SECURE: Self-Protection Approach in Cloud Resource Management

Sukhpal Singh Gill
The University of Melbourne, Australia

Rajkumar Buyya
The University of Melbourne, Australia

In the current scenario of cloud computing, heterogeneous resources are located in various geographical locations requiring security-aware resource management to handle security threats. However, existing techniques are unable to protect systems from security attacks. To provide a secure cloud service, a security-based resource management technique is required that manages cloud resources automatically and delivers secure cloud services. In this paper, we propose a self-protection approach in cloud resource management called SECURE, which offers self-protection against security attacks and ensures continued availability of services to authorized users. The performance of SECURE has been evaluated using SNORT. The experimental results demonstrate that SECURE performs effectively in terms of both the intrusion detection rate and false positive rate. Further, the impact of security on quality of service (QoS) has been analyzed.

Quality of service (QoS) plays an important role in the era of cloud computing in which delivered cloud services are measured and monitored in terms of QoS to ensure their availability. Nevertheless, offering committed cloud services that guarantee customer's changing QoS needs while precluding them from security attacks is a big challenge.[1] Provisioning and scheduling cloud resources is often done based on their availability without providing the required security.[2] To make cloud computing systems more effective, the security requirements of every cloud component should be satisfied. To realize this, a security-based resource allocation mechanism is required that automatically manages cloud resources and delivers secure cloud services.

Self-protection is the ability of a computing system to defend itself against threats and intrusions. A self-protection component aids in distinguishing and recognizing intimidating behavior and reacts autonomously to protect itself against malicious attacks.[3] These systems defend themselves from attackers by differentiating illegitimate from legitimate behavior and performing the

Copublished by the IEEE CS and IEEE ComSoc
2325-6095/18/$33.00 ©2018 IEEE

required actions to block such attacks without user awareness. Table 1 shows the list of security attacks, from which a system must be self-protected.[2-10]

Table 1. List of Security Attacks

Classification of Attack	Description	Attack Name
Denial of Service (DoS)	Attacker generates a large amount of network traffic, which damages the victim's network (in terms of QoS) by flooding.	SMURF: ICMP (Internet Control Message Protocol) Used to create DoS, in which a pointing packet generates echo requests toward the broadcast IP address. LAND (Local Area Network Denial): Attacker transfers spoofed SYN packet in a TCP/IP network when the destination and source address are the same. SYN Flood: To reduce storage efficiency, an attacker sends IP-spoofed packets to crash the system. Teardrop: Exploits a flaw in the deployments of older TCP/IP stacks.
Distributed-DoS (DDoS)	A DDoS attack occurs when several systems flood the bandwidth or resources of a victim's system, generally one or more Web servers.	HTTP Flood: Attacker exploits seemingly legitimate HTTP GET or POST requests. Zero Day Attack: A security loophole in a cloud based system that is unknown to the developer or vendor.
Remote to Local (R2L)	Attacker executes commands to get access to the system by compromising the network (in terms of QoS).	SPY: Installs itself secretly on a system and runs in the background for phishing. Password Guessing: Attackers guess passwords locally or remotely. IMAP (Internet Message Access Protocol): Finds an IMAP Mail server which is known to be vulnerable.
User to Root (U2R)	To destroy the network, attacker gets root access into the system.	Rootkits: Offers constant privileged access to a system while actively hiding its existence. Buffer Overflow: Occurs when a program copies a large amount of data into a static buffer.
Probing	To breach the personal information of victim, an attacker uses different programming languages.	Ports Sweep: Multiple hosts are scanned for a particular listening port. NMAP (Network MAPper): Performs port scanning.

Recently, researchers focused on identifying new techniques for detection and prevention of intrusions in computing systems and discovered that the Intrusion Detection System (IDS) is an effective way to protect network from attacks. IDS stops attacks, performs recovery after attacks, and investigate security loopholes to help avoid such problems in the future. IDS can be categorized into two types based on anomaly and signature. Signature-based IDS is used to detect the signatures of known attacks in the database, while anomaly-based IDS analyzes abnormal activities. SNORT[13] is the most effective IDS that can be used for attack detection. Different machine learning techniques are used for anomaly-based IDS, but State Vector Machine (SVM) is the most commonly used anomaly-based detector.[2,3]

This article proposes a **S**elf-prot**E**ction approa**C**h in clo**U**d **R**esource manag**E**ment (SECURE) approach for dealing with security attacks. SECURE can create new signatures automatically and provide security against DDoS, Probing, U2R, R2L and DoS security attacks. Based on MAPE-K loop, an algorithm for different phases has been developed to monitor, analyze, plan and execute. SECURE continuously monitors security attacks during the execution of workloads, performs analysis to understand alerts in the case of security attacks, makes a plan to perform required actions to manage threats, and executes the action. Security agents (sensors) are created on SVM as an anomaly detector. SECURE increases the security of cloud-based services and increases intrusion detection rate if the same threat arrives again.

SELF-PROTECTION IN CLOUD: STATE-OF-THE-ART

A Secure Autonomic Technique (SAT) was proposed to provide a secure cloud environment for execution of user applications.[4] SAT detects DoS (SYN Flood) attacks with excellent data accuracy while managing cloud services. Carpen-Amarie[5] proposed the Self-Adaptive Data Management (SADM) system, which manages large amounts of data through the Nimbus cloud environment. SADM also integrates a security policy as part of its framework to detect DoS (Teardrop) security attacks to discover malicious clients. Wailly and colleagues introduced a Virtual Environment based Self-Protecting Architecture (VESPA)[6] to protect cloud infrastructure resources using autonomic security loops. VESPA's performance is measured in terms of response time while detecting U2R (rootkit) security attacks. Sulistio and Reich[7] proposed a Self-Protecting Cloud Service (CPCS) architecture to reduce the barrier for cloud adoption, which decreases Service Level Agreement (SLA) violations. Cloud services of various providers (e.g. Microsoft Azure, Amazon EC2 and Google App Engine) are compared based on the R2L (SPY) security attack for different infrastructure configurations.

To protect a network against malware such as DoS (LAND) security attacks, Benkhelifa and Welsh[8] proposed a mechanism called Malware Inspired Cloud Self-Protection (MICSP), which uses signature analysis to detect malware and prevent the system from future such attacks. Paul[9] proposed a Cloud based Trust Management System (CTMS) to address authentication, authorization and data integrity of cloud security; it also measures the effect of DoS security attacks on the latency introduced during the processing of jobs. Di Pietro and colleagues proposed a Secure Management technique for Virtualized Resources (SMVR),[10] which detects U2R (buffer overflow) security attacks at runtime to improve virtualization security. SMVR reduces security breaching and improves memory utilization. Sarhan and Carr proposed a Self-Protection Data Scheme (SPDS),[11] which uses active data bundles and agent-based secure multi-party computation to protect sensitive data outsources to a cloud for processing. It further provides a secure key management using the RSA algorithm to protect form R2L (IMAP) security attacks.

Table 2 shows a critical comparison of SECURE with existing approaches based on different criteria. The existing resource management techniques considered only one type of the security attacks from DoS, R2L, U2R but all three types of security attacks have not been considered at the same time to the best of our knowledge. Moreover, there is a need to prevent DDoS and Probing security attacks.

Table 2. Comparison of SECURE with Existing Approaches

Year	Technique	Autonomic Mechanism	Type of Attacks	Analyzed Impact of Security on QoS	Performance Parameters
2010	SAT [4]	✓	DoS	✕	Resource Utilization
2011	SADM [5]	✓	DoS	✕	Throughput
2012	VESPA [6]	✓	U2R	✕	Response Time
2013	CPCS [7]	✓	R2L	✕	SLA Violations
2014	MICSP [8]	✓	DoS	✕	Resource Utilization
2015	CTMS [9]	✕	DoS	✕	Latency
2016	SMVR [10]	✕	U2R	✕	Resource Utilization
2017	SPDS [11]	✓	R2L	✕	Resource Utilization
2018	SECURE (Proposed)	✓	Probing, DoS, R2L, U2R and DDoS	✓	Intrusion Detection Rate, Response Time, Signature Generation Rate and False Positive Rate

SECURE protects a cloud-based computing system from five different types of security attacks including DDoS, Probing, U2R, R2L and DoS and analyzes the impact of security on QoS. Further, the performance of SECURE has been tested in terms of response time, false positive rate, and intrusion detection rate. Performance evaluation shows that SECURE performs effectively.

SECURE: SELF-PROTECTION APPROACH IN CLOUD RESOURCE MANAGEMENT

This section presents the architecture of SECURE, which offers self-protection against security attacks. Figure 1 shows the architecture of SECURE and includes the following sub-units:

- Cloud User: Cloud users submit their requests for execution.
- Request Service Handler: A buffer to store information about every request and pass the different workloads to the Dispatcher Service.
- QoS Manager: Identifies the different QoS requirements of a user request and forwards it to the Dispatcher Service.
- Dispatcher Service: Dispatches the user workloads along with their QoS requirements to Detection Engine.
- Detection Engine: Uses two types of IDS, which uses SNORT search for signatures of known attacks in the database (SNORT DB) and uses an SVM-based anomaly detector to analyze abnormal activities (unknown attacks). The training dataset is used to design SVM to find and diagnose input network traffic data to identify the attack. An action is taken once an attack is detected and stored into the database.
- Resource Provisioner: Resources are provisioned using Q-aware,[3] in which suitable resources are identified for a particular workload based on their QoS requirements as designated by QoS Manager. These resources are then provisioned as per the user requests.
- Resource Scheduler: QRSF[12] is used to schedule the provisioned resources with minimum execution cost and time.

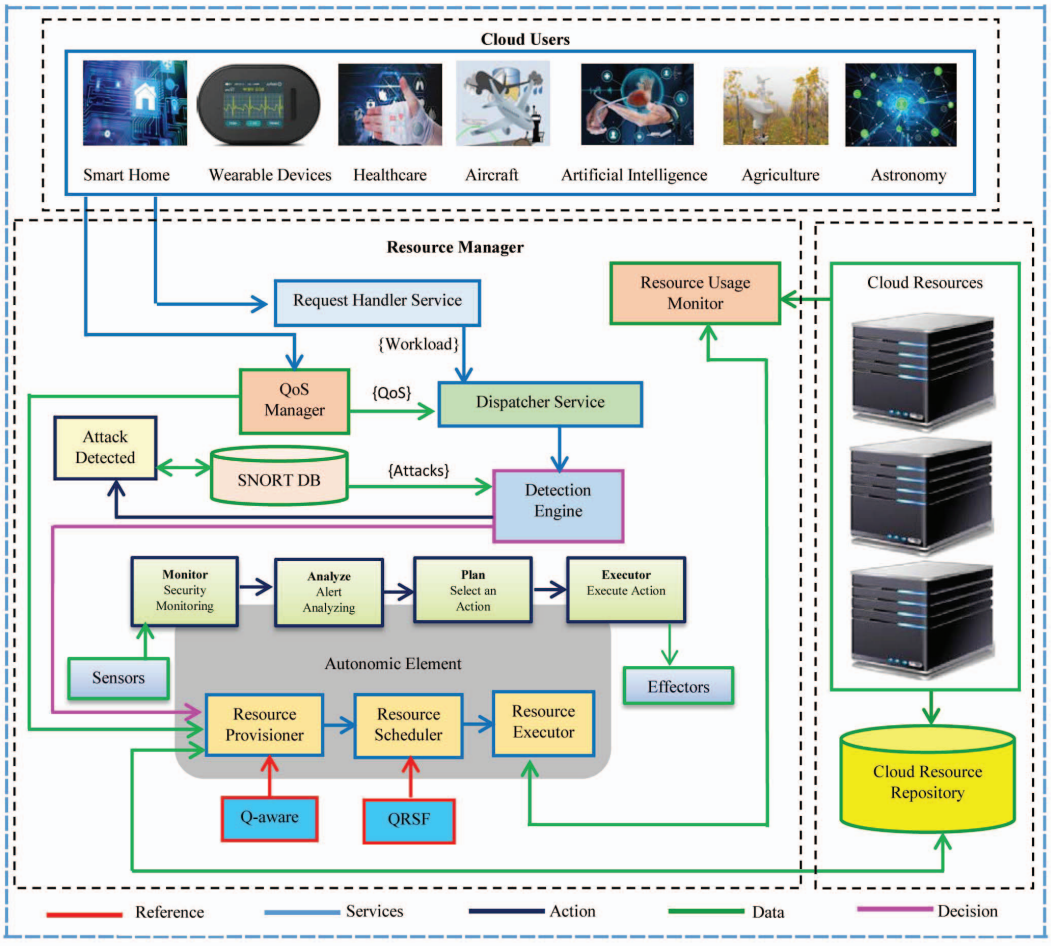

Figure 1. SECURE Architecture.

- Resource Executor: Executes the workloads using the scheduled resources.
- Autonomic Element: Comprises six components: sensor, monitor, analyze, plan, executor, and effector.
- Resource Usage Monitor: Measures the value of resource utilization during workload execution.
- Cloud Resource Repository: Stores the configuration of the cloud resources.

SECURE defends itself from attackers by differentiating illegitimate from legitimate behavior and performing the required actions to block those threats without user awareness. The threats covered by SECURE are DDoS, Probing, U2R, R2L, and DoS. To be efficient, the Detection Engine continuously detects security attacks while processing workloads. Alerts can be stimulated during security attacks. Figure 2 describes the steps of autonomic system i.e. monitoring, analyzing and planning, and execution. During execution of workloads on scheduled resources, *Sensors* detect the value of QoS in terms of response time of task execution. Then, manager node collects information from Sensors and forwards the updated information towards analysis module.

PSEUDOCODE: SELF-PROTECTING

```
# MONITORING
START
Capture Packets
Perform parsing on captured packets
for all Packets
do
  if Packet Payload Length != Range (MIN, MAX) then
    Store packet information into log file
  end if
end for
# ANALYZING and PLANNING
# Process logs
# check for the Security Attacks
Collect all the new alerts generated by AE [Autonomic Element]
for all alerts
  do
    Perform parsing to get URL, Port and Payload detail
    Categorize data based on URL, Port and Payload
    To find largest common substring apply LCS (Longest Common Subsequence)
    Construct new signature by using payload string identified by LCS
end for
# EXECUTION
for all Signature Analyzed [SIGN_ANA]
do
  if SIGN_ANA ⊂ Existing Data then
    Signature merged to existing
  else if SIGN_ANA = Already Existing then
    'IGNORE'
  else
    Add signature as new data
  end if
end for
```

Figure 2. The steps of autonomic system, i.e. monitoring, analyzing and planning, and execution.

The *Monitoring Module* deploys security agents on different computing systems, which trace unknown attacks (using an anomaly-based detector) and known attacks (using a signature-based detector). It captures new anomalies based on existing data stored in the central database (SNORT DB). SECURE captures and detects anomalies using the Intrusion Detection System and labels it as anomalous or normal traffic data by comparing its signatures with the signatures of known attacks.

An SVM-based security agent detects the new anomalies and stores the information into the database to maintain a log about attacks. SECURE protects from security attacks: DDoS (HTTP Flood and Zero-Day Attack), Probing (NMAP and Ports sweep), U2R (Buffer Overflow and Rootkits), R2L (IMAP, Guess password and SPY) and DoS (Teardrop, SYN Flood, LAND and Smurf) as discussed in Table 1. In order to detect a security attack, only those packets will be logged which fits in the range as specified. Detection engine detects the pattern of every packet transferring through the network and compares with the pattern of packets existing in database to find the packet payload length value (range of packet). Alert will be generated if current payload length is out of range [Range (Min, Max)] and attack is detected.

The *Analyzing and Planning Module* analyzes detected attacks and generates a signature for future detection of attacks. Figure 3 shows the functions performed to generate signature.

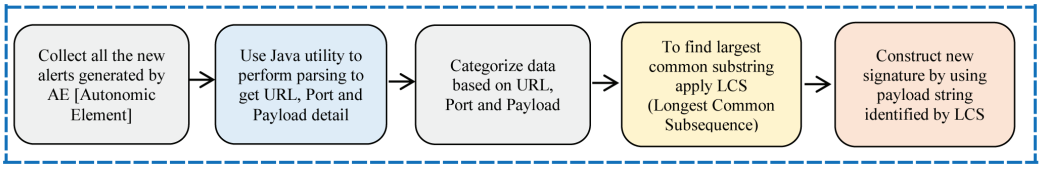

Figure 3. Process of signature generation.

For the *Execution Module*, SNORT IDS refines signatures received from previous modules and compare newly-generated signatures with existing signatures stored in SNORT database. New signatures are added to the SNORT database and new information is merged with existing signatures. Updated information among autonomic elements is exchanged by *Effector,* which communicates updated information about new alerts, rules, and policies.

PERFORMANCE EVALUATION

Figure 4 shows the experimental setup, which is used to evaluate the performance of SECURE. SNORT is a signature-based detector and works on Internet Protocol Networks to examine the real-time network for identification of malicious activity. It generates "analysis signatures" by comparing already stored signatures in the SNORT database and refines, finalizes, and stores new signatures in the SNORT database. SVM is used to detect abnormal behavior (unknown attacks). Different tools (NMAP for probing, DAVOSET for DDoS, NetCat for L2R, Hydra for R2L and metasploit for DoS) are used in this research work to launch different attacks.[1,2,3]

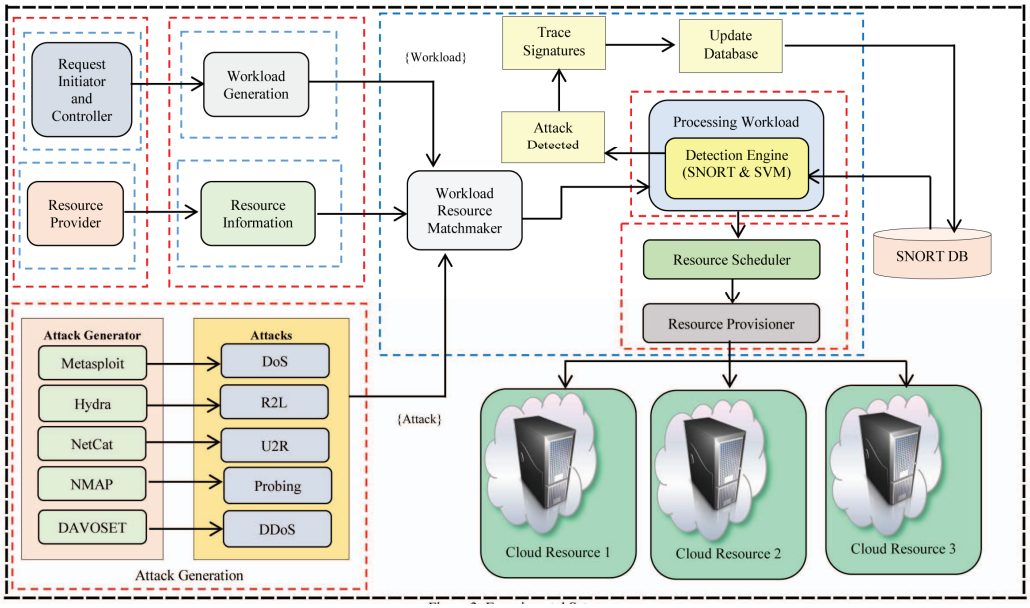

Figure 4. Experimental setup.

Experiments have been performed for five different types of attacks (DDoS, Probing, U2R, R2L and DoS) by comparing SECURE with existing security-aware resource management techniques, i.e. Self-Protection Data Scheme (SPDS).[11] Workload is conversion of larger image file of size 713 MB from JPEG format to PNG format.

Equation 1 describes the **False Positive Rate (FPR)** is described, which is the ratio of *False Positives* to summation of *False Positive and True Negatives.*

$$FPR = \frac{FalsePositives}{FalsePositives + TrueNegatives} \quad (1)$$

FPR decreases in SECURE with time and it is minimum at 50 hours as shown in Figure 5. We have considered five types of attacks (DDoS, Probing, U2R, R2L and DoS) and measured the value of FPR for each attack. The value of FPR is higher for R2L as compared to DDoS, Probing, U2R and DoS attacks.

Equation 1 describes the **Intrusion Detection Rate (IDR)**, which is the "ratio of total number of true positives to the total number of intrusions."

$$IntrusionDetectionRate = \frac{TotalNumberofTruePositives}{TotalNumberofIntrusions} \quad (2)$$

IDR considers the number of detected and blocked attacks and its value increases with respect to time. To avoid the same attack, new signatures are stored into the database continuously. For known attacks, this experiment has been conducted. Figure 6 clearly shows that SECURE gives better results as compared to SPDS in terms of IDR. Further, for more verification of SECURE, the signatures of some known attacks have been removed from the SNORT database.

Figure 7 shows that IDR is increasing with respect to time. An experiment for 144 hours has been conducted for verification of SECURE and the results show that SECURE performs better than SPDS in terms of IDR and SECURE performance is outstanding after 120 hours. Figure 8 shows the variation of IDR with respect to different numbers of workloads and different types of security attacks. With the variation of the number of workloads, the value of IDR is also rising. Figure 8 shows that SECURE performs better in probing.

Signature Generation Rate (SGR) is defined as the percentage of signatures generated over time. Figure 9 shows how SGR is calculated for both SECURE and SPDS and it shows that SECURE has higher signature generation rate than SPDS.

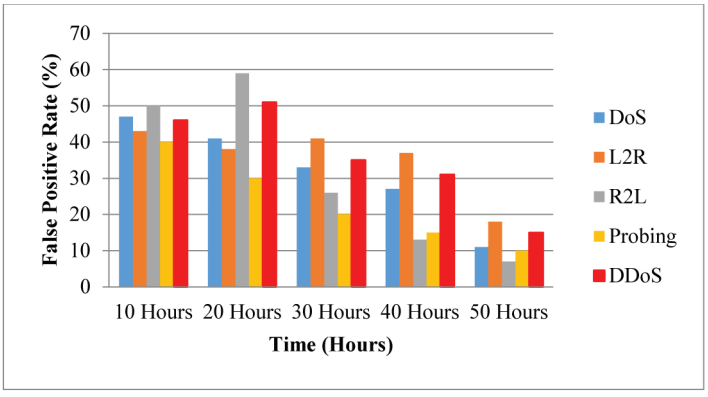

Figure 5. False Positive Rate (FPR) vs. time.

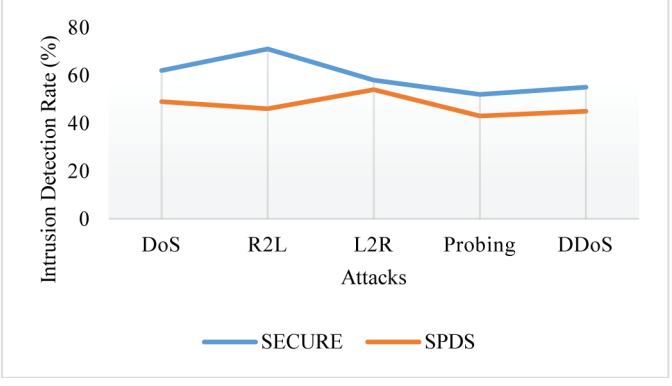

Figure 6. Intrusion Detection Rate (IDR) vs. attacks

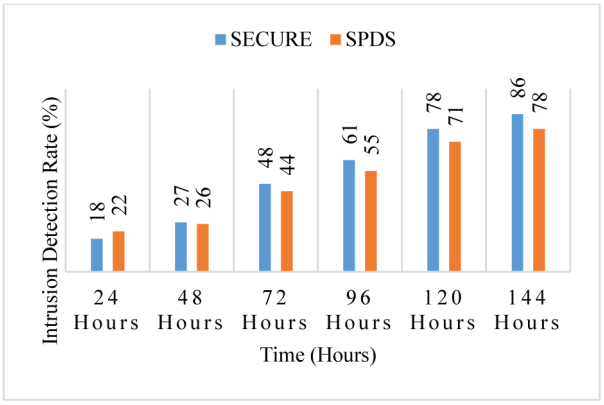

Figure 7. Intrusion Detection Rate (IDR) vs. time

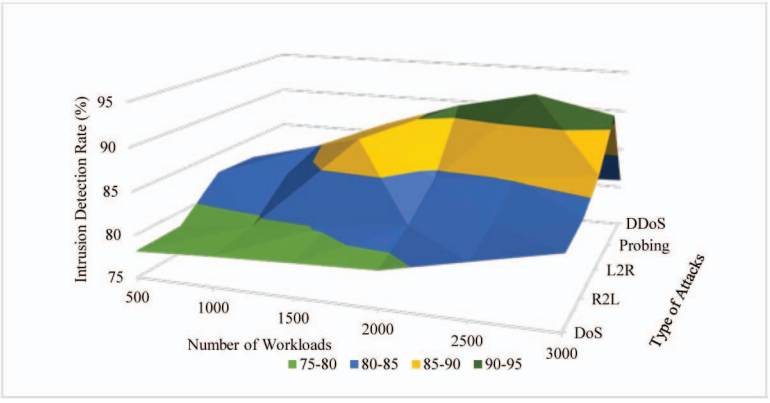

Figure 8. Intrusion Detection Rate (IDR) vs. attacks

Zero Day Attack: An Analysis

This attack denotes a security loophole in a cloud-based system that is unknown to the developer or vendor, known as zero-day attack. To perform this attack, hackers release malware before creating a patch to fix this vulnerability. To handle cloud specific zero-day attacks, a layered architecture is used to implement, which contains three different layers: i) detection layer, ii) analysis layer and iii) configuration layer as shown in Figure 10.

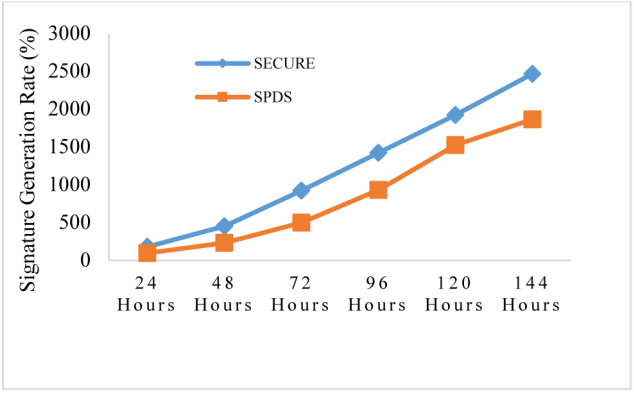

Figure 9. Signature Generation Rate (SGR) vs. time.

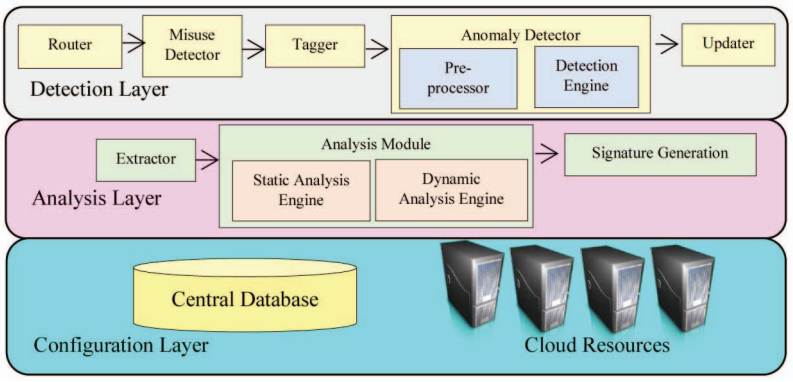

Figure 10. Layered architecture for detection of a zero day attack.

In *Detection Layer, Router* provides the input traffic data to detect unknown attacks. *Misuse Detector* models abnormal behavior because it contains the set of rules for different malicious behavior. SNORT[13] filters all the known attacks through signature matching, throws away all the known attacks, and passes the filtered network traffic. *Tagger* performs traffic tainting, in which it monitors the traffic and tags it using format *{arrival_time, source_IP, destination_IP, destination_port, Protocol}* and transfers it to *Anomaly Detector* for further processing. *Anomaly Detector* detects the unknown attacks, which are not detected by *Misuse Detector* and the pre-processor extracts features (destination bytes, source bytes, count, flag, source error rate, protocol type, and destination error rate) from tagged traffic and stores it into the log file. *Detection Engine* receives parsed traffic from the pre-processor and 1-class SVM compares this traffic with existing good traffic profile (from trusted subnet) to find zero-day attacks. *Updater* receives output from *Anomaly Detector* and forwards it to *Central Database* for future attack detection.

Analysis Layer analyzes the behavior of detected zero-day attacks. *Extractor* receives doubtful packets and parses them to detect the location using the current header length and the type of next header. The extracted data is stored as a binary file and forwarded to *Analysis Module*. Analysis Module consists of Static Analysis Engine and Dynamic Analysis Engine to merge dynamic and static functionalities to detect malware behavior. *Static Analysis Engine* analyzes the binary file to describe static features such as anti-virus scanning (to detect malicious attacks) and obfuscation (to evade detection systems). In our work, packing is used as an obfuscation technique to detect packer signatures. *Dynamic Analysis Engine* analyzes the behavior of binary files while executing and records network statistics and forwards it to next module. *Signature Generation* uses ClamAV format (*Signature = <HashString: FileSize:MalwareType>*) to generate new signatures from binary files and store them in the central database for future attack detection.

Configuration Layer contains information about cloud resources (for data processing) and the central database (to store the information about known and detected zero-day attacks). Further, Netbeans 7.0, MySql Database and Oracle Java 7 are used to evaluate the performance. The implementation system consists of different components: intranet machines, ethernet switch, IDS/IPS sensor and router. In total, 490 malware samples are used, both non-obfuscated and obfuscated, to measure the data accuracy in terms of False Positive Rate Figure 11 shows. Tools such as Metasploit, Hydra, Netcat, DAVOSET and NMAP are used to generate malware samples.[1,2,3]

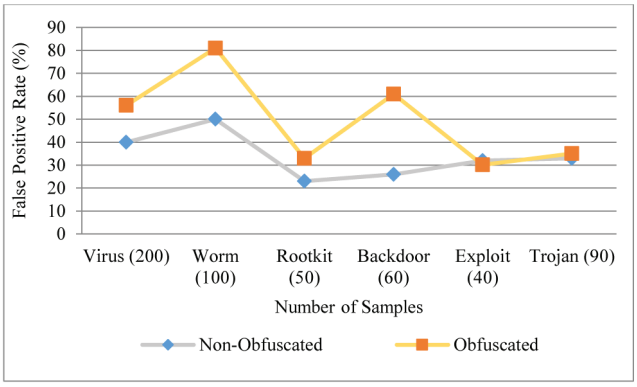

Figure 11. False Positive Rate vs. the rate of accuracy for a zero-day attack.

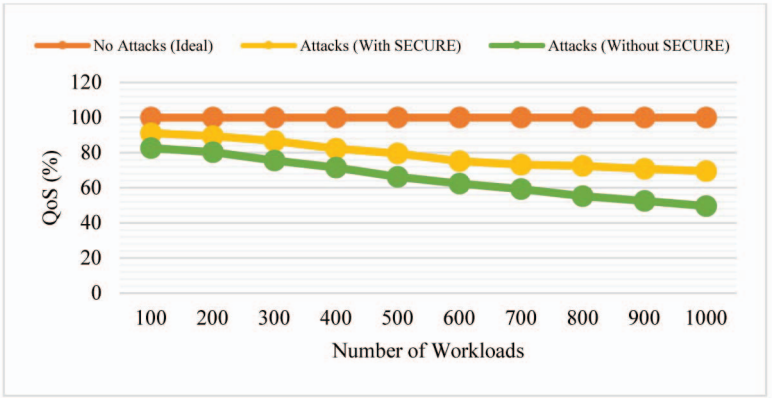

Figure 12. Impact of security on QoS (response time).

Security and QoS: Intertwined in Self-Protection of Cloud

This section analyzes the impact of Security on QoS. The *response time* is considered as a QoS parameter, which is the amount of time required to execute the workload completely and produce the required output. We have defined three different types of scenarios to measure the value of response time that Figure 12 shows. Three different scenarios follow:

- No Attack: In this scenario, there is no attack and the system executes the workloads successfully within the required response time.
- Attack Without SECURE: In this scenario, attacks occur, affecting the response time of different workloads and reducing the QoS value.
- Attack With SECURE: In this scenario, attacks occur, affecting the response time of different workloads, but SECURE improves the QoS value by detecting and neutralizing the attacks.

CONCLUSION AND FUTURE DIRECTIONS

We propose the SECURE approach for self-protection against security attacks. SECURE protects the system execution from five different types of security attacks including DDoS, Probing, U2R, R2L, and DoS, and analyzes the impact of security on QoS during the processing of user requests. Further, the performance of SECURE is tested in terms of intrusion detection rate, response time and false positive rate. Experimental results show that the performance of SECURE is better than existing techniques, which provide secure cloud based services by protecting from security attacks.

To further advance the work, we propose the following future directions:

- A practical realization of SECURE on a real cloud environment.
- SECURE can be extended to work with some other attacks also ransomware, NTP Amplification, slowloris, UDP Flood etc.
- The detection of zero-day attack can be improved by locating the source of the attack by analyzing the anomalous behavior patterns.
- Detection and analysis of multiple zero-day binaries simultaneously can improve throughput.
- Multiple execution path can be explored for effective malware analysis for detection of zero-day attack.
- Signatures for zero-day binaries can be generated in a more detailed manner using SNORT format.

ACKNOWLEDGEMENTS

We thank Arash Shaghaghi (The University of New South Wales, Australia) for comments on improving the paper.

REFERENCES

1. M.B. Mollah, M.A. Azad, and A. Vasilakos, "Security and privacy challenges in mobile cloud computing: Survey and way ahead," *Journal of Network and Computer Applications*, vol. 84, 2017, pp. 38–54.
2. S.S. Gill et al., "CHOPPER: an intelligent QoS-aware autonomic resource management approach for cloud computing," *Cluster Computing*, 2017, pp. 1–39.
3. S. Singh and I. Chana, "Q-aware: Quality of service based cloud resource provisioning," *Computers & Electrical Engineering*, vol. 47, no. C, 2015, pp. 138–160.
4. G. Qu, O.A. Rawashdeh, and D. Che, "Self-Protection against Attacks in an Autonomic Computing Environment," *International Journal of Computer Applications* (CAINE), vol. 17, no. 4, 2010, pp. 250–256.
5. A. Carpen-Amarie, "Towards a self-adaptive data management system for cloud environments," *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum* (IPDPSW 11), 2011, pp. 2077–2080.
6. A. Wailly, M. Lacoste, and H. Debar, "Vespa: Multi-layered self-protection for cloud resources," *Proceedings of the 9th International Conference on Autonomic Computing* (ICAC 12), 2012, pp. 155–160.
7. A. Sulistio and C. Reich, "Towards a Self-Protecting Cloud," *On the Move to Meaningful Internet Systems: OTM 2013 Conferences.*, Springer, 2013, pp. 395–402.
8. E. Benkhelifa and T. Welsh, "Towards Malware Inspired Cloud Self-Protection," *Proceedings of the 2014 International Conference on Cloud and Autonomic Computing* (ICCAC 14), 2014, pp. 1–2.
9. P. Manuel, "A trust model of cloud computing based on Quality of Service," *Annals of Operations Research*, vol. 233, no. 1, 2015, pp. 281–292.
10. R.D. Di Pietro, F. Lombardi, and M. Signorini, "Secure Management of Virtualized Resources," *Security in the Private Cloud*, CRC Press, 2016; doi.org/10.1201/9781315372211-14.
11. A.Y. Sarhan and S. Carr, "A Highly-Secure Self-Protection Data Scheme in Clouds Using Active Data Bundles and Agent-Based Secure Multi-party Computation," *Proceedings of the IEEE 4th International Conference on Cyber Security and Cloud Computing* (CSCloud 17), 2017, pp. 228–236.
12. S. Singh and I. Chana, "QRSF: QoS-aware resource scheduling framework in cloud computing," *The Journal of Supercomputing*, vol. 71, no. 1, 2015, pp. 241–292.
13. B. Caswell and J. Beale, *Snort 2.1 Intrusion Detection*, Syngress, 2004.

ABOUT THE AUTHORS

Sukhpal Singh Gill is a Postdoctoral Research Fellow with the University of Melbourne's Cloud Computing and Distributed Systems (CLOUDS) Laboratory. Contact him at Sukhpal.gill@unimelb.edu.au. For further information, please visit www.ssgill.in.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is one of the most highly cited authors in computer science and software engineering worldwide (h-index=115 and 70,000-plus citations). He was recognized as a "Web of Science Highly Cited Researcher" in 2016 and 2017 by Thomson Reuters. Buyya is a Fellow of IEEE, and Scopus Researcher of the Year 2017 with an Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing. Contact him at Rbuyya@unimelb.edu.au. For further information, please visit his www.buyya.com.