RTP-Miner: A Real-time Security Framework for RTP Fuzzing Attacks

M. Ali Akbar and Muddassar Farooq
Next Generation Intelligent Networks Research Center (nexGIN RC)
National University of Computer & Emerging Sciences (FAST-NUCES)
Islamabad, Pakistan
ali.akbar@nexginrc.org, muddassar.farooq@nu.edu.pk

ABSTRACT

Real-time Transport Protocol (RTP) is a widely adopted standard for transmission of multimedia traffic in Internet telephony (commonly known as VoIP). Therefore, it is a hot potential target for imposters who can launch different types of Denial of Service (DoS) attacks to disrupt communication; resulting in not only substantive revenue loss to VoIP operators but also undermining the reliability of VoIP infrastructure. The major contribution of this paper is an online framework – RTP-Miner – that detects RTP fuzzing attacks in realtime; as a result, it is not possible to deny access to legitimate users. RTP-Miner can detect both header and payload fuzzing attacks. Fuzzing in the header of RTP packets is detected by combining well known distance measures with a decision tree based classifier. In comparison, payload fuzzing is detected through a novel Markov state space model at the receiver. We evaluate RTP-Miner on a real-world RTP traffic dataset. The results show that RTP-Miner detects fuzzing in RTP header with more than 98% accuracy and less than 0.1% false alarm rate even when only 3% fuzzing is introduced. For the same fuzzing rate, it detects payload fuzzing – a significantly more challenging problem – with more than 80% accuracy and less than 2% false alarm rate. RTP-Miner has low memory and processing overheads that makes it well suited for deployment in real world VoIP infrastructure.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General—*Security and protection*

General Terms

Experimentation, Security

Keywords

Denial of Service, Real-time Transport Protocol, Fuzzing Attacks, VoIP, Machine Learning, Stochastic Models

NOSSDAV'10, June 2–4, 2010, Amsterdam, The Netherlands.

1. INTRODUCTION

The global communication market is rapidly moving towards Voice over Internet (VoIP). The prime motivation behind this trend is ubiquitous availability of high-bandwidth Internet at cheaper rates compared with circuit switched telecommunication networks. A market survey released in 2008 shows that VoIP traffic accounts for 49.7% of total voice traffic [22]. In 2007, a German company *Ipoque* carried out an in-depth analysis of 3 petabytes of Internet traffic collected from five regions of the world. The results of their study show that 30% of German Internet users subscribe to VoIP services [9].

The popularity of VoIP makes it an attractive target for attackers. VoIP servers are included in the SANS Top 20 Security Risks [15]. Another study shows that VoIP servers are among the top 5 emerging cyber security threats in the year 2009 [7]. The intruders know that a successfully launched Denial of Service (DoS) attacks on VoIP servers can result not only in huge financial losses to the operator and its customers, but also can seriously undermine its credibility. In VoIP, Session Initiation Protocol (SIP) is used during the signaling phase of establishing VoIP calls. Later, media is transferred using widely adopted Real-time Transport Protocol (RTP)[16]. The focus of this paper is to analyze vulnerabilities in RTP[1] protocol and propose effective and efficient countermeasures to mitigate them.

The open nature of Internet enables smart attackers to exploit vulnerabilities in media servers (mostly running RTP) and VoIP phones by mutating different fields in the header and changing byte patterns in the payload of RTP packets. In the first case, they aim at crashing the protocol processing component of an RTP based multimedia application. In the second case of payload fuzzing, they aim at exploiting vulnerabilities in the buffer management, alignment of bytes in the payload, decoding of received multimedia streams and playing them to the end user. In recent past, several researchers have identified these malformed (fuzzed) RTP packet vulnerabilities for multimedia proxies & servers, packet sniffers and SIP phones [3, 17, 25, 4]. Recently, security researchers have crashed a widely used Cisco IP phone when it was repeatedly exposed to fuzzed RTP packets [5]. An attacker, by exploiting these vulnerabilities, is able to crash a VoIP application/device that results in causing denial of service to the legitimate users. In the worst case

[1]In this paper, the term RTP only refers to the data transfer sub-protocol of RTP, and not the Real-Time Control Protocol RTCP, which is used for managing QoS of a session.

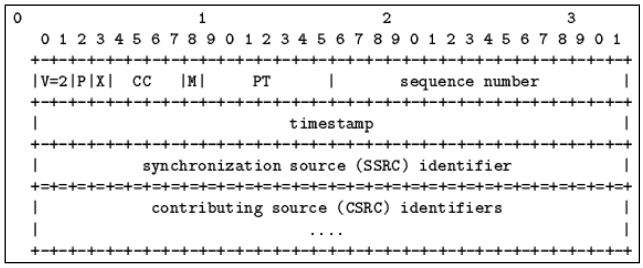


Figure 1: RTP Header Format

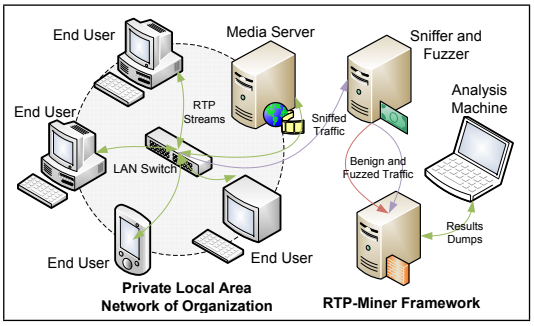


Figure 2: Real World Testbed

scenario, it might enable an attacker to remotely execute malicious code on the compromised system.

The major contribution of this paper is RTP-Miner – an efficient and effective online intrusion detection framework – that can detect header and payload fuzzing attacks in real-time. The fuzzing in the known header fields is detected by modeling the difference in the header fields of benign[2] RTP traffic with that of fuzzed RTP by using well known information theoretic distance measures. We use a decision tree based classifier to identify an RTP packet with fuzzed header. In comparison, it is a challenge to detect fuzzed payloads because here fuzzing means arbitrarily changing different bit patterns. We use a novel Markov state space model to identify payload fuzzing at the receiver application without requiring the sender application to insert checksums[3]. Other important contributions of the paper are collection of real-world RTP datasets, and development of a tool that can apply fuzzing transformations in header and payload of RTP packets. The results of our experiments show that RTP-Miner is able to detect fuzzing in RTP header with more than 98% accuracy and less than 0.1% false alarm rate even when only 3% fuzzing is introduced. For same fuzzing rate, it detects payload fuzzing with more than 80% accuracy and less than 2% false alarm rate. Last but not least, RTP-Miner has low memory and processing overheads that makes it well suited for online deployment in real world VoIP infrastructure.

The rest of the paper is organized as follows. In Section 2, we describe that how we have collected real-world RTP traffic datasets and fuzzed datasets. We present the architecture of RTP-Miner in Section 3. We discuss the results of our experiments in Section 4. The related work is described in Section 5. Finally, we conclude the paper with an outlook to our future work.

2. DATASET AND REAL-WORLD TESTBED

In this section, we describe our benign and fuzzed RTP traffic datasets. We also discuss the architecture of our testbed that we have used to collect real-world RTP media traffic. We also present our fuzzing process that can be customized to introduce (not so easy to detect) fuzzing at different rates – both in header and payload – in RTP packets.

Benign Dataset. Our real-world RTP testbed is shown in Figure 2 that we deployed in IMS Lab of our research center. Our center consists of a number of Labs and offices that are connected through a private local area network. Our network provides services to 25 users, which include professors, researcher engineers, students and support staff. The voice communication among these users is established through a local VoIP service that consists of a SIP soft-switch and RTP media server. For our experiments, we use *dsniff* [20] that sniffs the RTP traffic from the central switch. We logged RTP sessions for one day and used them in our experiments. Important statistics of collected RTP packets are: (1) the average RTP packet size in the collected dataset is 268 bytes, (2) total number of benign RTP packets are 40119, (3) the maximum inter-arrival time between two RTP packets is 564 msec, and (4) the average and maximum jitter are 43.85 msec and 60.7 msec respectively.

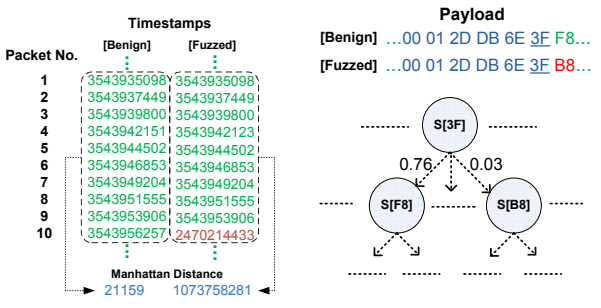
We have used MPEG-I/II Audio encoding which is widely deployed in many VoIP products [8]. An important reason for selecting this payload format is that a smart attacker can inject complex pathological datagrams in MPEG-I/II Audio encoded stream to launch a DoS attack [8]. In well known audio players, a number of vulnerabilities have been identified that could be exploited using specially crafted malformed MPEG encoded audio streams [14].

Fuzzed Datasets. We perform fuzzing in the header and payload of RTP packets. The structure of RTP header is shown in Figure 1. It consists of 9 fields and the total size of header is 96 bits. An interested reader can find description of these fields in [16]. In case of 1% fuzzing, we randomly select a bit and invert it. To generalize, x% fuzzing means that randomly selected x% bits are inverted. In case of payload fuzzing, if we want to apply y% fuzzing, then we randomly select y% bits of the payload and invert them. This process of fuzzing by introducing bit errors in RTP packets is not new and has been successfully used in VoIP fuzzing tools in the past [23]. We ensure that no obvious artifacts are introduced in fuzzed dataset that make detection intuitively simple.(For example our fuzzed dataset also contains 40119 fuzzed packets with same average size of 268 bytes as that of benign packets to rule out detection on the basis of size.)

Analysis of Datasets. We now analyze both datasets to build intuition about detecting fuzzed packets. In Figure 3(a), we show time stamps of 10 packets of benign and fuzzed datasets. We can see that the sum of differences – measured with well known Manhattan Distance – between time stamps of benign RTP packets is significantly smaller compared with a window that contains a packet with fuzzed time stamp. In order to detect fuzzing in header, we need to compute the

[2]The term *benign* refers to packets that do not have malformed (mutated) header or payload. In comparison, malformed RTP packets are termed as fuzzed.

[3]Checksums can be easily evaded in an unencrypted traffic flow because an attacker can recompute them after fuzzing the packet. Moreover, encryption becomes useless if the attacker is also the sender of the message.



(a) Header (b) Payload

Figure 3: Fuzzing in RTP Header and Payload

difference of each field within a window of packets and then apply suitable threshold mechanism to detect fuzzed header.

In comparison, it is not possible to apply a distance measure on payload because it does not consist of fixed fields; rather, it consists of byte sequences that follow some probability distribution. (This insight is developed through rigorous analysis of byte sequences appearing in payload of hundreds of RTP packets.) More specifically, we can determine the probability of encountering a given byte on the basis of current byte. We show a subset of state transition diagram of benign payload in Figure 3(b). If the current byte is 3F, it is highly unlikely in a benign packet that the following byte will be B8; as a result, a fuzzed packet can be potentially detected through a sequence of unlikely state transitions.

3. ARCHITECTURE OF RTP-MINER

We now present detailed architecture of our intrusion detection system – RTP-Miner – for detecting header and payload based fuzzing attacks. The system diagram of our framework is shown in Figure 4. An incoming RTP traffic stream is sniffed from an RTP session. We then split an RTP packet into header and payload and store it into memory. The header is processed by Header Fuzzing Detection Module (M_H) and the payload is processed by Payload Fuzzing Detection Module (M_P). Both modules use separate models to identify header or payload fuzziness. If any module raises an alarm, it is dropped; otherwise, it is given to the RTP application. We now discuss both modules separately.

3.1 Header Fuzzing Detection Module (M_H)

The basic working principle of this module – as mentioned before – is that the header fields of subsequent packets in an RTP session are closely related. Consequently, we monitor the spatial distribution of header fields in a sliding window of k packets. The sliding window is implemented using a queue: when n_{th} packet arrives, its header fields are enqueued and the $(n-k)_{th}$ packet's header fields are dequeued.

In order to monitor the spatial behavior of each of the 9 header fields (see Figure 1), we apply well known distance measures[4] (tabulated in Table 1) on the header fields of the packets within a given window. As a result, we are able to model difference between different fields of packets in a given

[4]For brevity, we skip details about distance measures, but an interested reader can find them in [21].

Table 1: Information Theoretic Distance Measures

Distance	Formula
Angular Separation	$d_{as} = \frac{\sum_{j=1}^{k-1} X_j . X_{j+1}}{(\sum_{j=1}^{k-1} X_j^2 . \sum_{j=1}^{k-1} X_{j+1}^2)^{\frac{1}{2}}}$
Chebyshev Distance	$d_{ch} = \max\|X_j - X_{j+1}\|$
Entropy	$d_{en} = -\sum_{j=1}^{k} X_j . \log_2(X_j)$
Manhattan Distance	$d_{ma} = \sum_{j=1}^{k-1} \|X_j - X_{j+1}\|$

sliding window. We also need to learn (from the pattern of these differences) a threshold value for a given distance measure of each header field (X) that acts as the classification boundary between benign and fuzzed packets.

We use well known rule based machine learning classifier – J48 – that is trained on a small subset of our labeled benign and fuzzed RTP datasets. J48 is an open source Java implementation of the C4.5 algorithm [13]. C4.5 classification algorithm builds a decision tree to do classification of a given instance into benign and fuzzed. The feature (in our case a distance measure) that has the greatest classification potential (in machine learning it is called information gain) is selected as root node in the decision tree. The algorithm is repeated on the subsets until all features have been evaluated or no additional information gain is achieved by splitting data using the remaining features. After the decision tree is constructed, it is pruned to remove useless branches.

During the testing phase, the decision rules generated by J48 are used for classifying a window of RTP packets as benign or fuzzed. An excerpt from the decision tree is given below which contains the rule – `[d(timestamp) > 21167: fuzzed]` – and it is fired when a fuzzed packet arrives (see Figure 3(a)). Here, `d(timestamp)` denotes the Manhattan distance between the timestamps of packets of a window.

```
...   d(timestamp) <= 21167
      |   d(version) <= 0
      |   |   ...
      |   d(version) > 0: fuzzed
...   d(timestamp) > 21167: fuzzed
```

3.2 Payload Fuzzing Detection Module (M_P)

We have already discussed that the fuzzing detection in the payload of an RTP packet is significantly more challenging compared with fuzzing detection in the header because payload does not contain well defined fields. Our analysis reported in Section 2 reveals that the byte sequences, appearing in the payload of RTP packets, follow some sort of random distribution. Therefore, we model the payload as a random process with each byte representing a state and the sequence of bytes representing state transitions. In this way, we create a first order finite Markov Chain [10] that models the benign payload as a discrete state space model. We use this state space model to assign an anomaly score to the payload of each incoming RTP packet. If the anomaly score exceeds a threshold, the packet is declared as fuzzed and is dropped from the packet stream.

State Space Model. We now present the state space model of our anomaly detection module. If b_i is the i_{th} symbol in the payload, then the payload can be represented as an ordered set of symbols $\mathbb{B} = \{b_1, b_2, ...b_n\}$, where n is the length of the payload. Let $\mathbb{S} = \{s_0, s_1, s_2, ...s_k\}$ be the possible states of the system and s_0 be the initial state. Each symbol in the payload is mapped to one state of the

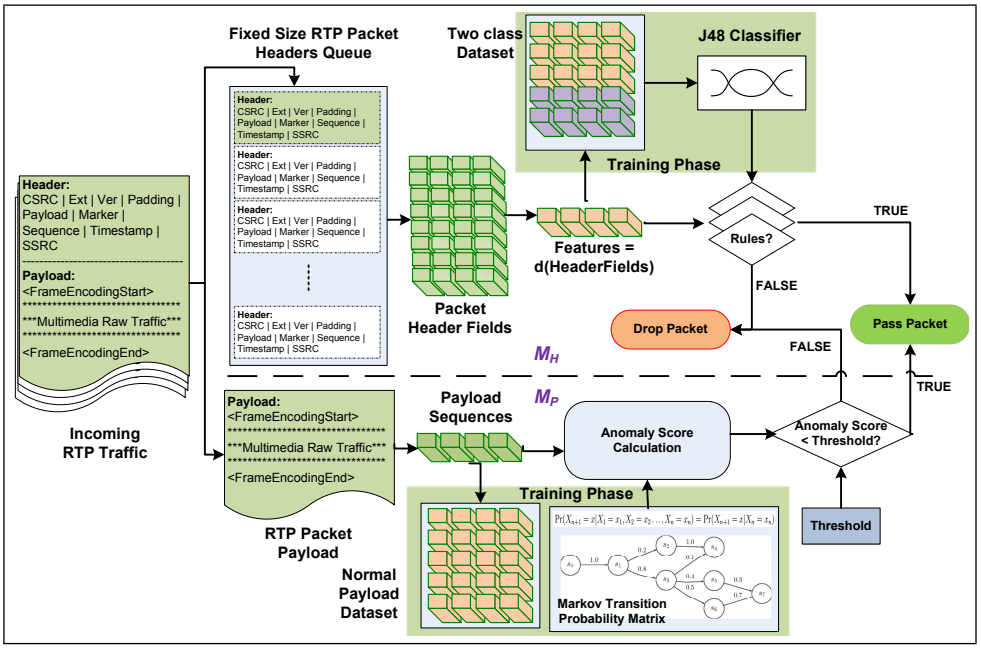


Figure 4: Architecture of RTP-Miner: Intrusion Detection Framework for RTP Fuzzing Attacks

system ($f : b_i \rightarrow s_x \in \mathbb{S}$). We say that a state transition has occurred when a symbol b_{i+1} follows the symbol b_i. Specifically, if $f : b_i \rightarrow s_x$ and $f : b_{i+1} \rightarrow s_y$ holds, we show transition between the two states as s_{xy} and the probability of this state transition as p_{xy}. We now define a transition function T that maps input symbols and state space to state transition probabilities. Mathematically, $T : \mathbb{S} \times \mathbb{B} \rightarrow P(\mathbb{S})$, where $P(\mathbb{S})$ denotes the state transition probability matrix of $\mathbb{S}$. As the total number of possible states is $k+1$ (including the initial state), and transitions are possible from one state to k different states; therefore, $P(\mathbb{S})$ is a matrix of size $(k+1) \times k$. Each row $0 \leq i \leq k$ of $P(\mathbb{S})$ satisfies the condition of probability which is $\sum_{j=1}^{k} p_{ij} = 1$.

During training phase, we read the payload of benign RTP packets as a byte stream. Each byte is taken as a symbol and the sequences of bytes represent state transitions. Assuming that the probability of each state transition is dependent only on the current state and not on any previous states[5], we calculate state transition probability matrix $P(\mathbb{S})$.

Anomaly Score Calculation. Our final objective is to detect fuzzed payloads during the testing phase within linear time ($O(n)$). With this objective in mind, we propose an incremental anomaly score calculation algorithm using the state transition probability matrix $P(\mathbb{S})$ and the incoming payload bytes $\mathbb{B}$. Let us start with an initial anomaly score $\chi_0 = 0$. For each byte sequence(b_{i-1}, b_i) that represents a state transition $(f : b_{i-1}, f : b_i)$, we increment the anomaly score by $\Delta\chi_i$. We quantify the change in anomaly $(\Delta\chi_i)$ on the basis of two factors: (1) it is directly proportional to the probability that the state transition will *not* occur $(\Delta\chi_i \propto (1 - p_{\rho\eta}))$, and (2) it is inversely proportional to the density of the state transition probabilities for a given state $(\Delta\chi_i \propto 1/(-\sum_{j=1}^{k} p_{\rho j}.\log_2(p_{\rho j})))$. If n is the size of the payload (size of the ordered set $\mathbb{B}$), then for $1 \leq i \leq n$,

$$p_{\rho\eta} = T(f : b_{i-1}, f : b_i) \quad (1)$$

$$\chi_i = \chi_{i-1} + \frac{1 - p_{\rho\eta}}{-\sum_{j=1}^{k} p_{\rho j}.\log_2(p_{\rho j})} \quad (2)$$

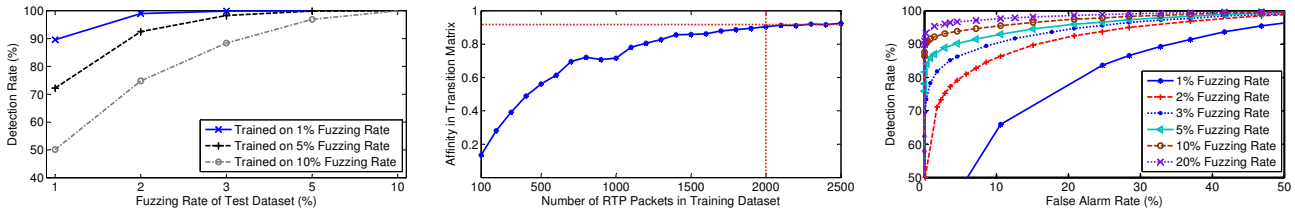
For example, the unlikely (low $p_{\rho j}$) state transition in fuzzed payload – as shown in Figure 3(b) – will result in a high change in anomaly score ($\Delta\chi_i = 2.064$) as compared to the likely (high $p_{\rho j}$) state transition in benign payload ($\Delta\chi_i = 0.51$). Therefore, we can identify RTP packet with fuzzed payload based on its high anomaly score using a fixed threshold. After completion of n iterations, the final anomaly score χ_n is normalized ($\bar{\chi}_n$). The density of the state transition probabilities for all states is already computed at the end of the training phase. As a result, the testing time for our proposed algorithm is linear ($O(n)$). We compare the final anomaly score $\bar{\chi}_n$ with a threshold χ_t. If $\bar{\chi}_n > \chi_t$, the packet is labeled as fuzzed and is dropped.

4. EXPERIMENTS & RESULTS

In this section, we evaluate the performance of RTP-Miner. We measure the performance on the basis of four metrics: (1) Detection Rate (DR) – the percentage fraction of fuzzed packets correctly detected as fuzzed, (2) False Alarm Rate (FAR) – the percentage fraction of benign packets incorrectly declared as fuzzed, (3) Processing latency (δ_t) – processing overhead to classify an incoming RTP packet as benign or fuzzed, and (4) Memory usage (M_u) – amount of memory needed to store features' set and discrete state space.

The accuracy of RTP-Miner is dependent on training datasets. We see in Figure 5(a) that if our system is trained on 1% fuzzed header datasets, the detection rate is significantly improved during testing phase. We run experiments to find out that a window size of 10 packets gives best accuracy. The results in Figure 5(a) are for Chebyshev Distance but the pattern remains same for other distance measures as well.

[5]This is known as Markov property $(p(S_{n+1} = s|S_1 = s_1, S_2 = s_2..., S_n = s_n) = p(S_{n+1} = s|S_n = s_n))$.



(a) Header Training Dataset (b) Payload Training Size (c) ROC for Payload Detection

Figure 5: Results

Table 2: Classification Accuracy Results for RTP Header and Payload Fuzzing Detection

	Fuzzing Rate											
	1%		2%		3%		5%		10%		20%	
Module	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**
Header(AngularS.)	75.2	0.0	94.0	0.0	98.9	0.0	98.9	0.0	100.0	0.0	100.0	0.0
Header(*Chebyshev*)	**89.6**	0.0	**99.0**	0.0	**99.9**	0.0	**100.0**	0.0	**100.0**	0.0	100.0	0.0
Header(Entropy	79.0	0.0	95.2	0.0	99.3	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Header(Manhattan)	88.9	0.0	98.8	0.0	99.9	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Payload	**34.3**	1.4	**71.2**	1.8	**81.8**	1.8	**87.0**	1.4	**92.2**	1.4	**95.4**	1.4

We know that payload fuzzing detection module is anomaly based and hence only requires benign packets; therefore, it is important to know the minimum number of benign packets needed to have complete state space model along with associated probabilities of state transitions. Let r_i be the number of packets used for training and $\mathbb{P}_{r_i}$ be the probability distribution of transitions from a particular state j to all states. We add Δr packets to the training dataset and the new probability distribution is $\mathbb{P}_{r_i+\Delta r}$. We measure the similarity between the two probability distributions by using *affinity* (a well known distance measure) [6]:

$$A_{ij}(\mathbb{P}_{r_i}, \mathbb{P}_{r_i+\Delta r}) = \sum_{\alpha=1}^{k} \sqrt{p_{r_i}^{(\alpha)} p_{r_i+\Delta r}^{(\alpha),}} \qquad (3)$$

where the parenthesized superscript $^{(\alpha)}$ is the index, not the power. We start with a training dataset of $r_0 = 100$ packets and iteratively add $\Delta r = 100$ packets. For each step, we calculate the minimum affinity between the states ($\min_j(A_{ij})$). The resulting minimum affinity for $i = 1$ to 25 (corresponding to a training size of 100 to 2500 packets) is plotted in Figure 5(b). It is obvious that the affinity does not increase if we use more than 2000 packets in our dataset; therefore, we conclude that training on 2000 benign packets is sufficient to evolve our benign Markov model.

4.1 Classification Accuracy Results

We now present and discuss the classification accuracy results for RTP-Miner in terms of *DR* and *FAR* separately for both header and payload fuzzing detection.

Header Fuzzing Detection. We evaluate RTP-Miner by using four well known distance measures and the accuracy results are tabulated in Table 2. (Remember we have trained our classifier on 1% fuzzed dataset with a sliding window of 10 packets.) Our header fuzzing detection module – using Chebyshev and Manhattan distance measures – achieves more than 88% DR with 0% FAR even for 1% fuzzed dataset. The DR exceeds 99% just for 2% fuzzed rates. We short list Chebyshev and Manhattan distance measures as the potential features for our header fuzzing detection module. From Table 1, we observe that Chebyshev distance can be computed in (O(k+1)) because of only one subtraction operator and a maximum operator, making it suitable for realtime deployment.

Payload Fuzzing Detection. We tabulate the accuracy results for payload detection in Table 2. Moreover, we plot ROC for different fuzzing rates in Figure 5(c). As expected, it is significantly more challenging to detect fuzzing in the payload. The reason is that RTP payload consists of multimedia traffic which has a random distribution; as a result, small fuzzing rates do not have significant perturbations in the distribution of payload. It is, however, interesting to see that just for 2% fuzzing, our system is able to detect 70% of fuzzed packets with less than 2% false alarm rate. An obvious trend in Table 2 is that FAR never exceeds 2%. We need to analyze the impact of dropping 2% benign packets on RTP sessions. Assuming that each packet contains 40 ms of data, the average packet size is 260 bytes and round trip time (RTT) is 200 ms; we estimate by using the methodology given in [12] that a packet loss rate of 6.2% is acceptable in our RTP media session. A drop rate of 2% is significantly smaller compared with this upper bound and hence we can safely conclude that this false alarm rate will not result in human perceivable degradation in the media quality.

To conclude, RTP-Miner is able to detect fuzzing in RTP header with more than 99% DR and 0% FAR even when only 3% fuzzing is introduced. For the same fuzzing rate, it detects payload fuzzing – a significantly more challenging problem – with more than 80% DR and less than 2% FAR.

4.2 Memory & Processing Overheads

The *memory usage* (M_u) and *processing latency* (δ_t) metrics (using Chebyshev distance) are listed in Table 3. It is clear from the table that processing overhead for each packet is significantly smaller compared with the average jitter (12.463 μsec as compared to 43.85 msec). Moreover,

Table 3: Memory and Processing Overheads

Module	Processing Latency (δ_t)	Memory Usage (M_u)
Header Module (M_H)	7.727 μsec/pkt	1.828 KB
Payload Module (M_P)	4.736 μsec/pkt	258 KB
Both Modules (series)	12.463 μsec/pkt	259.8 KB
Both Modules (parallel)	7.727 μsec/pkt	259.8 KB

our framework requires only 260KB of additional memory for storing its models. These resource requirements – coupled with high accuracy – make RTP-Miner a suitable candidate for online deployment in real-world scenarios even on resource constrained mobile devices.

5. RELATED WORK

Recently, a number of security frameworks have been proposed for VoIP intrusion detection [2, 19, 11]. However, most of these frameworks focus on SIP attacks or are restricted to RTP flooding attacks. Similarly, a number of research contributions present threat models for RTP protocol [24]. The objective of proposing different protocol state machines in [26, 18] is to detect RTP flooding and media spam.

In [5], the authors have shown security vulnerabilities of a widely used IP phone (Cisco 7960G) in handling RTP streams. Similarly, the authors in [1] also reported crash of several IP phones when subjected to fuzzing attacks. In comparison, little attention is paid to detect RTP fuzzing attacks; therefore, to the best of our knowledge, RTP-Miner is the first attempt to mitigate this serious threat in RTP.

6. CONCLUSION & FUTURE WORK

In this paper, we have presented an efficient, online intrusion detection framework (RTP-Miner) for real-time detection of RTP fuzzing attacks, that can ultimately lead to denial of service to legitimate users. Our results show that Chebyshev distance measure combined with J48 classifier is a good technique to accurately detect fuzzing in RTP header. In comparison, we use a novel Markov model to detect fuzzing in the payload of RTP packets. Our results show that our system can detect fuzzing in header and payload with significantly high accuracy. Moreover, the system has small processing and memory overheads. In future, we want to develop an intelligent model for dropping fuzzed packets. More specifically, we want to analyze the effect of dropping payload fuzzed packets on the quality of media stream.

Acknowledgments

The work is supported by National ICT R&D Fund, Ministry of Information Technology, Government of Pakistan through grant # ICTRDF/TRD/2007/59. The information, data, and views detailed herein may not necessarily reflect the endorsements of views of the National ICT R&D Fund.

7. REFERENCES

[1] H.J. Abdelnur et al. KiF: a stateful SIP fuzzer. In *IPTCOMM'07*, pages 19–20, 2007.

[2] M.A. Akbar et al. Application of evolutionary algorithms in detection of SIP based flooding attacks. In *GECCO'09*, pages 1419–1426. ACM, 2009.

[3] Asterisk-Dev. Asterisk crashes when receiving malformed RTP packets, 2004. http://www.mail-archive.com/asterisk-dev@lists.digium.com/msg03417.html.

[4] Ubuntu Bugs. Wireshark crash when analysing one RTP stream, 2008. https://bugs.launchpad.net/ubuntu/+source/wireshark/+bug/238486.

[5] I. Dacosta et al. Security Analysis of an IP Phone: Cisco 7960G. In *IPTCOMM*, page 255. Springer, 2008.

[6] M. Fannes et al. The mutual affinity of random measures. *Periodica Mathematica Hungarica*, 47(1):51–71, 2003.

[7] GTISC. Emerging Cyber Threats Report for 2009, 2008. http://www.gtiscsecuritysummit.com/pdf/CyberThreatsReport2009.pdf.

[8] D. Hoffman et al. RTP Payload Format for MPEG1/MPEG2 Video. *RFC 2250*, 1998.

[9] Ipoque. Internet Study 2007. http://www.ipoque.com/resources/internet-studies/internet-study-2007.

[10] J.G. Kemeny. *Finite markov chains*. Springer, 1976.

[11] M. Nassar et al. Monitoring SIP Traffic Using Support Vector Machines. In *RAID'08*, pages 311–330. Springer, 2008.

[12] C. Perkins et al. Options for Repair of Streaming Media. *RFC 2354*, 1998.

[13] J.R. Quinlan. *C 4.5: Programs for machine learning*. Morgan Kaufmann Publishers, USA, 1993.

[14] Secunia Advisory SA12478. mpg123 Mpeg Layer-2 Audio Decoder Buffer Overflow Vulnerability, 2004. http://secunia.com/advisories/12478/.

[15] SANS-Institute. SANS Top-20 2007 Security Risks, 2007. http://www.sans.org/top20/.

[16] H. Schulzrinne et al. RTP: A transport protocol for real-time applications. *RFC 1889*, 1996.

[17] Mu Security. Multiple buffer overflows in Asterisk [MU-200803-01], 2008. http://labs.mudynamics.com/advisories/MU-200803-01.txt.

[18] H. Sengar et al. VoIP intrusion detection through interacting protocol state machines. In *DSN'06*, 2006.

[19] H. Sengar et al. Detecting VoIP Floods using the Hellinger Distance. *IEEE Trans. on Parallel and Distributed Sys.*, 19(6):794–805, 2008.

[20] Dug Song. dsniff - Collection of tools for network auditing and penetration testing, 2001. http://www.monkey.org/~dugsong/dsniff/.

[21] S.M. Tabish et al. Malware detection using statistical analysis of byte-level file content. In *ACM CSI-KDD'09*, pages 23–31. ACM, 2009.

[22] The-VoIP-Network. VoIP Market Trends, 2008. http://www.the-voip-network.com/voipmarket.html/.

[23] VOIPSA. Voip Security Tool List, 2010. http://www.voipsa.org/Resources/tools.php.

[24] C. Wieser et al. Security analysis and experiments for Voice over IP RTP media streams. In *SSI'06*, pages 8–10, 2006.

[25] Wireshark-bugs. Wireshark crashes when trying to play RTP stream, 2009. http://www.wireshark.org/lists/wireshark-bugs/200910/msg00227.html.

[26] Y. Wu et al. SCIDIVE: A Stateful and Cross Protocol Intrusion Detection Architecture for Voice-over-IP Environments. *DSN'04*, 2004.