Data Integrity Protection through Security Monitoring for Just-in-Time News Feeds

Abdulrahman Althobaiti, Prasad Calyam, Ravi Akella, Prabhu Vallabhaneni
University of Missouri-Columbia, USA
{amadq7, calyamp, raxv8, pvqbc}@missouri.edu

Abstract—There has recently been a major shift in news related media consumption trends and readers are increasingly relying on just-in-time news feeds versus the traditional newspaper print medium. Cloud-networked infrastructures are being setup by the media companies to aggregate news feeds from affiliates, and to meet the elastic demands of Internet-scale users accessing news feeds. However, cyber attacks could compromise these just-in-time news feed services and hackers could particularly launch data integrity as well as denial-of-service attacks that: (a) tarnish the reputation of media companies and (b) impact the service availability for users. In this paper, we describe data integrity and availability checking techniques to protect just-in-time news feed services against cyber attacks in use cases such as: (a) “Data-in-Motion” - when obtaining just-in-time news feeds (e.g., RSS feeds) from affiliates and (b) “Data-at-Rest” - when compiled news feeds reside within cloud-networked infrastructure for real-time premium subscriber access. Using concepts of distributed trust and anomaly detection and a realistic testbed environment in the DeterLab infrastructure, we show the impact of the different cyber attacks and propose solutions to defend against them.

I. INTRODUCTION

Nowadays, reading online news through just-in-time news feeds is a regular routine for most people versus relying on the traditional newspaper print medium. Online news feeds provide fast, real-time and sharable news content that is accessible through mobile devices and subscription with popular standards-based mechanisms such as Really Simple Syndication (RSS) feeds [1]. Premium news content is being served by media companies such as Thomson Reuters, Dow Jones and Bloomberg, who service subscribers willing to pay for tailored ‘real-time’ news, particularly in sectors such as Finance and Health Care. Cloud-networked infrastructures are being setup by the media companies to aggregate news feeds from affiliates and to meet the elastic demands of Internet-scale users accessing news feeds.

Given the fact that just-in-time news feeds are rapidly being consumed widely, the cyber attack threats are also proportionately increasing to exploit vulnerabilities in the news feed services. In fact, the statistics collected by Kaspersky Labs [3] show that malicious URLs comprise over 93% of all cyber attacks monitored. Such cyber attacks could seriously compromise just-in-time news feed services and hackers could particularly launch data integrity attacks, malware attacks as well as denial-of-service attacks [4] that: (a) tarnish the reputation of media companies for whom credibility is a vital survival factor and (b) impact the service availability for users who rely on news feeds for important decision making activities. Particularly, if a business news article dealing with stock market trends is tampered, it could lead to huge financial losses for many individuals and also results in the doom of that news agency. Hence, it is critical to analyze instances of possible cyber-crime and provide mechanisms to detect and mitigate the emerging threats to services for just-in-time news feeds hosted in cloud-networked environments.

In this paper, we investigate techniques to monitor and protect just-in-time news feed services against cyber attacks in use cases such as: (a) “Data-in-Motion” - when obtaining just-in-time news feeds (e.g., RSS feeds) from affiliates and (b) “Data-at-Rest” - when compiled news feeds reside within cloud-networked infrastructure for real-time premium subscriber access. Using concepts of distributed trust and anomaly detection, we study the impact of the different cyber attacks on data integrity of just-in-time news feed services. We implement a DIPS-Mon prototype system that assigns “trust scores” for ranking news feeds collected (i.e., stored in a database) from different affiliates based on security parameters. In addition, it features an anomaly detection scheme that we developed as part of the security monitoring which can be used for notification of low, medium and high risk attacks.

To verify the pertinence of our DIPS-Mon prototype system implementation, we conduct experiments and collect results by leveraging realistic testbed environment within the DeterLab infrastructure [5]. We create a variety of data integrity cyber-attack templates within DeterLab (e.g., man-in-the-middle to compromise data integrity, DDoS to disrupt news feed service) in order to study the cause and effect of intrusions in our testbed configurations. Based on the insights from our experiments, we motivate the need for solutions that can be used to defend against cyber attacks that impact data integrity of news feeds hosted within cloud-networked infrastructures and also present mitigation solutions to the challenges we identify. We remark that our literature review suggests that our work in this paper features an exciting research and development problem related to data integrity monitoring of just-in-time news feeds, which is inter-disciplinary (journalism and cyber security) and is not extensively studied in existing literature.

The remainder of the paper is organized as follows: Section II provides a discussion on related works. Section III presents data integrity protection use case details through workflow and infrastructure issues that motivate our DIPS-Mon requirements. Section IV describes our data integrity...
Fig. 1: Web Services workflow for data integrity protection study with just-in-time news feeds

monitoring and protection scheme used in DIPS-Mon prototype system implementation for sorting news feeds based on results of anomaly detection and trust scores related to news feeds from affiliates. Section V lists our experiments and cyber-attack impact results from the various DeterLab testbed configurations for different attack templates. Section VI concludes the paper and presents future work.

II. RELATED WORK

Advances in web services within public cloud platforms has led the media companies to move from the traditional paper-print news to online news (e.g., RSS feeds) that provide faster access to information to their subscribers, and real-time updates of desired global events as they evolve [6]. Moreover, media companies are migrating their infrastructure operations from their private data centers into hybrid cloud infrastructures that provide public cloud benefits of rapid procurement and pay-as-you-go pricing [7].

Though the use of cloud computing has been proven to be efficient, flexible and cost effective, cloud infrastructure tends to be a ‘vulnerability amplifier’ that impacts infrastructure elements at large scale. The impact is seen when attackers launch cyber attacks to manipulate web service connections or to attack data center compute/network/storage resources [8]. Works such as [9] and [10] list the security challenges in maintaining cloud-networked infrastructures for application providers. Authors in [4] discuss several possible threats such as Denial of Service attacks, injection attacks, parameter tampering and SOAP Action spoofing that can impact web services in cloud-networked architectures; they argue that installing a firewall in the network-edge to filter out invalid packets can lead to dropping legitimate packets and thus do not present a pertinent solution. Works described in [11] and [12] explain that the DDoS attack frequency is increasing and more investment is being made in launching cyber attacks with highly sophisticated equipment to bring down vital end-user services. They show that none of the prevention systems, firewalls or packet scanning software can neither prevent nor sufficiently countermeasure the DDoS attacks; legitimate traffic is being penalized by dropping the packets from a high traffic route in current implementations and traceback tools suffer from performance issues such as low scalability, high processing and storage costs.

Authors of works such as [13] explain that each DDoS attack is specific at each phase and three attack detection techniques are developed for four categories: attack agent development phase, attack agent distribution phase, attack agent control phase and attack phase. The authors in [14] have shown the classification of the various DDoS based attacks, and the authors in [15] have discussed how to preserve the availability under distributed denial of service attacks. Cyber attack basics for Man-in-the-Middle strategy are explained in works such as [16] and [17]. The authors in [18] uses key distribution as a way to prevent man-in-the-middle attack, and the authors in [19] use sophisticated software and hardware solutions to mitigate such attacks.

Previously, authors in [20] propose anomaly detection on packet header information to identify modifications to the packets. In contrast, our work uses motivation from [21] - [23] to realize an anomaly detection algorithm that identifies change events in the context of just-in-time news feeds. We couple the anomaly detection with a trust model between media companies and their affiliates using concepts from [24].

III. DATA INTEGRITY PROTECTION NEEDS FOR JUST-IN-TIME NEWS FEEDS

In this section, we motivate the use cases that need to be supported within just-in-time news feeds cloud infrastructure that enables secure collection of user requests, processing of news feeds in a personalized manner, and subsequent delivery to subscribers on their mobile devices or PCs.

A. Use Case Workflow and Implementation

Fig. 1 shows the workflow to ensure data integrity protection through security monitoring for a media company (i.e., Thomson Reuters) use case to better serve premium subscribers of just-in-time news feeds. Our workflow consists of a secure portal for the premium subscribers to login, an application dashboard which displays the new feeds sorted based on the trust scores (the news feed with highest trust score appears on top of the list), and a Resource Discovery Portal that shows the various resources in the cloud and their performance.
information. The performance information is an important component of the infrastructure resource monitoring to detect anomalies due to potential cyber attack activities.

We implemented this workflow in our DIPS-Mon web application and hosted it on the DeterLab cloud testbed, which can be used to scale up or down based on the number of users and volume of news feeds. Firstly, the web application starts with a “Login Page” to request users (either news media stakeholder or subscriber/user) to access the news feeds interface that is customized for the user type (e.g., regular user or premium user or administrator). Upon login, the news feeds displayed in the “Feeds Dashboard Page” can be personalized based on the topic needs, and all of the news feeds are sorted and displayed as per their trust scores (as detailed in Section IV-C) and affiliate source. Each news feed consists of a short description of the main content, followed by the trust score in red font and then the linked reference to the original article from where it was retrieved.

B. Testbed Infrastructure

In this subsection, we describe the various testbed infrastructure components that are shown in Fig. 2 pertaining to our DIPS-Mon implementation for collecting the news feeds from the stakeholders/affiliates, processing them and then subsequently displaying the sorted news feeds (sorted based on the trust score of each news feed) to the user. Brief descriptions of the various testbed infrastructure components are as follows:

1) Feeds server: The feeds server performs the core logic of trust score calculation for the DIPS-Mon system. Whenever the feeds server receives a HTTPS request from the authorized user to display the news feed, it displays the news feeds to the user by collecting them from the ‘Collector Server’.

2) Database: The database stores the news feeds along with the metadata associated with the feeds. The meta data includes information such as the website from which the news feed was collected, when the news feed was collected, and so on.

3) Collector server: The collector server is responsible for collecting news feeds from different sources, parsing the news feeds and also calculating the trust score for each feed based on the criteria such as: encryption type, encryption length and anomaly scores. It then sorts the news feeds based on the trust score values to store them in any required format in the database.

IV. DATA INTEGRITY MONITORING AND PROTECTION SCHEME

In this section, we explain the details starting with the anomaly detection issues at the infrastructure-level and at the URL-level that help our DIPS-Mon’s core logic scheme to output anomaly detection scores. Next, we discuss the factors that are used to calculate the trust score of news feeds from a particular affiliate. Based on these two scores information, we lastly explain how we calculate the trust score for each news feed and subsequently how these news feeds are sorted and ranked before they are pushed to the subscribers for consumption.

A. Anomaly Detection for news feeds

Anomalies refer to the abnormal deviation of the network conditions from the regular/baseline network conditions. This deviation might be due to an attacker performing some malicious activity in the network path through which the stakeholders send the news feeds to the news agency or on the path through which the news agency send the news feeds to the regular or premium subscribers. The lower the number of anomalies in the network, greater is the trust score value for news feeds coming through that URL or network path. We then create alarms based on events at both the cloud infrastructure and URL levels.

To perform an anomaly detection, firstly we need to identify the steady state condition of the network path into the cloud infrastructure. This phase is called the “Training Phase”, where we train the anomaly detection algorithm using data collected from the steady state network condition. In the “Training Phase”, a threshold is setup by calculating the average mean value for the data, and then this average value is used to find the deviation of the data in the “Detection Phase”, where we detect for anomalies using deviation from the data in the “Training Phase”. Then alarms are created on event correlation data from “Detection Phase” at both the cloud infrastructure and URL levels.

The workflow of a basic Anomaly Detection algorithm is shown in Fig. 3. We can see that in the “Training Phase”, we have the audit data which is sent to a “Learning Engine”. The “Learning Engine” creates a profile using different metrics. This profile is used by the “Detection Engine” to create an alarm if an anomaly event occurs. The profile in Fig. 3 shows
Anomaly Detection is based on the concept that a significant amount of computer’s processing resources such as CPU usage, Network Bandwidth are consumed when a physical device such as a server is attacked.

We have setup the Infrastructure Anomaly Detection algorithm in an DeterLab instance, where we used different tools such as: bandwidth of the network, latency of the network, CPU usage of the server and Round Trip Time (RTT) of the packets being sent on the network. We then create an initial threshold for each of the above metrics by taking the incoming URL of the news feeds sent from affiliates to the news media company. In the following, we describe the details of both Cloud Infrastructure and HTTP URL anomaly detection algorithms.

1) Infrastructure Anomaly Detection: The Infrastructure Anomaly Detection is based on the concept that a significant amount of computer’s processing resources such as CPU usage, Network Bandwidth are consumed when a physical device such as a server is attacked.

We have setup the Infrastructure Anomaly Detection algorithm in an DeterLab instance, where we used different tools such as: bandwidth of the network, latency of the network, CPU usage of the server and Round Trip Time (RTT) of the packets being sent on the network. We then create an initial threshold for each of the above metrics by taking average values in the “Training Phase”. Following this, in the “Detection Phase”, observed events are compared with the initial threshold values and anomaly events are detected.

2) URL Anomaly Detection: The “URL Anomaly Detection” checks the abnormality of a URL through detection of the deviation of the character distribution in the received URL string, when compared with the character distribution in the training set. If the URL of the news feed is abnormal, it implies that either the URL was modified in the network or the URL received has been corrupted while being transmitting in the network path. Greater the anomaly score of the URL, lesser is the trust value for the news feed sent from the URL.

We use different models such as ‘Attribute length’, ‘Attribute Character distribution’ to identify anomalous entries in the HTTP request. Next, we combine each of these models with weight \(w_m\) and deviation score \(p_m\) from that model to get our final anomaly score as,

\[
\text{AnomalyScore} = \sum_{m \in \text{Models}} w_m \times (1 - p_m)
\]  

(1)

Attribute Character Length: The length of a request query attribute can be used to detect the anomalous request. Buffer-overflow and cross-site attacks tend to have a request query length higher than the normal query length.

Learning Phase: In the learning phase, we collect the lengths \(l_1, l_2, ..., l_n\), and the mean \(\mu\) and variance \(\sigma^2\) are calculated using the lengths.

Detection Phase: In the detection phase for a URL of length \(l\), the resulting anomaly value for the URL can be given by \(p(l)\) as,

\[
p(l) = \frac{\sigma^2}{(l - \mu)^2}
\]

(2)

Attribute Character Distribution: The “Attribute Character Distribution” analyzes the frequency of the character distribution provided in the URL. We have 256 characters in the ASCII codes, and so in the learning phase, we inject a number of ‘regular’ looking URLs into the system, and then update each of the corresponding ASCII code using characters in the URL. In the detection phase, when a new incoming URL is entered, we compare this URL with the existing ASCII code table, and give a trust score to the incoming URL.

As a sample example, let us use “passwd” as the input parameter string for which we can write -

\[
\sum_{i=0}^{255} \text{ICD}(i) = 1.0
\]

(3)

Learning Phase: The ASCII code breakup for individual characters of “passwd” are 112 for p, 97 for a, 115 for s, 115 for w, 100 for d. And thus the “Idealized Character Distribution (ICD)” for each of these characters is as shown in Table I.

\[
\begin{array}{c|cccccc}
\text{Key} & p & a & s & w & d \\
\text{Value} & 0.17 & 0.17 & 0.35 & 0.17 & 0.17 \\
\end{array}
\]

For each of the incoming URL in the learning phase, the “Idealized Character Distribution” values for each character gets updated in the table.

Detection Phase: Given an “Idealized Character Distribution (ICD)”, the job of the detecting phase is to determine the anomalies in the input URL string by analyzing it’s deviation from the above character distribution. This deviation from the “Idealized Character Distribution (ICD)” is calculated using a statistical test.

The detection algorithm uses a variation of Pearson \(X^2\) test. The incoming URL string values can be distributed into bins in Table II based on the \(X^2\) value for each of the characters in the input URL string. The Table II shows the corresponding bin values for a distribution of the \(X^2\) values. For example, bin 4 contains \(X^2\) values ranging between 12 and 15.
TABLE II: Distribution table

<table>
<thead>
<tr>
<th>Bin</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-values</td>
<td>0</td>
<td>1 - 3</td>
<td>4 - 6</td>
<td>7 - 11</td>
<td>12 - 15</td>
<td>16 - 155</td>
</tr>
</tbody>
</table>

The test performs the following steps -

1) Calculate the observed frequencies $O_i$, and we have the expected frequencies $E_i$

2) Compute $X^2$ as, $X^2 = \sum_{i=1}^{6} \frac{(O_i - E_i)^2}{E_i}$

3) The $X^2$ is compared to the value in the table and the corresponding probabilistic value is returned from the table.

Example using an URL Anomaly Detection: Herein, let us consider an example of anomaly score calculation using the URL anomaly detector. For the URL anomaly detection, we first train the data in the “Learning Phase” using 4 URLs as shown below to test our “Detection Phase” Anomaly Detection algorithm.

2) http://espn.go.com/college-football/story-id/10029963/georgia-bulldogs-aaron-murray-torn-
alabama-crimson-tide
3) http://espn.go.com/college-football/story-id/10045345/georgia-bulldogs-aaron
4) http://www.ncaa.com/game/football/fbs/2014/01/01/nebraska-georgia

We enter an uncommon input string in the form of an URL as shown below to test our “Detection Phase” Anomaly Detection algorithm.

@@@...@@@

Then,

1) The string length probability is found to be equal to 0.135583
2) The character distribution frequency probability is found to be 0.005
3) Thus, the URL’s final anomaly score = 0.4*0.135583 + 0.6*0.005 = 0.942767

This represents an abnormal link, and hence we are able to show that our Anomaly detection algorithm is able to detect this abnormality of a URL and thus the URL is assigned a very high anomaly score.

Next, we test our algorithm using a regular string as - http://espn.go.com/college-football/story-id/10045345/georgia-bulldogs-aaron

Then,

1) The string length probability is found to be equal to 1
2) The character distribution frequency probability is found to be 0.995
3) Thus, the URL’s final anomaly score = 0.4*1 + 0.6*0.995 = 0.995

We can see that the links close to those we used in the “Learning Phase” have a very low value of Anomaly Scores, and thus can be more trusted by the news media companies.

B. Trust Score Calculation for Affiliate News Feeds

The trust score calculation is an essential component in ranking and sorting of the news feeds. It determines the degree of trust that a news feed possesses before it is presented to a regular or premium subscriber. A lower value of trust score determines the possibility of偏低 security measures in transmitting feeds from affiliates, and as a result there would be higher chance of abnormal manipulation that might happen while transmitting the news feed or while the news feed is stored in a cloud database i.e., during Data-in-Motion and in Data-At-Rest.

The factors that determine the trust score for a particular news feed along with the anomaly detection score are: Certificate Level, Certificate Issuer, Encryption Type and Encryption Key Length.

1) Certificate Level: We use the hierarchical trust based Certificate Chain to define the Certificate Level. The stakeholders have to buy the certificates in order to prove they are secure. Better the certificate a news feed supplier holds, better is the affiliate’s trust value. The Root CA has the highest trust value as it is in the top of the certificate chain. They are self-signed and can be used to sign any low level or intermediate certificates; it is recommended that any party that has the highest level of certificate in the hierarchy is to be trusted by the premium subscribers. We use a scale from 0 to 5 to define the level of certificate. Root certificate has the highest value for “Certificate Level” i.e., 5. And lower the certificate level, lesser is the trust score for that Certificate.

2) Certificate Issuer: We have different Certificate Issuers who issue certificates to users. Each Issuer has a certain market share (in terms of percentages), which defines the number of websites that use the Certificates from this Issuer. In this model, we use this factor to say that - greater the market share of the Certificate Issuer used by the user, greater is the user’s trust score. Symantec certificate has the highest market share of 37.1% at the time of this writing, and thus our stakeholders using Symantec certificates have the highest trust scores.

3) Encryption Type and Encryption Key Length: News feeds are encrypted using suitable encryption methodologies when they are being sent from the stakeholders to the news agency. A good encryption algorithm is the one which uses the highest key length during the transmission of the news feeds. So, greater the key length, higher is the trust score for the news feed sent from the affiliate stakeholder to the news media company. We know that Diffie-Hellman and DSA algorithms have the highest key lengths of 1024 bits, and are thus given highest trust scores. Similarly, Data Encryption Standard (DES) that has 64-bit key length is given the lowest trust score.

C. Ranking and Sorting of News Feeds for Subscriber Access

Each of the factors are then weighed on a scale of 0-5 represented by scorec, with 0 indicating the least trust score for a particular factor, and 5 indicating the highest trust score for the factor. The weight factor and the weighing scores are arranged in such a way that the final trust score is less than or equal to 30 i.e.,
\[
\sum w_i \times (\text{score}_i) \leq 30
\]  

(4)

The individual scoring for a particular website along with the final trust score on a scale of 30 is calculated as shown below. For example, consider the link/URL from which we get a particular news feed as - http://espn.go.com/blog/ncfnation/tag-name/bcs-championship-2014

Let us assume the weighing factor for each of the below metrics is 1. Then the final trust score of the link/URL is calculated on a scale of 30 by summing up each of the metrics as shown below.

1) CA issuer: CN = Entrust.net Certification Authority (2048): **Trust score**: 1 This CA is the least trusted one because it has a very low market share and is being used by less number of people
2) Cipher Suite: SSL_RSA_WITH_RC4_128_SHA: **Trust score**: 5 This is a strongly encrypted Cipher Suite and hence the score 5
3) Key Length used: 2048: **Trust score**: 5 2048 is the strongest key found till now, and is very difficult to break, hence the score 5
4) Level of CA certificate: **Trust score**: 4 This level of certificate is more reliable and hence the score 4
5) Protocol used: TLSv1: **Trust score**: 4 Good Transport Layer security and hence trust score of 4
6) Anomaly score: **Trust score**: 5 The anomaly score for this URL is very less, equal to 0.03, hence the high trust score of 5
7) **Final trust value** = 1*1 + 1*5 + 1*5 + 1*4 + 1*4 + 1*5 = 24

This trust score is then used for sorting the news feeds. The sorted news feeds are displayed using the web application when users are directed to the “Feeds Dashboard” page described earlier in Section III-A. In the “Feeds Dashboard” page, the news feeds sorted in the order of their trust scores calculated using various factors such as Encryption Type and Key Length, Certificate Issuing Authority and the Anomaly Detection scores as explained above.

V. EXPERIMENTS AND RESULTS

This section describes the “DIPS-Mon” implementation we have setup on the DeterLab testbed, and the various network attacks we have performed to validate our cloud-networked architecture. Our implementation architecture is illustrated in Fig. 4, upon which we performed Data Integrity and DoS attacks to check the sustainability of the system, and the validity of our trust model when these attacks were performed.

A. Experiment Setup

The experiment setup on the DeterLab cloud testbed consists of the following components:

**Sources A, B, C, D, and BadSourceE:** These are the RSS news feed sources. Their purpose is to supply the latest RSS feeds when they receive HTTP GET requests from CollectorAndConvert. For the sake of simplicity, we assume the news feeds from SourceA to be SourceA1, SourceA2, SourceA3, SourceA4 and SourceA5 so that in the results we can just inspect these names, which makes it easy to analyze the results. The textitBadSourceE if enabled, sends out malicious scripts attached with RSS feeds and URLs after getting the GET request from CollectorAndConvert server.

**Clients A, B, C, D, Attacker1 and Attacker2:** Clients are created to simulate the client browsers, where they can send GET HTTP requests to the news agency, and as a response can receive the RSS feeds sorted based on the trust scores. The Attacker nodes if enabled, act as a zombie nodes, where they keep sending a large number of GET HTTP requests to the web server thereby blocking all the ports on the web server and thus creating a DOS attack.

**Middleware Components:** It consists of a CollectorAndConvert, which is responsible for collecting all the RSS feeds from different stakeholders and reformattting them from XML to other required formats, a DBORStorage, which is a Database for storing all the processed incoming feeds from the stakeholders, a WebServer1 and WebServer2, responsible for the core calculation of the trust score and ranking the news feeds from the stakeholders, and a FirewallORForward which is used to control the incoming flow. At first, it just acts as a reverse proxy that forwards all the incoming traffic. However, later on, it acts as a protecting agent, scanning the incoming traffic for malicious content.

B. Cyberattack Impact Results and Defense Solutions

1) Pre-Attack Phase: In this phase, the testbed runs in the regular mode where there are no attacks on the cloud system, and then the resulting data is collected and analyzed. This information is stored to compare with the resulting data in the attack phase and post-attack phase. After setting up the testbed...
and running the experiment, all the links deliver traffic without any problems. In this scenario, BadSourceE, Attacker1, and Attacker2 are disabled, and the sorted trust values for the data sent by each of the client is as shown in Fig. 5.

2) Attack Phase and Results: In this phase, we first perform Data Integrity attacks and analyze the collected results, and follow-up with DoS attack results. Lastly, we outline defense solutions to mitigate the impact of each of these attack types within our DIPS-Mon implementation.

1 Data Integrity Attacks:

1.1 Man-in-the-Middle Attack: In “Man-in-the-Middle” attack, a device in the middle corrupts the data coming from SourceA by injecting some noise into the RSS feeds. A very long URL was replaced with junk data in the new data coming from SourceA. The titles in SourceA4 and SourceA5 was also modified to make them appear as unreadable strings. Next, the testbed scripts were started and the results were collected. After injecting noisy data and running the testbed, we can see from Fig. 6 that the score of SourceA3 has been reduced due to faulty URL. Whereas, the trust scores of SourceA4 and SourceA5 have also been decreased due to the data modification using anomaly detection for network conditions. Thus, our DIPS-Mon works well under data modification attacks.

1.2 Bad Source Attack: The trusted source SourceA is replaced with BadSourceE. Consequently, the RSS feeds coming from SourceA are actually coming from BadSourceE. After the bad source attack is performed, BadSourceE feeds appear instead of SourceA feeds in ClientA with different trust scores. So, through our Anomaly Detection algorithm, we were able to detect a change in the network conditions of the new source and were able to identify that there is a change of the source and hence we assign a low score to such news feeds. Once the source properly identifies itself over a period of time, we increase it’s trust score gradually.

Protection from Integrity Attacks: From the testbed, all external incoming and outgoing traffic go through FirewallORForward1 and FirewallORForward2. These two nodes currently just act as forwarding traffic agents, and they are doing the expected functions. The nodes that perform the core logic need to be modified to check the traffic to see whether it contains malicious scripts, noisy data or not. Also, our anomaly detection algorithm needs to be applied to check for the abnormality in the URLs. Malicious URLs usually contain abnormal patterns in their strings. Also, we must ensure that SourceA3 coming from SourceA. The titles in SourceA4 and SourceA5 was also modified to make them appear as unreadable strings. Next, the testbed scripts were started and the results were collected. After injecting noisy data and running the testbed, we can see from Fig. 6 that the score of SourceA3 has been reduced due to faulty URL. Whereas, the trust scores of SourceA4 and SourceA5 have also been decreased due to the data modification using anomaly detection for network conditions. Thus, our DIPS-Mon works well under data modification attacks.

1.2 Bad Source Attack: The trusted source SourceA is replaced with BadSourceE. Consequently, the RSS feeds coming from SourceA are actually coming from BadSourceE. After the bad source attack is performed, BadSourceE feeds appear instead of SourceA feeds in ClientA with different trust scores. So, through our Anomaly Detection algorithm, we were able to detect a change in the network conditions of the new source and were able to identify that there is a change of the source and hence we assign a low score to such news feeds. Once the source properly identifies itself over a period of time, we increase it’s trust score gradually.

Protection from Integrity Attacks: From the testbed, all external incoming and outgoing traffic go through FirewallORForward1 and FirewallORForward2. These two nodes currently just act as forwarding traffic agents, and they are doing the expected functions. The nodes that perform the core logic need to be modified to check the traffic to see whether it contains malicious scripts, noisy data or not. Also, our anomaly detection algorithm needs to be applied to check for the abnormality in the URLs. Malicious URLs usually contain abnormal patterns in their strings. Also, we must ensure that
the incoming RSS news feeds are really coming from the authenticated and trusted sources before storing them in the cloud database.

2. DoS Attacks: Nodes Attacker1 and Attacker2 send out too many GET requests trying to occupy and slow down all web server resources to stall the service. While this attack is going on, ClientA sends a regular request for accessing the resources. The response time for ClientA with 2 attackers is shown in Fig. 8, and the response time with 4 attackers is shown in Fig. 9. We can clearly see that the response time for legitimate users increases exponentially with the number of attackers.

**Protection from DoS attack:** Our anomaly detection algorithm needs to be applied to check for change events in the cloud network whenever a user sends a request to the server, and also to check for the time periods between all the requests coming from one source. If this time is less than the threshold time, we do not forward the request to the web server. Another solution is moving the trust score calculation from the web server to a stage that is before the storage, and after collecting and converting RSS feeds. This approach has the potential to free-up a significant amount of web server resources in the cloud, and can make the delivery speed of news feeds relatively more faster.

VI. Conclusion

In this paper, we introduced an exciting research and development problem related to data integrity monitoring of just-in-time news feeds, which is inter-disciplinary (journalism and cyber security) and is not currently being studied in cloud networking community. More specifically, we described our DIPS-Mon approach for cloud-networked infrastructures that assigns “trust scores” for ranking news feeds based on security parameters such as: SSL encryption, certificate level, certificate authority (CA) issuer, encryption schemes, and security audit rating of affiliates. In addition, we presented an anomaly detection scheme that we developed as part of the security monitoring for notification of low, medium, and high risk attacks that cause data integrity violations, and impact cloud-networked news media delivery services. Lastly, we explained how we created a variety of data integrity cyber-attack templates (e.g., man-in-the-middle to compromise data integrity, DDoS to disrupt news feed service) in order to study the cause and effect of intrusions, by leveraging cloud testbeds in DeterLab.

In future work, we plan to couple our anomaly detection scheme with data mining techniques to monitor data integrity violations of RESTful web services based URLs, as well as actual content of news feeds that involves new challenges in the application of the realm of natural language processing to enhance our DIPS-Mon system.

**References**