouse-policy-v12.html (accessed on 5 March 2020).
44. Falkenberg, A.; Mainka, C.; Somorovsky, J.; Schwenk, J. A new approach towards DoS penetration testing on web services. In: 2013 IEEE 20th International Conference on Web Services, Santa Clara, CA, USA, 28 June–3 July 2013; pp. 491–498.
45. Mirkovic, J.; Hussain, A.; Fahmy, S.; Reiter, P.; Thomas, R.K. Accurately measuring denial of service in simulation and testbed experiments. *IEEE Trans. Dependable Secur. Comput.* 2008, 6, 81–95.
46. Iqbal, S.; Kiah, M.L.M.; Daghighi, B.; Hussain, M.; Khan, S.; Khan, K.; Choo, K.K.R. On cloud security attacks: A taxonomy and intrusion detection and prevention as a service. *J. Netw. Comput. Appl.* 2016, 74, 98–120.
47. Lippmann, R.; Haines, J.W.; Fried, D.J.; Korba, J.; Das, K. The 1999 DARPA off-line intrusion detection evaluation. *Comput. Netw.* 2000, 34, 579–595.
48. Simmons, C.; et al. AVOIDIT: A cyber attack taxonomy. In Proceedings of the 9th Annual Symposium on Information Assurance (ASIA’14), Albany, NY, USA, 3–4 June 2014; pp. 2–12.
49. Islam, T.; Manivannan, D.; Zeadally, S. A classification and characterization of security threats in cloud computing. *Int. J. Next-Gener. Comput.* 2016, 7, pp. 1–17.
50. Stafford, S.; LI, J. Behavior-based worm detectors compared. In *International Workshop on Recent Advances in Intrusion Detection*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 38–57.
51. Bogdanoski, M.; Suminoski, T.; Risteski, A. Analysis of the SYN flood DoS attack. *Int. J. Comput. Netw. Inf. Secur. (IJCNIS)* 2013, 5, 1–11.
52. Rana, D.S.; Garg, N.; Chamoli, S.K. A Study and Detection of TCP SYN Flood Attacks with IP spoofing and its Mitigations. *Int. J. Comput. Technol. Appl.* 2012, 3, 1476–1480.
53. Shea, R.; Liu, J. Performance of virtual machines under networked denial of service attacks: Experiments and analysis. *IEEE Syst. J.* 2012, 7, 335–345.
54. Siaterlis, C.; Maglatis, V. Detecting incoming and outgoing DDoS attacks at the edge using a single set of network characteristics. In Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC’05), Murcia, Spain, 27–30 June 2005; pp. 469–475.
55. Templeton, S.J.; Levitt, K.E. Detecting spoofed packets. In Proceedings of the DARPA Information Survivability Conference and Exposition, Washington, DC, USA, 22–24 April 2003; pp. 164–175.
56. Grill, Martin, et al. Detecting DGA malware using NetFlow. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015; pp. 1304–1309.
57. Sachdeva, Monika, et al. Performance analysis of web service under DDoS attacks. In Proceedings of the 2009 IEEE International Advance Computing Conference, Patiala, India, 6–7 March 2009; pp. 1002–1007.
58. Kim, J.; Kim, H.S. PBAD: Perception-based anomaly detection system for cloud datacenters. In Proceedings of the 2015 IEEE 8th International Conference on Cloud Computing, New York, NY, USA, 27 June–2 July 2015; pp. 678–685.
59. RuBBoS. Available online: http://jmob.ow2.org/rubbos.html (accessed on 5 March 2020).
60. Moustis, D.; Kotzanikolaou, P. Evaluating security controls against HTTP-based DDoS attacks. In Proceedings of the IISA 2013; pp. 1–6.
61. Slowloris. Available online: https://github.com/OGglas/Original-Slowloris-HTTP-DoS/blob/master/slowloris.pl (accessed on 5 March 2020).
62. Metasploit. Available online: https://www.metasploit.com/ (accessed on 5 March 2020).
63. Hping3. Available online: https://tools.kali.org/information-gathering/hping3 (accessed on 5 March 2020).

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).