Table of contents

Introduction

Threats definition
  Threat models
    STRIDE (Microsoft)
    OCTAVE (CERT)
    CAPEC (MITRE)
    OWASP Top-10
  Cloud tenant threats
  Cloud provider threats
  Targeted attacks and APTs

Defensive techniques
  Mitigating threats

Incident response

Mirantis OpenStack security
  Encryption strategies
  Key and certificate management
  Host security
    Mandatory access control security enhancement
    Rootwrap
  Compute and hypervisor security
  Virtual consoles
  Compute hardware platform security
  Images
  Containers

Networking
  Network equipment
  Network services
    Networking service
    OpenContrail

CLI and API

Messaging

Identity
  Authentication
    External authentication
    Multi-factor authentication
Tokens
Domains
Groups
Identity Federation
Authentication middleware
PKI and cryptography service (Barbican)
Storage
  Block storage
  Object storage
  Ceph
OpenStack Dashboard
Fuel Master node
Monitoring and incidents detection
Security features enabled in Mirantis OpenStack
Mirantis OpenStack updates and configuration management
  Mirantis OpenStack Security Updates
  Configuration Management
Cloud security architecture
  Demilitarized zone
  Security domain
Cloud security solutions
  WAF and load balancers
  Reverse proxies
  DDoS protection
  Firewall-as-a-Service
Intrusion detection and prevention systems
  Network-based IDPS
  Host-based IDPS
  IDPS acceleration
Next-generation solutions
Cloud antivirus
  Agent-based
  Agentless
Forensics solutions
Use cases
  Palo Alto Networks VM-series deployment as a VNF on Mirantis OpenStack
  IDPS Suricata deployment as a VNF on Mirantis OpenStack with OpenContrail
Deploy Mirantis OpenStack with OpenContrail SDN
Install IDPS
IDS mode
IPS mode
File extraction in Suricata
Additional Suricata configuration
Alerts management with Snorby and Barnyard2
IDPS as a VNF
Incident detection and prevention with IDPS and malware sandbox
Modeling targeted attacks

Appendix A
Compliance with security standards
Recommended security solutions
IT host security policy
  STIG hardening recommendations
  FedRamp recommendations
  PCI DSS recommendations
Examples
  Example of incident investigation and analysis of CozyDuke APT
  Example of threat modeling for Ceph RBD
Introduction

Mirantis Security Best Practices covers the following topics:

- recommendations on secure configuration of Mirantis OpenStack components
- description of typical threats that may affect a customer's cloud
- threat modelling techniques
- references to security standards
- recommendations on threats mitigation

Mirantis OpenStack is a set of hardened OpenStack packages. Mirantis engineers put efforts to make the components more secure and to deploy architecture capable to withstand with cyber threats.

This guide starts with explaining cyber attacks models used for threat modeling. Considering these models you can understand against which type of threats you need to protect your cloud. We run through the most popular threat models: STRIDE from Microsoft, OCTAVE from CERT, and CAPEC from MITRE. In addition, we mention cloud specific threats and affected objects.

The next chapter describes general mitigation techniques for the threat models mentioned earlier such as using cryptography for digital signing and encryption of data and communications, access controls, logging, and so on. The security tools mentioned in this chapter can help you to implement the proposed techniques to mitigate corresponding threats.

Some components and hosts may need additional configuration after deployment depending on current environment and/or your specific needs. The “Fuel and Mirantis OpenStack Security” chapter describes these aspects. Additionally, you can refer to OpenStack Security Guide where you can find more information explaining the reasons to configure a service or host in the way we recommend.

The next chapter guides you through the best practices of designing a secure cloud architecture including DMZ and installation of security solutions on top of Mirantis OpenStack to help with incident detection, prevention, and investigation processes. At the end, we propose several use cases that will help you to address given recommendations.

Threats definition

Before stepping into recommendations we want to give an overview of available threat models to define possible attack vectors and suggest mitigation techniques before cloud deployment or in a process of environment configuration.
Threat models

There are three different approaches to threat modeling focusing on:

- software
- assets
  - things you protect
  - stepping stones
  - things attackers want
- attacks and attackers

We will consider three threat models proposed by Microsoft, CERT, and MITRE depending on what you are going to focus on when deploying Mirantis OpenStack environment. Let us consider these models first to be able to recommend mitigation techniques for every class of threats in the next chapters.

**STRIDE (Microsoft)**

STRIDE model focuses on software. We recommend using [Microsoft Threat Modeling Tool](https://docs.microsoft.com/en-us/azure/security/microsoft-threat-modeling-tool) when planning your environment to model potential threats you might have in future when running your cloud. This may affect architectural solutions and change a deployment scenario.

In STRIDE there are six classes of threats corresponding with the letters in the abbreviation.

<table>
<thead>
<tr>
<th>Threat Class</th>
<th>Description</th>
<th>Affected Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spoofing</td>
<td>Pretending to be something or someone other than yourself</td>
<td>Process, file, host, account, certificate/TLS-protected session</td>
</tr>
<tr>
<td>Tampering</td>
<td>Modifying something on disk, on a network, or in memory</td>
<td>File, memory, data store, data flow, network, cache</td>
</tr>
<tr>
<td>Repudiation</td>
<td>Claiming that you didn’t do something, or were not responsible.</td>
<td>Attack to logs, sources of time synchronization</td>
</tr>
<tr>
<td>Information Disclosure</td>
<td>Providing information to someone not authorized to see it</td>
<td>From process, storage, network, cache</td>
</tr>
<tr>
<td>Denial of Service (DoS)</td>
<td>Absorbing resources needed to provide service</td>
<td>Service</td>
</tr>
<tr>
<td>Elevation of Privileges (EoP)</td>
<td>Allowing someone to do something they are not authorized to do</td>
<td>Process and authorization exploits</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------</td>
<td>---------------------------------</td>
</tr>
</tbody>
</table>

This guide will refer to STRIDE as a primary threat model.

See also:

**OCTAVE (CERT)**

OCTAVE (Allegro) model focuses on information assets and allows you to perform risk assessment. It consists of eight steps:

1. Establish Risk Measurement Criteria
2. Develop an Information Asset Profile
3. Identify Information Asset Containers
4. Identify Areas of Concern
5. Identify Threat Scenarios
6. Identify Risks
7. Analyze Risks
8. Select Mitigation Approach

These steps are organized into four phases:

1. Develop risk measurement criteria consistent with the organization's mission, goal objectives, and critical success factors.
2. Create a profile of each critical information asset that establishes clear boundaries for the asset, identifies its security requirements, and identifies all of its containers.
3. Identify threats to each information asset in the context of its containers.
4. Identify and analyze risks to information assets and begin to develop mitigation approaches.

See also:

**CAPEC (MITRE)**

The Common Attack Pattern Enumeration and Classification (CAPEC) model provides comprehensive threat classification and focuses on mechanisms and vectors of attacks.

See also:
- CAPEC model [https://capec.mitre.org/data/index.html](https://capec.mitre.org/data/index.html)
You can find an example of Ceph RBD threat modelling using STRIDE in Use cases.

**OWASP Top-10**

Open Web Application Security Project (OWASP) provides information on top-10 cloud threats:

1. Accountability & Data Risk
2. User Identity Federation
3. Regulatory Compliance
4. Business Continuity & Resiliency
5. User Privacy & Secondary Usage of Data
6. Service & Data Integration
7. Multi-tenancy & Physical Security
8. Incidence Analysis & Forensics
9. Infrastructure Security
10. Non-production Environment Exposure

See also:
- [OWASP Cloud Top-10 project](#)

**Cloud tenant threats**

Threats to tenants may come from a cloud provider (insider threats) or another tenant (co-tenant threats).

Insider threats (from a cloud provider):
- OpenStack services misconfiguration may lead to EoP
- A failure in maintenance. For example, not wiping disks on nodes between re-allocations leads to Information Disclosure.
- Improper configuration of security services or turning them off when high loaded. For example, disabling rules and taking protocols out of scan by IDPS may lead to missing the attack (EoP).
- Connecting VMs to the management network may lead to Information Disclosure and, as a result, EoP.

Mitigate insider threats in two ways:
- Contractually - negotiate agreements related to privacy, security, and reliability, even though, it may increase costs (Information Disclosure).
- Cryptographically - encrypting data on a cloud storage and when transferring through the network channel (Information Disclosure).
- Isolate the management network from tenants’ networks (Information Disclosure).

Co-tenant threats
● Another tenant might try to escape a VM and take over the host (EoP).
● Getting access to shared resources such as storage, network, and so on (EoP, Information Disclosure, Availability, Tampering).
● Another tenant might be taken over to run a DoS attack (EoP, Spoofing, DoS).
● Brute-force and dictionary attack (EoP).
● Shared cloud provider's infrastructure such as:
  ○ a shared mail service may lead to spear-phishing attacks from one tenant to another (Spoofing),
  ○ a shared DNS service may led to DNS poisoning attack (Spoofing, Tampering)
  ○ a shared Web service such as cloud admin web interfaces may be a source of XSS, CSRF, SQL injection, and so on attacks. (EoP)

Mitigate co-tenant threats
● Provide cloud separation. Use Host Aggregation and Availability Zones to separate VMs with different security level. (EoP, Tampering, Information Disclosure, Availability)
● Use nodes from Trusted Computing Pool based on Intel TXT - a technology designed to harden platforms from the emerging threats of hypervisor attacks, BIOS, or other firmware attacks, malicious rootkit installations, or other software-based attacks. (EoP)
● Brute-force protection. Lock out an attacker's logins after repeated failures. (EoP)
● Change default passwords (EoP).
● Use network IDPS to monitor and detect anomalies in management and tenants networks (Information Disclosure, EoP).

Cloud provider threats

A tenant to hack the provider:
● a tenant may run out of a VM or container using security breaches and get access to management network (EoP)
● a fraud tenant can sign up using stolen credentials, for example, to organize a botnet, bitcoin miner, or Command-and-Control server that will be paid by the victim (EoP, Repudiation, Spoofing)
● brute-force and dictionary attacks (EoP)
● resource exhaustion (DoS)

Malicious tenant behaviour that leads to blacklisting or loss of reputation of a cloud provider may include:
● Outgoing DDoS attacks
● Spamming
● Mining Bitcoins
● Distributing malware, pirated, or other illegal content
Outsider threats:
- Targeted Attacks (EoP)
- DDoS
- Human-related threats:
  - insider access (EoP)
  - social engineering (Spoofing, EoP)
- 3-d parties access (Information Disclosure)
- MITM (Information Disclosure) and DoS attacks using BGP exposed to Internet access
- Vulnerabilities in network devices (EoP)

Mitigate cloud provider threats:
- Brute-force protection - lock out an attacker’s logins after repeated failures. (EoP)
- Disable indexing by search engines using ‘robots.txt’ or similar for public administrative interfaces. (EoP)
- Change default passwords. (EoP)
- Use WAF to limit access to admin interfaces. (EoP)
- Use network IDPS to monitor and detect anomalies in management and tenants networks. (EoP, Information Disclosure, Spoofing, Repudiation)
- Enable logging to trace EoP attempts and mitigate repudiation attacks. (EoP, Repudiation)
- Disable indexing by search engines using ‘robots.txt’ or similar for public administrative interfaces. (Information Disclosure)
- Enable BGP peer filtering. (Information Disclosure, EoP)
- Enable vulnerability management (EoP).

See also:

**Attack surface**
The more public interfaces the system has, the larger attack surface becomes, and the more it is exposed to external attacks. Minimize the attack surface to save efforts on protecting it against external attacks. To do that, place your API endpoints behind a trust boundary such as Firewall or DMZ.

See also:
Targeted attacks and APTs

Modern cyber attacks happen now via a set of cyber espionage processes called Advanced Persistent Threats (APTs) that are capable to run silently for a long period of time collecting specific information on a victim’s computer or network.

Nowadays, attackers do not try to penetrate a security perimeter in a straightforward manner by scanning and exploiting found vulnerabilities, as it may attract too much attention to the attack and it will be blocked in a matter of minutes. They prefer using more sophisticated techniques based on social engineering that allow a spy program to operate in hidden way for unlimited amount of time not attracting extra attention from a victim and having an ability to harvest sensitive information and send it to a Command and Control server (C&C).

The most popular techniques used in targeted attacks are spear phishing emails, watering hole attacks, and 0-day exploits. When attackers run targeted attack they know pretty well who the victims are and how a targeted environment looks like.

The general APT model might look like:

1. Reconnaissance
2. Penetration using:
   a. spear-phishing
   b. watering hole
   c. USB removable storage
3. Delivery of the APT kit
4. Lateral movements and EoP
5. Data collection
6. Data exfiltration

For example, some of the enterprise attacks run with APT are designed to infect SCADA systems such as WinCC, which was successfully infected by Stuxnet in 2010 at Iran’s nuclear facilities. Still many cases are kept secret by owners and stakeholders of enterprises, as it may significantly harm a business reputation.

The attackers may choose someone, for example, from an accounting department to attack, as such person seems to be a low hanging fruit to pick. Next, having a backdoor installed behind a company’s security perimeter, penetration testing tools can be utilized to scan a corporate network for other victims. To guarantee a high proliferation rate within a network 0-day exploits
can be used. For example, Stuxnet was equipped with CVE-2010-2729/MS10-061 0-day vulnerability in the Printer Spooler Service to spread itself in a local network.

In case of using 0-day exploits and polymorphic malware, traditional security solutions such as Firewall and IDPS using a standard set of rules can not recognize targeted attacks because they are aimed to detect known general purpose attacks.

See also:
- Use cases: “Office Monkeys” APT Incident Response

**Defensive techniques**

The mitigating techniques presented in this chapter are mostly based on STRIDE as we focus on Mirantis OpenStack, which is a software product. Each threat class is represented with a corresponding set of mitigation techniques and recommended tools. The table below represents information about threats and mitigation techniques based on the STRIDE model.

**Mitigating threats**

<table>
<thead>
<tr>
<th>Violates</th>
<th>Threat Type</th>
<th>Mitigation</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Spoofing</td>
<td>PKI:</td>
<td>• Secrets Manager (Barbican)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TLS and certificates</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Digital signatures</td>
<td></td>
</tr>
<tr>
<td>Integrity</td>
<td>Tampering</td>
<td></td>
<td>• SELinux, AppArmor, grsecurity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Identity Federation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Secrets Manager (Barbican)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-repudiation</td>
<td>Repudiation</td>
<td></td>
<td>• LMA toolchain</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Keystone’s CADF events</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Information Disclosure</td>
<td></td>
<td>• Volume encryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Ephemeral disk encryption in LVM format</td>
</tr>
<tr>
<td>Availability</td>
<td>Denial of Service</td>
<td>Object encryption</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
<td>--------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Secrets Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Barbican)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SELinux,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AppArmor,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>grsecurity</td>
<td></td>
</tr>
<tr>
<td>Availability Zones</td>
<td></td>
<td>Firewall (layer 3,4,7)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Load Balance</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DDoS protection</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Availability Zones in OpenStack</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Authorization</th>
<th>Elevation of Privilege</th>
<th>SELinux,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AppArmor,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>grsecurity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Identity Federation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DMZ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privacy</th>
<th>Information Disclosure</th>
<th>MAC/RBAC,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Encryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SELinux,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AppArmor,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>grsecurity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Volume encryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ephemeral disk encryption in LVM format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Object encryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Secret Manager(Barbican)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DLP</td>
</tr>
</tbody>
</table>

**Cloud Threats**

<table>
<thead>
<tr>
<th>Cloud Tenant Security</th>
<th>Insider Threats (Information Disclosure, Spoofing)</th>
<th>Contractually</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MAC/RBAC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data encryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Information Disclosure)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Isolate the management network from tenant’s networks (Information)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SELinux,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AppArmor,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>grsecurity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Volume encryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ephemeral disk encryption in LVM format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Object encryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Secret Manager(Barbican)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DLP</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Brute force protection (EoP)</td>
<td>Cloud separation using HA and AZ to avoid running VMs with different security level on the same Compute node (EoP, Information Disclosure, DoS)</td>
<td>Cloud provider to create DMZ (EoP)</td>
</tr>
<tr>
<td>Cloud separation using HA and AZ to avoid running VMs with different security level on the same Compute node (EoP, Information Disclosure, DoS)</td>
<td>Increasing staff security awareness (Spoofing, EoP)</td>
<td>Firewall (layer 3,4,7), Load Balancer, DDoS protection, Sandbox</td>
</tr>
<tr>
<td>Outsider Threats (EoP, Information Disclosure):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Targeted attacks (EoP)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDoS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Human-related threats: insider access, social engineering (Spoofing, EoP)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-d parties access (Information Disclosure)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cloud Provider Security</td>
<td>Tenants hack the provider:</td>
<td>Outsider threats:</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>• Running out of VM and get access to the management interface or network (EoP)</td>
<td>• Targeted attacks (EoP)</td>
</tr>
<tr>
<td></td>
<td>• Using stolen another tenant’s credentials (EoP, Spoofing, Repudiation)</td>
<td>• DDoS</td>
</tr>
<tr>
<td></td>
<td>• Brute-force and dictionary attacks (EoP)</td>
<td>• Human-related threats: insider access, social engineering (Spoofing, EoP)</td>
</tr>
<tr>
<td></td>
<td>• Resource exhaustion (DoS)</td>
<td>• 3-d parties access (Information Disclosure)</td>
</tr>
<tr>
<td></td>
<td>• Brute force protection (EoP)</td>
<td>• MITM (Information Disclosure) and DoS attacks using</td>
</tr>
<tr>
<td></td>
<td>• Limit access to admin interfaces (EoP)</td>
<td>• DMZ (EoP)</td>
</tr>
<tr>
<td></td>
<td>• Change default passwords (EoP)</td>
<td>• Increasing staff security awareness (Spoofing, EoP)</td>
</tr>
<tr>
<td></td>
<td>• Monitor and detect anomalies in management and tenants’ networks (EoP, Information Disclosure)</td>
<td>• BGP peer filtering (Information Disclosure, EoP)</td>
</tr>
<tr>
<td></td>
<td>• Disable indexing by search engines (Information Disclosure)</td>
<td>• Firewall (layer 3,4,7)</td>
</tr>
<tr>
<td></td>
<td>• Logging (Repudiation)</td>
<td>• Load Balancer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DDoS protection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sandbox</td>
</tr>
</tbody>
</table>

- WAF
- IDPS
- LMA toolchain
- MOS brute force protection
- DMZ (EoP)
- Increasing staff security awareness (Spoofing, EoP)
- BGP peer filtering (Information Disclosure, EoP)
- Firewall (layer 3,4,7)
- Load Balancer
- DDoS protection
- Sandbox
<table>
<thead>
<tr>
<th></th>
<th>BGP exposed to Internet access</th>
<th>Logging</th>
<th>Setting Security Domain/Project with forensic tools</th>
<th>LMA toolchain Network sniffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Forensic data</td>
<td>Forensic threats</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See also:
- SELinux on CentOS [https://wiki.centos.org/HowTos/SELinux](https://wiki.centos.org/HowTos/SELinux)
- SELinux on Debian and Ubuntu [https://wiki.debian.org/SELinux/Setup](https://wiki.debian.org/SELinux/Setup)
Incident response

The following chapter provides incident response procedure. Incident response procedure describes a set of steps to be performed by the incident response team (IRT) when an information security incident happens within an organization. Incident response aims at revealing the intruder, mitigating the damage, recovering and preventing further penetration.

Typically, incident response procedure includes the following stages:

1. Preparation.
   a. See the recommendations below.

2. Detection.
   a. A user or installed security system such as IDS, firewalls, or sandbox report a potential issue.

3. Containment.
   a. Minimize damage, prevent wiping compromised systems to take forensic images and other digital evidence.
   b. To isolate the compromised VMs or project, temporarily switch them from the Internet to the Security Domain for further investigation.

4. Investigation.
   a. IT service collects incident-related data, such as network traffic, files, and logs, and deliver it to IRT.
   b. Analysis. IRT begins threat analysis using data gathered by the IT service to report recommendations on mitigating the security issue, remediation, and future prevention.
   c. IRT writes the recommendations to IT service, for example:
      i. how to remove malicious code and signs of its presence on the infected hosts and/or VMs
      ii. what password should be changed if any
      iii. what keys should be regenerated if any
      iv. what certificates should be revoked if any

5. Remediation.
   a. IT service removes infection and change passwords, generate new keys.
   b. IT service can recover hosts, VMs, or network devices from backups reverting changes made by malware.
   c. IT service can scan the recovered VMs, hosts, and networks with IDS updated and restarted with new rules and a vulnerability scanner to discover possible breaches.
   d. IT service can get the affected VMs and project back to operation.

   a. IRT writes recommendations to IT service describing incident prevention steps, for example:
      i. revise enabled protocols
ii. install security updates to address vulnerabilities
iii. update IDS, firewalls, and sandbox with new rules based on mined IoCs

7. Lessons learned.
   a. Write an incident report.
   b. Analyse IRT performance.
   c. Write missing documentation.
   d. Organize lessons learned meeting within two weeks after the incident covering the following topics:
      i. Who and when detected the problem.
      ii. The scope of the incident.
      iii. How it was contained.
      iv. Data collected during the investigation.
      v. Work performed during analysis.
      vi. Remediation steps.
      vii. Areas that need improvement.

Recommendations:
- Create a plan or strategy to handle incidents.
- Create IRT, which may include IT and security specialists, as well as an attorney, PR, and HR specialists.
- For access control, add a system administrator to IRT to adjust permissions for IRT accounts during incident handling.
- Prepare software and hardware tools for incident handling. As an option, you can create a Security Domain in your cloud that may contain network sniffers, malware scanners, debuggers, and a sandbox. Once an incident happens, you can switch the affected project (tenant) from the Internet to the Security Domain so the network traffic will go through network scanners and the suspicious files extracted from the traffic can be analyzed in a sandbox.
- Allocate storage for forensic dumps of compromised VMs and hosts.
- Prioritize incidents based on organizational impact, which will determine resources allocated for IRT.
- Create a communication plan to know who to contact during an incident and why. Create a contact list of IRT members.
- Document an incident. IRT should use Incident Handlers Journal to record any actions performed during incidents handling. Later you can use this documentation as evidence to bring the attacker to justice.
- Train your IRT and organize drills.

See also:
- Computer Security Incident Handling Guide, NIST 800-61
- The Incident Handlers Handbook, SANS Institute
• SANS incident forms
Mirantis OpenStack security

In this chapter, we will point out the security features enabled by default in Mirantis OpenStack and provide recommendations on proper configuration of hosts and main services.

Encryption strategies

Recommendations:

- Follow NIST recommendations for choosing appropriate cipher suites and key management techniques.
- For storing passwords, always use salt. Salt should be unique for every stored password and randomly generated.
- For symmetric encryption with passphrases, use a passphrase with appropriate entropy valid for particular cipher key strength and expected brute-force durability. For example, a valid passphrase for 128 bit cipher (CAST-5, AES-128) should contain at least 128 bits of entropy.
- Whenever possible, use ephemeral keys to maintain forward secrecy. Use Diffie-Hellman for exchanging keys.
- Whenever possible, use Elliptic Curve Cryptography (ECC) as it requires less computational power than RSA or DSA.

To protect sensitive data (encrypting and digitally signing) in a long perspective (2031 year and beyond), use cipher suites and key length with security strength 128 or more (192, 256).

Note:

The finite-field cryptography (FFC) and integer-factorization cryptography (IFC) algorithms with higher security strength of 192 and 256 bits are not currently included in the NIST standards for interoperability and efficiency reasons.

Use the algorithms that have security strength of 128 bits that are secure and efficient at the same time. To protect data until 2030, you can use cipher suites and key length with the security strength of 112 bits.

<table>
<thead>
<tr>
<th></th>
<th>Until 2030 (key strength = 112 bits)</th>
<th>After 2030 (key strength = 128 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hashing and digital</td>
<td>SHA-224, SHA-512/224, SHA3-224</td>
<td>SHA-256, SHA-512/256, SHA3-256</td>
</tr>
<tr>
<td>signatures</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symmetric</td>
<td>3TDEA</td>
<td>AES-128</td>
</tr>
</tbody>
</table>
### Key and certificate management

#### Recommendations:
- Use different key pairs to sign and encrypt messages to mitigate information disclosure and tampering attacks.
- Do not use the public same key in different certificates due to possible substitution (spoofing) attacks.
- Use secure protocols for dissemination of certificate and revocation information such as LDAP repositories.
- Update keys and corresponding certificates every three quarters.
- Provide reliable storage for expired keys that can be used later to retrieve and recover encrypted data.
- Consider using the OpenStack Anchor - an ephemeral PKI certification system that uses automated issuing rules and short life certificates to mitigate common certificate security issues.

#### See also:
- [SANS Key and certificate management in PKI technology](#)
- [OpenStack Anchor Project](#)

### Host security

To improve host security, create the IT host security policy described in Appendix A.

#### Mandatory access control security enhancement

Use Linux mandatory access control (MAC) enhancements to mitigate EoP threat. SELinux, AppArmor, and grsecurity are possible MAC security enhancements for Linux. The choice of a Linux security enhancement may depend on your personal experience and type of a host OS distribution.
Rootwrap

Rootwrap is a security wrapper designed to allow a service-specific unprivileged user to run a number of actions as the root user in the safest manner possible mitigating EoP such as when an attacker takes advantage of a running service with the ‘root’ privileges. The ‘rootwrap.conf’ file contains filter definition directories and specifies command filters to be loaded for them. Since the configuration file is in the trusted security path, it needs to be owned and writeable only by the root user to avoid tampering.

On a host Linux machine you have an option to enable encryption for a home directory when creating a privileged user to mitigate information disclosure threat. For example, on Ubuntu you can use the following command:

```
# adduser --encrypt-home
```

You might want to encrypt not the whole Home directory but only a specific folder of files. In such case you can use ‘~/.Private’ folder to store keys and configuration files. The data stored in this folder will be decrypted when the folder is automatically mounted on logon.

See also:
- Payment Services Compliance Standard: PCI DSS
- US Government agencies security standard: FedRAMP/FISMA
- Rootwrap Documentation
- Home directory encryption on Ubuntu

Compute and hypervisor security

In this context we see two types of threats:
- hypervisor threats
- multi-project threats

Malicious application can escape the VM through exploitation of a vulnerability or a direct access to hardware or hypervisor OS and compromise other VMs running on a physical node which may belong to another projects (EoP).

The hypervisor security should be a prime concern because a single fault on the hypervisor level may compromise the whole environment.
For example, having access to the hypervisor an attacker can look into VMs’ images by simply mounting a virtual disk (information disclosure). Even more, mounting a filesystem as ‘read only’ can help to avoid tampering. An attacker becomes untraceable because peeking files this way does not update files’ access time (repudiation).

**Recommendations for hypervisor to mitigate EoP:**

- Use the hypervisor certified against [FIPS 140-2](http://www.nist.gov) and take [Common Criteria Certification](http://www.commoncriteriaportfolio.org) requirements into consideration.
- Limit access to hypervisor OS (a full OS or kernel the hypervisor runs on), the VM manager, and all interfaces used to manage VMs.
- Do not use hypervisor memory optimization such as Copy-on-Write (COW) mechanisms shown to be vulnerable to side-channel attacks when used in multi-project environment.
- Disable PCI passthrough for your hypervisor, which means that an VM instance should not have a direct access to hardware such as memory (DMA) or video cards (GPUs).
- Harden virtual hardware (QEMU for KVM) by:
  - minimizing the code base by removing unused components from a QEMU configuration
  - building QEMU with compiler hardening enabled, which may include: stack protection, data execution prevention, Address Space Layout Randomization (ASLR) by enabling Position Independent Executable (PIE)
  - using mandatory access controls such as sVirt, SELinux, AppArmor, or grsecurity to put QEMU process into a separate security context
- A hypervisor must host only VMs of the same security level that can be classified based on their role, function, or access to sensitive data. Use Host Aggregates and Availability Zones to group Compute nodes for running VMs of the same security level.

**See also:**

**Recommendations for Compute service:**

- limit an access by providing strict access permissions to:
  - ‘nova.conf’ file
  - ‘/var/lib/nova’ folder

Additionally, you can use:

- file integrity monitoring (FIM) tools such as iNotify or Samhain to trace unexpected files modifications
- rootwrap to execute Compute commands as the root user
- trusted compute pools
Implement or enable:

- centralized logging to mitigate a risk of a repudiation attack
- policy.json
- mandatory access control (MAC) with SELinux or other operating systems
- encryption for Compute metadata traffic

See also:

- Secure with rootwrap
- Compute security hardening
- Security advisories by:
  - VMWare: [http://blogs.vmware.com/security/](http://blogs.vmware.com/security/)
  - Others (KVM, and more): [http://seclists.org/oss-sec](http://seclists.org/oss-sec)

**Virtual consoles**

In Compute service you can use VNC or SPICE protocols. The OpenStack Dashboard service supports both protocols. When using VNC, enable TLS for desktop traffic encryption.

See also:

- Securing remote VNC console with TLS
- Securing SPICE console with TLS

**Compute hardware platform security**

Consider using Intel Trusted Execution Technology (TXT) to build a chain of trust from server’s firmware to a hypervisor to prevent EoP and tampering attacks to BIOS, MBR, and boot loader that can be implemented by bootkits or ransomware but mostly for the Windows platform. A bootkit is an advanced malware capable to inject itself at a booting stage before OS starts to avoid being detected by a host security solutions such as HIPS (antiviruses). For example, the recent MBR bootkit called HDRoot discovered in 2015 managed to poison MBR to launch later the backdoor as a system service when Windows starts. Another threat is cryptolockers. For example, Petya and Mamba cryptolockers can encrypt Master File Table and disk partitions correspondingly.

To mitigate EoP attacks and tampering attacks, use Trusted Filter for Filter Scheduler in OpenStack that implements Intel TXT to schedule workloads requiring trusted execution only to
trusted compute resources. Clusters can have both trusted and untrusted compute resources. Trusted compute resources are grouped into the Trusted Computing Pool.

Workloads not requiring trusted execution can be scheduled on any node, depending on utilization, while workloads with a trusted execution requirement will be scheduled only to trusted nodes.

See also:
- Mirantis Blog: Trusted Cloud computing with Intel TXT: The challenge
- Intel® Trusted Execution Technology (Intel® TXT) Enabling Guide

Images

Recommendations:
- Consider using image signing to mitigate tampering attack
- Use trusted and verified VM images. Images from non-trusted sources may contain security breaches or unsolicited malicious code (spoofing, information disclosure).
- Scan a VM image that you are going to use with a vulnerability scanner like Nessus and an antivirus scanner.

Containers

Recommendations:
- Use a minimal Linux OS distribution to reduce the attack surface. (EoP)
- Enable security modules SELinux, AppArmor, or grsecurity to provide a container with least privileges. (EoP)
- Use cgroups to set CPU shares and memory to mitigate resource exhaustion by neighbors. (DoS)
- Reduce kernel capabilities with ‘--cap-drop’. (EoP)
- Do not use pre-installed and unverified containers because they may have security breaches or unsolicited malicious code. (Tampering)

Networking

Network equipment

Recommendations:
- Change initial vendor’s default passwords for all networking equipment to mitigate EoP.
- Use a network management utility provided by a vendor. The management utility helps to ensure that network operations are less error-prone and have adequate change
management record history. Note, that a network equipment must be homogenic, provided by a single vendor.

- Document network equipment changes and use change management process such as ITIL / ISO 2000 service management techniques.

Network services

Networking service

Recommendations:

- Use an isolated management network to provide communication between the OpenStack Networking services and other OpenStack core services to mitigate spoofing and tampering attacks.

- Enable security groups to specify the type of traffic and a direction (ingress/egress) that is allowed to pass through a virtual interface port. Disable security groups in Compute service and proxy all security group calls to Networking API. To do that, set 'firewall_driver' to 'nova.virt.firewall.NoopFirewallDriver' to prevent 'nova-compute' from performing iptables-based filtering; 'security_group_api' - to 'neutron' to have all security group requests proxied to Networking service.

- Secure Networking API endpoint through TLS 1.2 or later. Use TLS 1.2 or later with available stack of ciphers. For example, you can use DHE-RSA-AES256-GCM-SHA384 cipher with DH public key size 3072 bit and private key size 256. In case of ECC, use TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 cipher with a 256 bits DH key length using elliptic curves. SP800-131A approves AES-128, 192, 256 bits encryption to mitigate information disclosure threat. See the ‘Encryption strategies’ section above.

- Keep private keys secure on API endpoints by using appropriate file permissions and other controls to mitigate information disclosure threat.

- Define a network policy enforcement (RBAC) to Networking-related actions, depending on customer's requirements, policy, and use case to mitigate EoP threat. Customize the Networking ‘policy.json’ file.

- Networking service separates projects by utilizing iptables along with ‘ebtables’ rules. These rules prevents MAC and ARP spoofing attacks on “virtual” or NFV L2 layer.

- Configure per-tenant quotas for L2 and L3 resources and security groups for projects to avoid overconsumption of network resources and mitigate DoS attacks. See OpenStack Admin Guide for basic quotas configuration.
OpenContrail
[to do]

See also:
- OpenSSL stack of ciphers
- Manage Networking service quotas, OpenStack Admin Guide
- OpenStack Security Guide: Networking
- SP800-131A
- FIPS 186-4

CLI and API

OpenStack CLI python clients require a username and password supplied to perform a request. The OpenStack CLI client can authenticate a user in several ways by using:

- the `OS_USERNAME` and `OS_PASSWORD` environment variables that may result in information disclosure and EoP.
- the OpenStack RC file, which you can download from the Dashboard with environment variables already set for a user. However, storing credentials into unencrypted files on a disk is prohibited and may result in information disclosure and EoP.
- OpenStackClient that supports authentication:
  - by typing password for each request
  - with the provisioned authentication token

To avoid risk of revealing passwords, use OpenStackClient.

To use old OpenStack CLI Python clients, perform EoP mitigation steps described below:

- For accessing OpenStack API through OpenStack python CLI clients, dedicate additional node or virtual machine, place it into separate internal DMZ and use it solely only for this purpose (a jump host).
- On this node disable all unnecessary services and disable SFTP service, or make SSH/SFTP only accessible from dedicated, trusted network segment.
- On this node, consider using grsecurity patched kernel.
- Implement bash or other shell script that will wrap standard OpenStack python CLI clients and will require password to be entered for each run. Supplied password will set environment variable and unset it after every run.
- Implement an internal policy for users who interact with CLI to never supply password as command arguments to prevent sniffing through process listing (even when using grsecurity kernel).
• Disable shell (bash) history for all users.

See also:
  • OpenStack Configuration Reference
  • OpenStackClient Configuration

**Messaging**

OpenStack components uses the OSLO messaging security library to communicate with “worker” processes running on compute nodes and a cloud controller node. For best possible performance and scalability OSLO library does not employ signing or encryption. As a result, messaging security depends on message broker’s security.

You need to protect a messaging broker. Mirantis OpenStack uses the RabbitMQ messaging broker.

Mirantis recommendations for messaging security and RabbitMQ:
  • Delete the RabbitMQ guest user.
  • Separate API functional publishers (Nova, Cinder, Neutron, and others) by leveraging “rabbit_virtual_host” configuration setting for each API and creating appropriate Rabbit virtual host (rabbitmqctl add_vhost)
  • For each RabbitMQ virtual host create unique credentials along with appropriate permissions (rabbitmqctl add_user, rabbitmqctl set_permissions)
  • Monitor RabbitMQ network activity with iptables or other monitoring tool to get accounting information.
  • Forward the RabbitMQ and HAProxy logs to the central syslog server.
  • Use TLS for messaging transport security.

See also:
  • OpenStack Security Guide: Messaging

**Identity**

Use Identity API v3 that obtained the following features:
  • Identity Federation
  • External authentication
  • Multi-factor authentication
  • Authorization mechanisms
  • Multiple domains support
Authentication

For authentication we do not recommend using simple login and password credentials, as Identity does not enforce policies on password strength, expiration, or failed authentication attempts as recommended by NIST Special Publication 800-118. Consider using Identity extensions or external authentication. Also refer to the Require/enforce strong admin/users passwords in built-in Identity Service blueprint.

The easy way to integrate Identity authentication with an existing directory service is using LDAP. LDAP users can be mapped into roles and groups within Identity in `/etc/keystone/keystone.conf` for use by the various OpenStack services.

External authentication

Use an SQL identity backend together with X.509 authentication or Kerberos for Keystone under Apache instead of using the username and password pair. Attributes coming with X.509 certificate could be matched against OpenStack identity data structures such as projects, domains, and groups.

For more information, please visit OpenStack’s external authentication mechanism web page: http://docs.openstack.org/developer/keystone/external-auth.html.

Multi-factor authentication

Multi-factor authentication reduces the risk of passwords being compromised. We recommend using at least two-factor authentication (TFA) for privileged accounts such as admin to comply with NIST 800-53 IA-2(1) guidance.

Implement multi-factor authentication by leveraging the external authentication mechanism. Similar to the Federation scenario Keystone process is executed on Apache HTTPD. Once authenticated with multi-factor authentication mechanisms, Apache web server will pass down an authenticated user to Keystone using the REMOTE_USER environment variable.

Also we recommend that you enable TLS for client authentication to provide an additional factor of authentication. This requires certificates to be issued for OpenStack services, which can be self-signed and issued by internal authority. However, in this case you need to disable the validity check or mark a certificate as trusted. Read configuration details in http://docs.openstack.org/security-guide/identity/authorization.html.

Tokens

- By default a token expiriess in one hour. The recommended expiry value should be set to a lower value that allows enough time for internal services to complete tasks.
• Fernet tokens are the most preferable to use. Fernet provides a secure messaging protocol specially designed for REST API communication being non-persistent and lightweight to reduce operational overhead. It uses AES-CBC to encrypt data and SHA HMAC to sign.

**Domains**

The Identity V3 API introduces a multi-tenancy model via using multiple domains where users can be represented with different authentication back ends and even have different attributes. Users of different domains can be mapped to a single set of roles and privileges, that are used in the policy definitions to access the various service resources.

You can enable domain-specific authentication drivers for multiple domains in the '[identity]' section of the keystone.conf file. Read more details in [OpenStack Security Guide: Domains](#).

A domain owner can create additional users, groups, and roles to be used within the domain.

**Groups**

A group is a container representing a collection of users. Rather than assign a role directly to a user/project, a domain owner can assign a role to a group, and then add users to that group.

**Notes:**

- Generally, groups and domains are optional. However, when using Federation (e.g. SAML), roles or policies are mapped to groups.
- The domain name and role name is globally unique across all domains.
- The username, project, and group name are only unique to the owning domain.

**Identity Federation**

Identity Federation brings an ability to have several clouds served by the same Identity provider. Requirements:

- Identity API v3 OS-FEDERATION Extension
- Apache 2.2.22 or later
- Ubuntu 12.04 or later

**Note:**

Currently, Mirantis does not support Federation as a stable feature with Identity API v3. However, our support engineers can help you to enable it via Identity API v2.
You can configure your Identity service to be used as a Service Provider or an Identity Provider.

There are three major protocols for Identity Federation: SAML, OpenID, and OAuth. Two of them are supported under Apache now:

- SAML 2.0 implementations:
  - Shibboleth - see [Setup Shibboleth](#)
  - Mellon - see [Setup Mellon](#)
- OpenID Connect - see [Setup OpenID Connect](#)

OpenStack Security Guide explains the way of configuring Federation using the Shibboleth protocol on Ubuntu with the Apache HTTPD server.

Authentication middleware

Mirantis does not recommend using a custom WSGI authentication middleware as it may bring additional security risks due to improper implementation.

Mirantis recommends removing “admin_token” middleware. This WSGI middleware effectively bypasses identification + authentication. There is no traceability or accountability in its use. It is exclusively intended for bootstrapping Identity service before any user accounts exists and is useful for a SQL-based identity deployment, but not necessarily against a read-only LDAP deployment. To mitigate the risk admin_token middleware, disable it and move to domain-based approach for security management, like described below:

- Create a new domain for cloud management purposes: “cloud_admin_domain”
- Assign “admin” role to an appropriate user, for example: “John Mc. Admin”
- Update Identity policy.json file to match newly created domain.
  
  Replace:

  
  "cloud_admin": ["rule:admin_required", "domain_id:admin_domain_id"]

  with

  "cloud_admin":

  ["rule:admin_required","domain_id:<cloud_admin_domain_id>"]

- Remove admin_token from /etc/keystone/keystone.conf
- Remove admin token auth middleware from /etc/keystone/keystone-paste.ini:
  
  [filter:admin_token_auth] paste.filter_factory =
  keystone.middleware:AdminTokenAuthMiddleware.factory

See also:

- [Configuring Keystone for Federation](#)
- [Using Identity API v3](#)
PKI and cryptography service (Barbican)

Barbican is a ReST API designed for secure storage, provisioning, and management of secrets. It is useful for all types of environments, including large, ephemeral Clouds.

We recommend using Barbican to protect your data and images by storage encryption.

See also:

Storage

Block storage

- Set strict access permissions (at least 640) for the following configuration files in `/etc/cinder/`: `cinder.conf`, `api-paste.ini`, `policy.json`, `rootwrap.conf`.
- Do not set `noauth` value to parameter `auth_strategy` under `[DEFAULT]` section.
- Enable TLS for authentication.
- Enable secure file permissions for Network-attached storage (NAS) by setting parameter `nas_secure_file_permissions` under `[DEFAULT]` section in `/etc/cinder/cinder.conf` to `auto`.
- Ensure `osapi_max_request_body_size` under `[DEFAULT]` section in `/etc/cinder/cinder.conf` or `max_request_body_size under` under `[oslo_middleware]` section in `/etc/cinder/cinder.conf` is set to 114688. This helps to avoid a DoS attack when an attacker sends an oversized request.

Object storage

- Use a private (V)LAN network segment for your storage nodes in the data domain.
- Configure each Object Storage service to run under a non-root service account, for example use a username `swift` with the primary group `swift`.
- Object storage architecture implies using whether an individual proxy node or multiple proxy nodes with a possibility to use a load balancer. Every proxy node should have at least two interfaces: public and private. Set up a firewall to protect the public interface on a proxy node. The public facing service on a the proxy node is an HTTP web server that handles endpoint client requests, authenticates them, and performs the appropriate action. The private interface establishes outgoing connections to storage nodes on the private storage network.
Ceph

- Use ‘cephx’ to authenticate users and daemons in order to protect against MitM attacks (Information Disclosure, Tampering). ‘cephx’ uses shared secret keys for authentication. Note, that network communication channel is not encrypted including the messages used to configure said keys. The system is primarily intended to be used in trusted environments.
- For block storage encryption Ceph-disk can utilize Linux dm-crypt functionality via ‘--dmcrypt’ parameter to mitigate Information Disclosure threat. Note, that keys are stored in ‘/etc/ceph/keys’ by default, which requires setting strict permissions for this folder.
- Ceph provides authentication and protection against MitM attacks once secret keys are in place. However, data over the wire is not encrypted, which may include the messages used to configure said keys (Information Disclosure). The system is primarily intended to be used in trusted environments.
- Use Ceph in a multi-project mode to mitigate EoP.

See also:
- [Block Storage Checklist](#)
- [OpenStack Block Storage documentation](#)
- [OpenStack Storage documentation](#)
- [Ceph documentation](#)

OpenStack Dashboard

Consider the OpenStack Dashboard service security in context of Horizon’s building blocks:
- Linux node security - security of Linux node hosting Dashboard web app
- Django security
- Application security (Horizon)
- Apache httpd web application container and its AppArmor/SELinux profiles
- Apache httpd and mod_wsgi configuration
- Apache TLS and cipher suite configuration

Recommendations:
- Do not deploy OpenStack Dashboard on a shared subdomain with user-generated content (EoP).
• Disable local image uploads through Horizon by setting
  ‘HORIZON.Images.AllowUpload’ to ‘False’ in your ‘local_settings.py’ file to
  protect against a DoS attack.

• Configure the ‘ALLOWED_HOSTS’ setting with the fully qualified host name(s) that are
  served by the OpenStack Dashboard (EoP).

• Deploy the OpenStack Dashboard service behind the HTTPS web server with TLS v1.2.

  **Note:** A user should set up a local DNS resolver to resolve hostnames (FQDN) of
  TLS-wrapped endpoints to corresponding IP addresses of these endpoints to mitigate
  Spoofing threat.

• For HTTPS set session cookie to use ‘HTTPONLY’. Update the following options in the
  ‘/etc/openstack-dashboard/local_settings.py’ file to secure the session and ‘CSRF’
  cookie:

  ```
  SESSION_COOKIE_HTTPONLY = True
  CSRF_COOKIE_SECURE = True
  SESSION_COOKIE_SECURE = True
  ```

• Configure your web server to send a restrictive Cross Origin Resource Sharing (CORS)
  header with each response, allowing only the dashboard domain and protocol:

  ```
  Access-Control-Allow-Origin: https://example.com/
  ```

  Do not allow the wild card origin to mitigate DoS threat.

• Deploy the OpenStack Dashboard service to a dedicated virtual machine or container, in
  a demilitarized zone (DMZ) separated from other services.

• Protect a Linux host and Apache web server following security best practices.

• For storing as session state use dedicated Memcache servers, not shared with other
  OpenStack services (EoP threat).

• Disable HTTP methods you do not need.

• Consider using TFA for a Web access to mitigate EoP.

• Follow “OWASP Top-10” security guidelines for web application security,

• Place the OpenStack Dashboard service beyond Web Application Firewall (WAF) to
  mitigate EoP and DoS threats.

• Use IDPS along with real time threat monitoring software to mitigate EoP.

• Prior to deploying the OpenStack Dashboard service into production, perform security
  assessment:

  o Scan all publicly exposed IPs with a vulnerability assessment tool.

  o Run a penetration test according to OWASP Top 10 guideline.
Fuel Master node

The Fuel Master node plays a critical role in clouds deployed by it. It is vital to understand, that after cloud deployment, a cloud considered as a production environment could be also redeployed from the Fuel Master node. Fuel Master node stores also SSH private key, which allows passwordless access through SSH to all cloud nodes (machines) deployed by Fuel. Therefore, once compromised, an attacker can get access to the whole environment.

Taking above into account, a cloud owner should consider the Fuel Master node as critically important asset and protect it from potential compromise by all possible technical and organizational means.

Mirantis recommendations in regard to Fuel Master node security:

- Protect a network access to the Fuel Master node on a network layer by requiring two factor authentication (TFA) while accessing a network segment with the Fuel Master node. You can use VPN with TFA enabled using a shared secret key as a second authentication factor.
- Apply node hardening guidelines to the Fuel Master node:
  - logs audit forwarded to the central log collector
  - Fuel uses GRUB as a bootloader, so please follow the recommendations in the GRUB Manual
- Change root and Fuel web UI initial credentials (‘root:r00tme’ and ‘admin:admin’ correspondingly), follow secure credentials guidelines.
- Create additional accounts on the Fuel Master node to regularly connect via SSH and use ‘sudo’ to elevate privileges if necessary.
- Add a passphrase to the RSA private key of “root” user, store a passphrase string in a secure place.
- Encrypt ‘astute.yaml’ file by PGP, store the PGP (GnuPG) passphrase string in a secure place.
- If using backups, allow encrypted backups of the Fuel Master node only. Pay attention to backup media management process. If you do not want to have encrypted backups, use file integrity monitoring (FIM) tools such as iNotify or Samhain to trace unexpected files modifications instead.
Monitoring and incidents detection

Since the version 6.1 Mirantis OpenStack has an option to deploy certified LMA (Logging, Monitoring, Alerting), now officially called StackLight, plugin with your environment.

Recommendations:
- Forward logs from all cloud nodes to central log collector.
- Protect syslog protocol by using TLS.
- Separate duties for management of logging subsystem and cloud management.
- For security operations monitoring use Ceilometer along with security intelligence solution to detect potentially unwanted or harmful activities, for example IBM QRadar SIEM.
- Establish an incident response plan and an incident response team for this purpose.
- Perform threat modeling, trainings, and simulations of incident response at least annually.

See also:
- Monitoring Guide

Security features enabled in Mirantis OpenStack

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| All Linux nodes conform to baseline hardening, including hardened SSH daemon configuration, hardened firewall rules, hardened TLS cipher suites with TLS 1.2 support, hardened HTTP/REST interfaces passing all OWASP tests | Scope of basic hardening:  
  - iptables rules  
  - SSH configuration and encryption protocols  
  - Apache HTTP TLS 1.2 cipher suites  
  - TCP/IP stack and network settings  
  - Linux kernel vfs and file system layer |
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELinux / AppArmor for improved security on ‘compute’ nodes</td>
<td>AppArmor and SELinux provides improved security for compute virtual machines by confining workloads and ensuring that different workloads doesn’t interfere (sVirt Linux capability)</td>
</tr>
<tr>
<td>Intel TXT support (OpenAttestation) for improved hardware and platform</td>
<td>Mirantis OpenStack supports Intel’s Trusted Execution Technology to ensure that entire hardware and software stack is trustworthy and not compromised.</td>
</tr>
<tr>
<td>security**</td>
<td></td>
</tr>
<tr>
<td>MOS compatibility with FIPS 140-2 certified hardware (some HP ProLiant</td>
<td>Mirantis OpenStack could be deployed on a FIPS 140-2 certified hardware. HP’s Enterprise Secure Key Manager (ESKM) along with HP servers enables Mirantis OpenStack customers to protect digital assets and ensure continuous access to business-critical, sensitive, data-at-rest encryption keys, both locally and remotely.</td>
</tr>
<tr>
<td>platforms offer it)**</td>
<td></td>
</tr>
<tr>
<td>Seamless LDAP/AD integration for secure authentication purposes</td>
<td>Mirantis OpenStack can leverage OpenLDAP and Microsoft Active Directory for appropriate account security including password policies and account security policies. This functionality is enabled by using Fuel’s LDAP plugin.</td>
</tr>
<tr>
<td>Customized RBAC policies for granular access control**</td>
<td>Mirantis OpenStack enables customers to develop customized RBAC policies, meeting sophisticated RBAC requirements for appropriate separation of duty (SOD) and granular access control to mitigate EoP attacks.</td>
</tr>
<tr>
<td>HAProxy for DoS/DDoS attack protection for Web and REST API access**</td>
<td>Mirantis OpenStack by default hides all sensitive API and HTTP web UI services behind reverse proxy making mitigation of DoS/DDoS attacks easy to implement and monitor.</td>
</tr>
<tr>
<td>TLS support with AES128/256 cipher suites and Diffie-Hellman Ephemeral</td>
<td>Diffie-Hellman Ephemeral Cipher Suites support provides forward secrecy, making Mirantis OpenStack resistant to eavesdropping and sniffing attacks.</td>
</tr>
<tr>
<td>Key Exchange (ECDHE)</td>
<td></td>
</tr>
<tr>
<td>LMA toolchain for improved security analytics and early anomaly detection</td>
<td>LMA (Logging Monitoring Alerting) toolchain provides advanced analytics for Mirantis OpenStack, enabling control plane authentication and operations early anomaly detection.</td>
</tr>
<tr>
<td>Broad range of network security technologies for workloads protection:</td>
<td>Mirantis OpenStack is certified to work with Juniper Contrail and Palo Alto Networks Next Gen Firewalls, providing outstanding security experience for all types of cloud workloads.</td>
</tr>
<tr>
<td>VPNaaS, FWaaS, advanced SDN network stacks</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>integration like Juniper Contrail for advanced network service chaining and security services</td>
<td></td>
</tr>
<tr>
<td>CADF Ceilometer integration for out of the box auditing capabilities**</td>
<td>CADF framework incorporated into Mirantis OpenStack makes it easy to monitor all operations related to authentication, authorization, privileged users operations, sensitive assets changes to mitigate repudiation and tampering threats.</td>
</tr>
<tr>
<td>Enhanced auditing and security intelligence capabilities through integration with 3rd party product offering (IBM QRadar)**</td>
<td>Security intelligence solutions offers improved visibility and advanced analytics of cloud events. All events related to cloud operations, privileged user operations, sensitive asset operations are monitored and analyzed in semi real-time. In case suspicious activity occurs, alarm events are triggered.</td>
</tr>
<tr>
<td>TLS Mutual Authentication for improved API and web access security**</td>
<td>Mutual TLS Authentication along with TLS 1.2 support provides additional layer of security by mitigation of information disclosure, for example, Man-In-The-Middle (MITM) attacks.</td>
</tr>
<tr>
<td>OTP token support (Multi Factor Authentication)</td>
<td>Multi Factor Authentication and integration with broad range of OTP tokens, including RSA SecurID, SafeNet, Yubikey provides substantially improved authentication security when compared to password only authentication.</td>
</tr>
<tr>
<td>Federation: SAML 2.0 and OpenID Connect (OIDC) support</td>
<td>Mirantis OpenStack is verified to work with corporate federation solutions including Microsoft’s Active Directory Federation Services, Shibboleth, IBM’s Web Sphere Security, Gluu, Ping Federate</td>
</tr>
<tr>
<td>Secure network architecture allowing seamless DMZ integration**</td>
<td>Fuel advanced network templating mechanisms provides support for DMZ network topologies for API and Web UI security.</td>
</tr>
<tr>
<td>PCI DSS, FISMA/FedRamp compliance**</td>
<td>For PCI DSS and FISMA/FedRamp compliance we collaborate with QSA/3PAO security services providers to achieve required level of conformance</td>
</tr>
<tr>
<td>SSH brute force protection</td>
<td>Grants access from all networks (except the provided ones), but Fuel checks the networks against the brute force attack.</td>
</tr>
</tbody>
</table>

** - advanced features requiring deployment engineer engagement. These features are usually not available when using Fuel web UI.
Mirantis OpenStack updates and configuration management

Mirantis OpenStack starting from version 5.1 supports maintenance updates of Mirantis OpenStack packages, which include security updates as well. Since the version 7.0, Mirantis OpenStack has an update automation capability.

Mirantis delivers a rolling patch management solution with Mirantis hardened OpenStack packages. Mirantis offers a update and security repository with Mirantis hardened OpenStack packages. These repositories should be used to patch cloud environment. The Mirantis repositories are consistent with the Ubuntu/Centos approach for Linux patching. A customer can:

- apply one or more patches according to the list of available updates
- query and apply updates to all nodes

Mirantis OpenStack Security Updates

Every vulnerability reported to the OpenStack community or directly to us is thoroughly analyzed using our threat modelling and research process. Mirantis follows CVSS and DREAD threat risk models and prioritizes vulnerabilities based on their CVSS score. Fixes for issues with a CVSS base score of **High** are developed, tested, released, and described in technical bulletins as soon as possible. Fixes for issues with a base score of **Low and Medium** are grouped together and released in Mirantis OpenStack Maintenance Updates.

For more information on Mirantis OpenStack Maintenance Updates, please visit the following web sites:

- Mirantis OpenStack 5.1 Maintenance Updates
- Mirantis OpenStack 6.0 Maintenance Updates
- Mirantis OpenStack 6.1 Maintenance Updates
- Mirantis OpenStack 7.0 Maintenance Updates
- Mirantis OpenStack Technical Bulletins

Configuration Management

- Use roles assigned to nodes instead of individual recipes.
- Use git tag to check-out the needed cookbook version
- Store Data bags in git in encrypted form
- Avoid creating of large data bags, split/segregate data bags for various sub-components
- Use staging/testing environment to test configuration changes
● Use appropriate versioning scheme for cookbooks, for example one version upgrade per promotion from staging to stable environment.

● Implement policy to track dependencies between development components. e.g. make sure that when you promote cookbook or package you have updated the dependent part in other components, like data bag or node/environment variables. A good stat can be release notes for deployment with the list of related and depended changes.

● Implement “branching” strategy to avoid linear cookbook version increase and get an ability to promote host fixes to individual environment.

● Perform a vulnerability scan and security assessment after the change is committed to the production environment.

● Identify changes in environment behavior after the change is committed to the production environment. Track carefully logs to observe undesirable side-effects.

See also:

● Mirantis OpenStack Maintenance Updates
Cloud security architecture

You can secure your cloud environment by introducing specific security zones and domains where you can aggregate such tools as intrusion detection and prevention system, firewall, antivirus, and sandbox for deep data analysis and incident response. However, it should be noted, that it will consume extra computational resources for which a cloud owner has to pay. When you consider a deployment of a new security solution or creation of security domain or zone, you should follow the rule, Do not invest more resources than the cost of your information assets you want to protect. Cloud providers should consider a Security-as-a-Service model. According to that model tenants can buy extra security services for some of their projects if necessary. Most security threats for cloud users come mostly from guest VMs and external/public networks used to transfer North-South Internet traffic. However, monitor the East-West traffic as well to detect proliferation and presence of APT. With APT attackers can perform internal reconnaissance, for instance, scan tenant’s network for vulnerable systems, and propagate further to take over tenant’s and provider’s resources needed to complete the mission.

Recommendations:
- provide users with guidance on securing their workloads up to installing malware defense software in guest OS to mitigate EoP
- create demilitarized zone (DMZ) when deploying your environment to mitigate EoP
- deploy a Dashboard service on a separate node behind DMZ to mitigate DoS and EoP
- create a Security Domain for deep traffic inspection and incident investigation to mitigate EoP
- use network IDPS for both N-S and E-W traffic inspection to reveal network indicators of compromise

Demilitarized zone

Demilitarized zone (DMZ) represents a neutral zone and aims to isolate private network from the Internet by including redundancy in cloud infrastructure. The main idea is moving frontend Web server, which needs to be externally accessible and may become a target of an attack, to DMZ which prevents attackers penetrating an internal network in case of a successful attack. There are different types of DMZ configuration that suit various cloud architectures depending on the security level and performance you want to have. NIST 800-44 recommends establishing DMZ.

Consider the example of a “service leg” DMZ architecture with at least three network interfaces to communicate with an external (for example, an Internet border router), internal (API services endpoints and Dashboard), and DMZ networks. Even though, two-firewall DMZ is considered to be more secure, because in case of “service leg” DMZ once the host that runs firewall is compromised an attacker can get an access to an internal network interface, we recommend “service leg” DMZ with a single firewall as requiring less resources and easy to manage.
The only problem that “service leg” DMZ can not solve is protecting against DoS attacks aimed to Web server that may affect an internal cloud services operation. To fix this problem DDoS protection tool is introduced on the scheme. However, it may happen that DDoS will consume all incoming bandwidth and overload a border router.

As a rule, Firewall is configured to permit external web traffic. This brings a risk of malformed HTTP packets entering the environment. That is why, IPS system accompanies Firewall on the diagram. IPS can parse the most popular L7 protocols to detect and drop malicious streams minimizing the risk of compromise.

Place the OpenStack Dashboard service and OpenStack APIs endpoints behind Firewall and access the OpenStack services through reverse proxy hosts in DMZ.
WAF presence is mandatory. Consult with WAF vendor regarding options to protect public APIs.

See also:
- NIST 800-44 recommendations: Guidelines on Securing Public Web Servers

**Security domain**

Security Domain is a separate project with a set of tools needed for IRT to perform incident analysis and cloud forensics. Once a security issue is discovered in one of the projects’ networks, you can switch traffic from the compromised network or the whole project from the Internet to Security Domain to contain and investigate an incident.

Security Domain may contain the following components but not limited to:
- Network IDPS - for deep packet inspection and filtering out malicious files from the North-South traffic traveling through DMZ. You can create numerous virtual instances of network IDPS as a VNF to scan East-West traffic in your SDN as well.
- Network monitor such as tcpdump or Wireshark - to record the traffic for investigation and forensics purposes.
- Antivirus scanner - for fast scan of the files extracted from the network N-S traffic.
- Sandbox - to analyze discovered malware or documents with exploits in a form of PDF, SWF, DOC found, for example, in the mail traffic, in a matter of minutes.
- Forensic tools - to collect digital evidence for the court.
- Storage specially allocated for forensic purposes - to store collected digital evidence such as infected VM images, stored illegal data, dumps of network traffic, logs, and so on.
- Proxy server to redirect traffic of the compromised network to the Internet through the Security Domain network, where IRT can perform deep packet inspection and hidden monitoring of the ongoing attack.
Cloud security solutions

When choosing cloud security solutions, consider the following tradeoffs:

- Open source vs. Proprietary
- Single tool vs. Multi tool
- Hardware vs. Virtual appliance

For example, virtual appliances fit more scalable environment with NFV enabled. Open source solutions help to build vendor-unlocked architecture but are limited in the number of features and have no support service in most cases. Proprietary solutions are usually a multi tool comprising several functions in one software. For example, Next Generation Firewall (NGFW) include a traditional L3-4 firewall, intrusion prevention system (IPS), L7 firewall with in-line deep packet inspection (DPI), an intrusion prevention system (IPS), NAT, and VPN. And Unified Threat Management (UTM) solution can also include anti-spam, antivirus, and DLP applications.

Being structured by the composition criterion the list of the recommended solutions is as follows:

- Single tool
  - Firewall (L3-4)
  - WAF
  - Load balancer
  - DDoS protection
  - Network and Host Intrusion Detection and Prevention System (IDPS)
  - Cloud Antivirus
  - Sandbox

- Multi tool
  - Next-Generation Firewall (NGFW)
  - Next-Generation IPS (NGIPS)
  - Unified Threat Management (UTM)

WAF and load balancers

Web Application Firewall (WAF) is used to filter malformed HTTP requests, for example, bringing XSS attacks, SQL injection, and remote file inclusion.

Load balancers are not security solutions, but they may help to preserve the availability of services mitigating an impact of DoS attacks.

You can find solutions that combine both WAF and load balancing functionality.

See also:

- Recommended security solutions section
Reverse proxies

To minimize risks of Web server exploitation a pool of reverse proxies is introduced. Use reverse proxies in DMZ to isolate OpenStack services inside of the cloud that may be vulnerable to attacks when directly accessed from the Internet.

Install a minimal set of packages on the latest vanilla Linux distribution (Ubuntu) with all security patches installed. Only web server components must be provided with latest security updates. No OpenStack components are exposed in DMZ.

Reverse proxies can be also used as:
- a load balancer
- a web accelerator to accelerate encryption for TLS connections
- a security gateway to filter out malformed requests

We recommend Nginx as a lightweight reverse proxy server with a support of encryption and caching.

DDoS protection

Criminals utilize various techniques such as attempted DNS reflection attacks or L7 HTTP floods involving large botnets to force organizations to pay a ransom.

According to the recent Kaspersky Lab DDoS attacks report we see the following numbers characterizing DDoS attacks nowadays:
- 25 and 200 bitcoins ($6,500 – $52,500) - is the ransom that the cybercriminal group DD4BC (Distributed Denial of Service for Bitcoin) asked victims to pay to terminate DDoS attack
- 275,000 HTTP requests per second were made by the attack using an adware extension in Internet browsers
- 109-179 Gbps SYN and DNS flood was generated by the Linux botnet
- SYN, TCP, and HTTP DDoS attacks were the most popular
- channel overloading attacks were: NTP amplification, SSDP amplification, and RIPv1 amplification which reached 40 Gbps
- 45.6% of DDoS attacks were made by Linux-based botnets, 54.4% - by Windows-based botnets
- 320 hours was the longest DDoS attack
- The botnet with about 2,000 zombies generated the HTTPS flood attack reaching 150 Mbps
When mitigating DDoS attack, good practice is to cooperate closely with upstream internet service providers and ISP CERT teams. ISP’s could help mitigate DDoS attacks as they could provide additional mitigation techniques, BGP blackholing to note only one of few.

**Firewall-as-a-Service**

The OpenStack Firewall-as-a-Service plugin helps configuring firewall rules and policies on firewalls and IDPS. It can be used in addition to Security Groups that offer filtering of East-West traffic in an environment. However, Security Groups and FWaaS have several limitations. Security Groups work at the instance level, and FWaaS - on virtual routers within a project. FWaaS operates with a single set of rules applied for all virtual routers on all subnets within a project, which limits flexibility of such a solution.

Therefore, we recommend using Palo Alto Networks VM-series firewall as a virtual network function (VNF). This solution helps securing East-West traffic between applications by steering it to VM-series firewall to detect malicious activity such as generated by advanced persistent threats (APT). This solution perfectly fits the needs of a scalable and flexible cloud environment.

See also:
- [Advanced Security Service Insertion in OpenStack Cloud](#)
- [Use case: Palo Alto Networks VM-series deployment as a VNF on Mirantis OpenStack](#)

**Intrusion detection and prevention systems**

Intrusion Detection System (IDS) monitors network or applications activities for being malicious. Intrusion Prevention System (IPS) is aimed to identify malicious activity similarly to IDS, log it, block it, and finally send an intrusion report to an administrator. IPS can be considered like an extension to IDS that actively block detected cyber attacks, for example by triggering alarms in a system, dropping malicious packets, or even blocking intruder’s IP address. Network IDPS is a part of commercial NGFW, NGIPS, and UTM solutions.

There two types of IDPS:
- Network-based
- Host-based

**Network-based IDPS**

Network-based IDS is a more suitable solution to deploy in the cloud infrastructure. There are two types of Network IDS:
- signature-based detection
- anomaly-based detection
They work more or less the same way performing real-time traffic scan and analysis, logging, protocol analysis, and content detection.

They can be used to detect:
- general purpose cyberattacks
- probes initiated as a part of targeted attacks
- policy violations, for example, usage of prohibited communication tools, social networks, TOR access points, bitcoin miners, so on.

There are several open source solutions available: Suricata, Snort, and Bro.
For Snort IDPS Snorby, Base, or Squil GUI applications are available that help visualizing information from logs.

Download rulesets for Snort and Suricata from the EmergingThreats repository.

If you want to improve detection capabilities of your IDPS we recommend purchasing ET Pro ruleset as it provides more frequent updates and extra rules to block targeted attacks such as malware command and control, DoS attacks, botnets, informational events, exploits, vulnerabilities, SCADA network protocols, exploit kit activity, and more.

IDPS can be connected to a channel to monitor a network traffic in several ways:
- Inline
  
  **Note:**
  This may affect a network channel throughput and a failure of IDPS may lead to disconnecting an internal network from external resources.

- using switch spanning port
  
  **Note:**
  You can use vSwitch Switched Port Analyzer (SPAN) or Remote Switched Port Analyzer (RSPAN) for port mirroring.

- using a network tap
  
  **Note:**
  You can use Tap-as-a-Service (TaaS) - a project developed to introduce the functionality of port mirroring in OpenStack Neutron provisioned networks.

- listening directly to a physical interfaces on a host (for example, on a compute node)

IDPS can be deployed:
- inline with a Firewall on a gateway when using the “service leg” DMZ
- on a compute node to listen to a network traffic related to projects hosted on this node
- on a network controller
on a VM - is the preferable way as it allows easy scaling, migration, and deployment. You need only to mirror traffic from any VM to the monitoring VM with IDPS on board.

See also:

- SANS Network IDS & IPS Deployment Strategies
- Tap-as-a-Service project
- Tap-as-a-Service: Enabling Traffic Monitoring in OpenStack Clouds
- Tap-as-a-Service (TaaS): Port Monitoring for Neutron Networks

Host-based IDPS

Host and platform security plays crucial role in the whole solution security. When securing your cloud, deploy Linux Audit Daemon (auditd) on all OpenStack and Fuel nodes. If deploying heterogeneous cloud, consider the open source OSSEC Host-based IDS for Linux, Solaris, AIX, HP-UX, BSD, Windows, Mac and vSphere/ESXI.

IDPS acceleration

If you want to sniff more than 1 Gbps channel we recommend using a packet steering solution such as PF_RING. PF_RING is an open source packet processing framework for Linux used to provide Direct NIC Access (DNA) to bypass kernel for line rate RX/TX packet processing. This means that we do not need to use CPU cycles to process network packets by kernel. PF_RING sends packets directly from NIC to IDS application bypassing kernel. This is called zero-copy (ZC, 0-copy) and helps improving performance of network traffic analysis.

PF_RING also supports 1-copy mode for non-Intel NICs and wireless connections. These packets can be injected into 0-copy stream.

PF_RING ZC can send packets to a VM with an IDPS installed on it.

The following solutions support PF_RING ZC:

- Snort, Suricata, Bro, Wireshark
- KVM
- Docker containers

PF_RING supports the NIC adapters 1/10/40/100 Gbit by the following vendors:

- Intel
- Accolade Technologies
- MYRICOM
- telesoft
- Napatech
- Mellanox Technologies
See also:

- Use case: [IDPS Suricata deployment as a VNF on Mirantis OpenStack with OpenContrail](#)
- [Using Suricata over PF_RING](#)
- [Install Suricata with PF_RING](#)
- ntop.org

**Next-generation solutions**

To reduce infrastructure complexity and vendor management effort we recommend using NGFW/NGIPS/UTM solutions instead of separate Firewall and IDPS solutions. NGFW/NGIPS/UTM may include on board the following modules:

- firewalls L3,4,7
- network IDPS
- gateway antivirus
- gateway anti-spam
- VPN
- content filtering
- load balancing
- DLP

**Cloud antivirus**

You may consider installing a cloud antivirus solution to enforce security of the environment. An antivirus can be thought as an advanced Host-based IPS. It is possible to install a regular enterprise antivirus solution on VMs that does not know anything about your cloud environment and manage them using the administration console supplied by a vendor. The problem with regular antiviruses is that being offline for a long time VMs have not updates antivirus databases making AV protection less efficient. Therefore, specially designed for virtualization antivirus solutions were introduced of two types:

- agent-based
- agentless

**Agent-based**

AV agent is deployed on every VM within the project and communicates with the module on a hypervisor. This creates computing resource consumption especially what makes this approach not efficient at scale.
Agentless

Agentless approach is based on Virtual Security Appliance (VSA) to scan files access by VMs and Network Security Appliance (NSA) to scan network traffic between VMs sitting on a host. Unfortunately, it is supported only by VMWare, Citrix, and Microsoft.

These solutions leverage the power of a hypervisor to reduce the load to VMs caused by regular antivirus applications. However, they have several limitations related to detection of 0-day attacks.

First, antiviruses even equipped with a heuristic engine in the majority of cases does not detect unknown 0-day malware. To avoid being detected modern cyber espionage platforms like EquationDrug utilized by Regin and Epic Turla use Windows kernel-mode rootkit driver to hide its files, registry keys, and processes by hooking some of the Native API functions.

Second, an antivirus solution can be disarmed by an evasive malware when antivirus is discovered on a targeted machine.

Third, a cloud environment in many cases is a heterogenous (hybrid) environment built on different operating systems and hypervisors that increases deployment and operational costs, and also makes your protection not flexible and vendor dependent.

Mirantis does not recommend heavy investments into malware protection on OpenStack nodes (Controllers and hypervisors). It is sufficient to configure built-in Linux firewall properly, use sensible access control policy (logins/passwords, ssh keys, etc), and deploy NGFW or IDPS for network traffic monitoring. For advanced protection against 0-day and targeted attacks, use sandboxing solutions where you can upload suspicious files and URLs for analysis.

Forensics solutions

If you need to make your environment bulletproof against 0-day attacks or reduce impact of a security incident. Your incident response team (IRT) should react fast to determine the scope and impact of the incident. In such case IRT needs automation tools:

- to analyse malware used to run an attack
- to figure out a vector of the attack, a breach in security, compromised services, lost data
- to minimize overall losses by rapid neutralizing the attack.

However, automated malware analysis systems can be used not only for forensics purposes, but for early detection of suspicious objects. For example, you can use any of IDPS such as
Suricata, Snort, or Bro to extract files from a mail or web unencrypted stream and send them to a malware analysis node called sandbox to verify behavior by opening or executing it in an isolated environment.

For forensics purposes we recommend creating Security Project within the Security Domain with Firewall, Security Groups, IDPS, malware analysis solutions. This way, once any project is compromised, the Gateway IP can be changed to connect to the internal Forensics network with a security tenant there full of forensics tools.
Use cases

Palo Alto Networks VM-series deployment as a VNF on Mirantis OpenStack

See the Installation Runbook for Palo Alto Networks virtual Firewall and Juniper Contraill plugin for Fuel.

IDPS Suricata deployment as a VNF on Mirantis OpenStack with OpenContrail

The use case shows Intrusion Detection and Prevention System (IDPS) Suricata installation as a virtual network function (VNF) on Mirantis OpenStack.

Software versions used:
- Mirantis OpenStack 7.0 (Kilo)
- Fuel 7.0
- Fuel Contraill plugin 3.0
- Suricata IDPS 3.0

To enable Suricata IDPS as a VNF, go through the following steps:
1. Deploy Mirantis OpenStack with Contraill SDN that will bring NFV into your cloud. Use Fuel, as it can deploy Contraill SDN using the Fuel Contraill plugin.
2. Create a VM image with the Suricata IDPS installed.
3. Configure the Contraill SDN to run an IDPS service instance (VNF) and steer traffic to this instance for further analysis.

Deploy Mirantis OpenStack with OpenContrail SDN

To deploy Mirantis OpenStack with OpenContrail SDN, do the following:
1. Deploy the Fuel Master node and allocate the pool of Fuel Slave nodes. You need at least one extra node for Contraill Controller. Assign ‘contraill-control’, ‘contraill-config’, and ‘contraill-db’ roles to the Contraill Controller node.
2. Download the Fuel Contraill plugin as well as the Contraill plugin documentation from the Mirantis Fuel plugins catalog: https://www.mirantis.com/validated-solution-integrations/fuel-plugins/
3. Follow the Fuel Contraill plugin documentation to install Contraill SDN with Mirantis OpenStack.
Install IDPS

To create VM image with IDPS, follow the instructions below:

1. Configure network interfaces. For example:
   
   ```
   #The loopback network interface
   auto lo
   iface lo inet loopback
   
   #The internal network interface
   auto eth0
   iface eth0 inet dhcp
   
   #The external network interface
   auto eth1
   iface eth1 inet dhcp
   
   #The management network interface (recommended)
   auto eth2
   iface eth2 inet dhcp
   ```

2. Install Suricata IDPS. Find Suricata installation guides by the link:
   

   **Note:**
   
   To enable NFQUEUE for IPS mode, install Netfilter packages and configure Suricata with `--enable-nfqueue` option before build. See the ‘IPS mode with NFQUEUE’ section for steps.

3. To capture traffic as it comes to a NIC avoiding packets reassembling by a network adapter, turn off offloads for a network interface you want to sniff. For example, configure eth0 with ‘ethtool’ as root:
   
   ```
   ethtool -K eth0 rx off tx off sg off tso off gso off rxvlan off txvlan off gro off lro off
   ```

   **Note:**
   
   If offloads are enabled, this may lead to reassembling incoming packets that results in changing packet structure and increasing its size. Suricata may not process reassembled packets correctly. For example, if GRO and LRO are enabled, you will see the error message when launching Suricata:
   
   `[ERRCODE: SC_ERR_PCAP_CREATE(21)] - Using Pcap capture with GRO or LRO activated can lead to capture problems.`
4. If you plan to extract files over 1 Mb in size from HTTP traffic, increase a value of the stream engine option 'stream.reassembly.depth' (default '1 Mb'), which controls the depth into a stream in which Suricata looks, or set to '0' for no limit in suricata.yaml.

You can run IDPS in two modes based on a service type:
- IDS (Contrail's Analyzer) - analyzing traffic and generating alerts
- IPS or inline (Contrail's Firewall) - blocking or modifying packets and generating alerts

Based on a service mode you can deploy IDPS in two ways:
- as a router (Contrail’s In-Network) - IDS or IPS is placed between at least two networks and packets are routed
- as a bridge (Contrail’s Transparent) - IDS or IPS forwards packets between network interfaces without modification

IDS mode

In IDS mode you can use the following actions:
- Pass - stops scanning the packet matched by a signature and skips to the end of all rules (only for this packet)
- Alert - IDS fires up an alert for the packet matched by a signature

If you want to set up IDS as a router between two networks, you need to configure iptables to forward and masquerade packets between networks.

To route traffic in *IDS mode* between internal network connected to the eth0 network interface of the IPS VM and external network connected to eth1, run the following commands:

```
sysctl -w net.ipv4.ip_forward=1
/etc/init.d/networking restart

iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
iptables -A FORWARD -i eth1 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
```

**Note:**

If you need to route the traffic that goes to the Internet, add an extra network interface to the IDPS VM and connect it to the Contrail internal network (SNAT).

Route packets from internal and external networks to the Contrail internal network, so Contrail Controller can deliver them to a BGP router connected to the Internet in its turn. The IDPS VM can be configured by analogy with a default Router instance created by Fuel to connect default internal 'net04' and external 'net04_ext' networks.
You can also use In-Network NAT mode to simplify packets routing between networks. In this mode return traffic does not need to be routed to the source network.

Test the **IDS mode** on the created VM:

1. Create `'/etc/suricata/rules/test.rules'` file and write the following rule:
   
   ```
   "alert http any any -> any any (msg:"Alarm detected"; content:"Alarm"; nocase; classtype:policy-violation; sid:1; rev:1;)
   ```
   
2. Add `test.rules` to a list of rules in the ‘suricata.yaml’ configuration file.

3. Reboot IDPS VM or start Suricata manually in a daemon mode:
   ```
   sudo suricata -c /etc/suricata/suricata.yaml -i eth0 -D
   ```

4. Verify if suricata.log contains no errors after Suricata starts up:
   ```
   tail -f /etc/suricata/suricata.log
   ```

5. Make any HTTP request with the word “Alarm”. For example:
   ```
   curl http://google.com/Alarm
   ```

6. View the fast.log to check if Suricata generated the appropriate alert message: “**Alarm detected**”:
   ```
   tail -f /etc/suricata/fast.log
   ```

**IPS mode**

In **IPS mode** you can block traffic bridged between two network interfaces using the following actions:

- **Drop** - a packet containing a signature is discarded immediately and will not be sent any further. The receiver does not receive a message resulting in a time-out. All subsequent packets of a flow are dropped.
- **Reject** - an active rejection of the packet, both a receiver and sender receive a reject packet. If the packet concerns TCP, it will be a reset-packet, otherwise it will be an ICMP-error packet for all other protocols.

**Note:**

Suricata generates an alert in both IPS modes.

To enable **IPS or inline mode** (the ‘Firewall’ Contrail’s service type), you can use:

- NFQ (Netfilter on Linux)

**Note:**

NFQ supports multiple queues processing, which you should specify explicitly in iptables rules and suricata command line options.

For example, you can configure load balancing with NFQ as follows:

```
iptables -A INPUT -j NFQUEUE --queue-balance 0:3
suricata -c /etc/suricata/suricata.yaml -q 0 -q 1 -q 2 -q 3
```
- IPFW (a divert socket on FreeBSD)
- AF_PACKET (Level 2 Linux bridge), which supports automatic load balancing for better performance.
- PF_RING

Note:
You can improve your performance with PF_RING ZC if your NIC supports Zero Copy (ZC) mode as well.

IPS mode using NFQ
NFQUEUE is an iptables and ip6tables target entity that delegate the decision on packets to a user space software like IPS.

To enable IPS mode using NFQ:
1. Install Netfilter packages:
   ```
sudo apt-get -y install libnetfilter-queue-dev libnetfilter-queue1
libnfnetlink-dev libnfnetlink0
   ```
2. Configure Suricata with ‘--enable-nfqueue’ option. Then build and install.
3. Configure iptables.
   For example, to scan bridged packets, add the rule:
   ```
sudo iptables -I FORWARD -j NFQUEUE
   ```
   To use ‘repeat’ Suricata NFQ mode, add the rule below specifying a source chain you need:
   ```
iptables -I FORWARD -m mark ! --mark $MARK/$MASK -j NFQUEUE
   ```

   This rule forwards packets to NFQUEUE only if they do not have a specified mark that can be set by Suricata after packet processing.

Warning:
If you stop Suricata, the packets that come into ‘NFQUEUE’ will not be processed and, as a result, will not be passed further.

Note:
On Linux >= 3.6, you can set the ‘fail-open’ option to ‘yes’ in ‘suricata.yaml’ to make the kernel accept the packet if Suricata is not able to keep pace.

4. Configure Suricata NFQ modes in the ‘suricata.yaml’ configuration file:
   a. `accept` - in default NFQ mode Suricata generates a terminal verdict: pass or drop. A packet will not be inspected by the rest of the iptables rules.
   b. `repeat` - Suricata generates a non-terminal verdict and mark the packets that will be re-injected again at the first rule of iptables. Add the following rule to iptables:
      ```
      iptables -I FORWARD -m mark ! --mark $MARK/$MASK -j NFQUEUE
      ```
   c. `route` - if you want a packet to be sent to another queue after an ‘ACCEPT’ decision, set mode to ‘route’ and set ‘route-queue’ value. You can use a route mode to scan packets with multiple network scanners on the same VM.
The example of the NFQ configuration in 'suricata.yaml':

```yaml
nfq:
    mode: accept    # nfq mode: accept, repeat, route
    repeat-mark: 1  # used for repeat mode to mark a packet
    repeat-mask: 1  # used for repeat mode to mark a packet
    route-queue: 2  # for 'route' mode
    batchcount: 20 # max length of a batching verdict cache
    fail-open: yes  # a packet is accepted when queue is full
```

5. Start Suricata to filter packets in ‘NFQUEUE’:

```
suricata -c /etc/suricata/suricata.yaml -q 0 -D
```

**Note:**

By default all incoming packets go the queue with the number ‘0’. However, you can define the queue number explicitly:

```
iptables -A FORWARD -j NFQUEUE --queue-num 0
```

6. Test IPS mode with NFQ.

a. Modify the test rule in the '/etc/suricata/rules/test.rules' file to drop or reject packets:

```
drop http any any -> any any (msg:"Alarm detected"; content:"Alarm";
nocase; classtype:policy-violation; sid:1; rev:1;)
```

or

```
reject http any any -> any any (msg:"Alarm detected"; content:"Alarm";
nocase; classtype:policy-violation; sid:1; rev:1;)
```

b. Start Suricata to scan the queues with forwarded packets:

```
suricata -c /etc/suricata/suricata.yaml -q 0 -q 1 -D
```

c. Download the 'test' file with the 'Alarm' word inside on the IPS VM. For example:

```
wget http://<web server IP>:8080/test
```

d. Verify that wget successfully downloads the file.

e. Add the following rules to iptables on IPS VM to forward incoming and outgoing packets:

```
iptables -A INPUT -j NFQUEUE
iptables -A OUTPUT -j NFQUEUE --queue-num 1
```

f. Download the 'test' file again:

```
wget http://<web server IP>:8080/test
```

g. Suricata should block the downloading file.

Example of output:

```
$ wget http://10.20.0.2:8080/test
Connecting to 10.20.0.2:8080... connected.
HTTP request sent, awaiting response…
```
h. Confirm that ‘/var/log/suricata/fast.log’ contains the corresponding alert message showing the Suricata dropped the packet:

```
[Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP}
10.20.0.2:8080 -> 10.20.0.8:49628
```

**IPS mode using AF_PACKET**

AF_PACKET establishes a software bridge between two interfaces by copying packet from one interface to another (and reverse).

To enable IPS mode using AF_PACKET Linux bridge, run the following steps:

1. Edit the ‘af-packet’ section in the ‘suricata.yaml’ configuration file:

   ```yaml
   af-packet:
   - interface: eth0
     threads: auto
     defrag: yes
     cluster-type: cluster_flow
     cluster-id: 98
     copy-mode: ips
     copy-iface: eth1
     buffer-size: 64535
     use-mmap: yes
   - interface: eth1
     threads: auto
     cluster-id: 97
     defrag: yes
     cluster-type: cluster_flow
     copy-mode: ips
     copy-iface: eth0
     buffer-size: 64535
     use-mmap: yes
   ```

**Note:**

‘cluster-id’ is used to group threads for a corresponding interface when load balancing. ‘cluster-id’ values should be different for every interface.

2. Start Suricata with ‘--af-packet’ option:

   ```
   suricata -c /etc/suricata/suricata.yaml --af-packet -D
   ```

3. Test if Suricata turned on an IPS mode.
   a. Modify the test rule in the ‘/etc/suricata/rules/test.rules’ file to drop or reject packets:
“drop http any any -> any any (msg:"Alarm detected"; content:"Alarm"; nocase; classtype:policy-violation; sid:1; rev:1;)”

or

“reject http any any -> any any (msg:"Alarm detected"; content:"Alarm"; nocase; classtype:policy-violation; sid:1; rev:1;)”

b. View http traffic on the destination interface eth1 on the IPS VM:

   `sudo tcpdump -i eth1 tcp port 8080 -A -w tcpdump.output`

c. Download the ‘test’ file with the ‘Alarm’ word inside on the IPS VM. For example:

   `wget http://<web server IP>:8080/test`

d. Terminate tcpdump and verify if the ‘test’ file with the ‘Alarm’ word was blocked by IPS in the tcpdump output log that contains traffic bridged to the destination interface eth1.

**IDPS mode using PF_RING**

PF_RING is a Linux network socket that use NAPI to copy packets from the NIC to the PF_RING circular buffer, and then the user space application reads packets from the ring.

**Note:**

For servers with physical network adapters you can use ZC, Napatech, Myricom, Linux TCP/IP Stack injection, Sysdig, Link Aggregation, Libzero consumer, DAG, DNA modules. You can also leverage PF_RING with ZC on a KVM virtual machine.

To enable IPS mode using vanilla PF_RING:

1. Install PF_RING packages:

   `sudo apt-get install build-essential bison flex linux-headers-$(uname -r) libnuma-dev`

2. Download the latest PF_RING library, extract, and build the library:

   `wget http://sourceforge.net/projects/ntop/files/PF_RING/PF_RING-6.2.0.tar.gz`

   `tar -xvzf PF_RING-6.2.0.tar.gz`

   `cd PF_RING-6.2.0/`

   `make`

   `cd kernel; sudo make install`

   `cd ../userland/lib; sudo make install`

   `cd ../userland/libpcap; ./configure; make`

   `sudo cp libpcap* /usr/local/lib/; sudo cp pcap.h /usr/local/include/`

   **Note:**

   Verify that new libpcap library can be found by Suricata when configuring, for example in ‘/usr/local/lib/’. Otherwise you will see the following warning message:

   `WARNING! libcap-ng library not found`

3. Verify that the PF_RING Linux kernel module was successfully loaded:

   `modinfo pf_ring`
cat /proc/net/pf_ring/info

4. Configure Suricata with `--enable-pfring` option and path to libs and include headers. For example:

```bash
LIBS="-lrt -lnuma"
./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var \ 
--enable-pfring --with-libpfring-includes=/usr/local/include \ 
--with-libpfring-libraries=/usr/local/lib
```

5. Build and install Suricata:

```bash
make
sudo make install-full
```

6. To run Suricata with PF_RING, enter:

```bash
sudo suricata --pfring-int=eth0 --pfring-cluster-id=99 \ 
--pfring-cluster-type=cluster_flow \ 
-c /etc/suricata/suricata.yaml -D
```

7. You can also tune up PF_RING in `suricata.yaml`. The `eth0` interface is enabled by default:

```yaml
pfring:
  - interface: eth0
    threads: 1
    cluster-id: 99
    cluster-type: cluster_flow
```

8. Test Suricata in IDS mode:
   a. Create `/etc/suricata/rules/test.rules` file and write the following rule:
      ```bash
      "alert http any any -> any any (msg:"Alarm detected"; content:"Alarm"; nocase; classtype:policy-violation; sid:1; rev:1;)
      ```
   b. Add `test.rules` to a list of rules in the `suricata.yaml` configuration file.
   c. Start Suricata manually in a daemon mode:
      ```bash
      sudo suricata --pfring-int=eth0 --pfring-cluster-id=99 \ 
      --pfring-cluster-type=cluster_flow \ 
      -c /etc/suricata/suricata.yaml -D
      ```
      Or update the active ruleset without rebooting Suricata:
      ```bash
      sudo kill -USR2 <suricata pid>
      ```
   d. Verify if suricata.log contains no errors after Suricata starts up:
      ```bash
      tail -f /etc/suricata/suricata.log
      ```
   e. Make any HTTP request with the word “Alarm”. For example:
      ```bash
      curl http://google.com/Alarm
      ```
   f. View the fast.log to check if Suricata generated the appropriate alert message:
      ```bash
      "Alarm detected":
      ```
      ```bash
      tail -f /etc/suricata/fast.log
      ```

See also:
File extraction in Suricata

The file extraction feature has been included in Suricata since the version 2.0. The feature enables file extraction from HTTP and SMTP traffic. You can take advantage of this feature when you want to analyze incoming files from Web or mail traffic in a sandbox or multiscanner. To enable file extraction feature, follow the steps below:

1. In the ‘test.rule’ file Create or modify a rule to add ‘filestore’ option:
   ```
   "alert http any any -> any any (msg:"Alarm detected"; content:"Alarm"; nocase; classtype:policy-violation; filestore; sid:1; rev:1;)
   ```
2. Verify that you have a proper value for the stream engine option ‘stream.reassembly.depth’ (default 1 Mb) in ‘suricata.yaml’. Increase the value for files >1Mb or set to ‘0’ for no limit.
3. Verify that you turned off the offloads rx, tx, sg, tso, gso, rxvlan, txvlan, gro, lro:
   ```
   ethtool -k eth0
   ```
4. Start Suricata or reload the active ruleset:
   ```
   kill -USR2 <suricata pid>
   ```
5. Download the ‘test’ text file containing ‘Alarm’ word inside on the IDPS VM:
   ```
   wget http://<web server IP>:8080/test
   ```
6. Go to the folder with extracted files:
   ```
   cd /var/log/suricata/files
   ```
7. View the ‘file.1’ content if it equals to the downloaded ‘test’ file:
   ```
   sudo cat file.1
   ```
   Expected output:
   ```
   Alarm
   ```
8. Open ‘file.1.meta’ to see the file’s metadata:
   ```
   sudo cat file.1.meta
   ```
   Expected output:
   ```
   TIME: 05/12/2016-18:02:58.514611
   SRC IP: 10.20.0.2
   DST IP: 10.20.0.8
   PROTO: 6
   SRC PORT: 8080
   DST PORT: 41632
   APP PROTO: http
   HTTP URI: /test
   HTTP HOST: 10.20.0.2
   HTTP REFERER: <unknown>
   HTTP USER AGENT: Wget/1.15 (linux-gnu)
   FILENAME: /test
   MAGIC: ASCII text
More rules to detect and extract Windows and Linux executables are below:

- alert http any any -> any any (msg:"==ELF file=="; content:"ELF"; distance:0; classtype:policy-violation; filestore;sid:3; rev:1;)
- alert http any any -> any any (msg:"==PE file=="; content:"|0D 0A 0D 0A|MZ"; distance:0; classtype:policy-violation; filestore;sid:4; rev:1;)
- alert http any any -> any any (msg:"==EXE file=="; fileext:"exe"; classtype:policy-violation; filestore;sid:14; rev:1;)

See also:
- Suricata file extraction [https://suricata-ids.org/tag/file-extraction/](https://suricata-ids.org/tag/file-extraction/)

Additional Suricata configuration

When you deploy Suricata on a service instance, you can start Suricata automatically on system boot up and run in a daemon mode.

To run Suricata in a daemon mode, you need to disable console output and set it to a file in the suricata.yaml configuration file:

```yaml
outputs:
- console:
  enabled: no
- file:
  enabled: yes
  filename: /var/log/suricata.log
```

To start Suricata on system boot up, create the initialization script `/etc/init/suricata.conf`:

```bash
# suricata
description "IDPS Daemon"
start on runlevel [2345]
stop on runlevel [!2345]
expect fork
exec suricata -D --pidfile /var/run/suricata.pid -c /etc/suricata/suricata.yaml -i eth0
```

**Note:**

You can also specify the `--pidfile` option in `suricata.yaml`.

Alerts management with Snorby and Barnyard2

Barnyard2 processes the alerts generated by IPS and adds them into a database. Snorby is the visual front end to the event data that is written into a database.
IDPS as a VNF

In OpenStack environments with a network functions virtualization infrastructure (NFVI) enabled you can run an IDPS instance as a virtualized network function (VNF). VNFs are building blocks that you can use to create a scalable service that includes a sequence of virtual functions in service chaining.

To enable IDPS as a VNF, follow the steps:

1. Once IDPS VM is configured and verified, upload the IDPS VM image to your cloud environment using OpenStack Dashboard. QCOW2 or VMDK formats of the image are preferable.

2. Go to the Contrail web UI. For example: https://172.16.0.3:8143/

3. Open the ‘Service Templates’ panel in the ‘Configure’ tab.

4. Create an IPS template service.
   a. You can choose out of three Service Modes for your service:
      - In-Network or routed mode - service VM instance is between at least two networks and packets are routed. Examples include NAT, Layer 3 firewall, load balancer, HTTP proxy, and so on.
      - In-Network NAT - similar to in-network mode, however, return traffic does not need to be routed to the source network. In-network-nat mode is particularly useful for NAT service.
      - Transparent or bridge mode - is transparent for communication between instances and packets are not modified. The transparent mode fits Layer 2 firewall and IDPS.
   b. Service Type specifies the number of interfaces for a service.
      - For ‘Firewall’ type Contrail allocates at least 2 interfaces: ingress and egress.
      - For ‘Analyzer’ type - at least 1 interface is needed.

See also:
- Suricata Snorby and Barnyard2 set up guide
  https://redmine.openinfosecfoundation.org/projects/suricata/wiki/suricata_snorby_and_barnyard2_set_up_guide
c. In Advanced Options you can enable Service Scaling, select Availability Zone for your service instances, and choose an instance flavor.
5. Then switch to ‘Service instances’ panel, where you can create a ‘ips-instance’ service instance based on the available service template and connect network interfaces to internal ‘net04’ and external ‘net04_ext’ networks. Thus, ‘ips-instance’ will act as an alternative router connecting two networks.

   ![Image of Create Service Instance]

6. Next go to the ‘Networking->Policies’ panel to create a service policy. The ‘ips-policy’ tells Contrail to pass any traffic between ‘net04’ and ‘net04_ext’ to the ‘ips-instance’ service instance.

   ![Image of Create Policy]

   Contrail provides traffic mirroring feature as well.

7. Then assign the created ‘ips-policy’ policy to the affected networks ‘net04’ and ‘net04_ext’ in the ‘Networking->Networks’ panel.

8. Test your service chaining.
   a. Create a VM ‘TestVM’ from where you can download malicious content from external network.
   b. Configure iptables to route traffic between internal and external traffic (a router mode).
Note:
You can also test your IPS service instance in a bridge mode by installing it between the default router and external network.

c. The current network topology with ‘ips-instance’ and the ‘TestVM’ running in the internal network will look like:

![Network Topology Diagram]

d. Verify if Contrail correctly direct traffic between networks to the service instance.
   - From the ‘TestVM’ ping the ‘net04_ext’ gateway or any other than ‘ips-instance’ VM:
     ```
ping 172.16.0.1
     ```
   - On the ‘ips-instance’ VM run:
     ```
sudo tcpdump -i eth0
     ```
   - Verify that you can see ICMP packets sent between two networks on the ‘ips-instance’ VM.

e. Set up a web server in external network and create a test file with the ‘Alarm’ word inside on it.

f. Download the test file from ‘TestVM’. For example:
   ```
curl http://172.16.0.30:8080/test
   ```

g. View the fast.log on ‘ips-instance’ VM to check if Suricata generated the appropriate alert message: “Alarm detected”:
   ```
tail -f /etc/suricata/fast.log
   ```

See also:
Incident detection and prevention with IDPS and malware sandbox

This use case will demonstrate how to detect an attack using IDPS Suricata and a malware sandbox.

[to do]

See also:
- Watch the demo 'IDPS Suricata as a VNF' at OpenStack Summit 2016 in Austin: https://youtu.be/uakDqkKmfiE?t=1250

Modeling targeted attacks

To verify if your private cloud environment is well protected against such an evasive threat as targeted attack, you can create a simulation tool or script to model an APT behavior inside of the network.

The tool should cover the following stages of the APT model:

1. Penetration using:
   a. spear-phishing
   b. watering hole
   c. USB removable storage
2. Delivery of the APT kit
3. Lateral movements and EoP
4. Data collection
5. Data exfiltration

[to do]
Appendix A

Compliance with security standards

See:

- Payment Card Industry (PCI) Data Security Standard
- US Government agencies security standard: FedRAMP/FISMA
- HIPAA Security and Privacy standards
- ISO 27001:2013

Recommended security solutions

<table>
<thead>
<tr>
<th>Class of tool</th>
<th>Vendor</th>
<th>Solution</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewall</td>
<td>Cisco</td>
<td>Firepower NGFW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Palo Alto Networks</td>
<td>Palo Alto NGFW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Juniper</td>
<td>SRX Series: Next-Generation Anti-Threat Firewall, vSRX Virtual Firewall, cSRX Container Firewall</td>
<td></td>
</tr>
<tr>
<td>IDPS</td>
<td>Open Source</td>
<td>Snort</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Open Source</td>
<td>Suricata</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cisco</td>
<td>Firepower NGFW</td>
<td>NGFW includes IDPS</td>
</tr>
<tr>
<td></td>
<td>Palo Alto Networks</td>
<td>Palo Alto NGFW</td>
<td>NGFW includes IDPS</td>
</tr>
<tr>
<td>Load balancers</td>
<td>Open Source</td>
<td>HAProxy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Imperva</td>
<td>Imperva hardware appliances or Incapsula SaaS solution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F5</td>
<td>F5 Load balancer</td>
<td></td>
</tr>
<tr>
<td>DDoS Protection</td>
<td>Arbor Networks</td>
<td>Arbor Cloud</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Imperva</td>
<td>Incapsula DDoS Protection</td>
<td></td>
</tr>
</tbody>
</table>
## IT host security policy

To design your own IT host security policy, take into account the security recommendations provided by:

- Security Technical Implementation Guides (STIGs)
- Federal Risk and Authorization Management Program (FedRAMP)
- Payment Card Industry Data Security Standard (PCI DSS)

**STIG hardening recommendations**

The Security Technical Implementation Guides (STIGs) are the configuration standards for secure installation and maintenance of computer software and hardware introduced by Defense Information Systems Agency (DISA) in support of the United States Department of Defense (DoD). The guides include recommended administrative processes to reduce exploitation possibility. STIG scanning software is used to implement and validate proper configuration.

<table>
<thead>
<tr>
<th>LMA toolchain</th>
<th>Mirantis</th>
<th>StackLight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mirantis</td>
<td>Zabbix</td>
</tr>
<tr>
<td>Splunk</td>
<td>Splunk</td>
<td></td>
</tr>
<tr>
<td>IBM</td>
<td>Q-radar</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Forensics</th>
<th>Open Source</th>
<th>Cuckoo Sandbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>FireEye</td>
<td>Network Forensics Platform (PX series)</td>
<td></td>
</tr>
<tr>
<td>FireEye</td>
<td>Investigation Analysis system (IA series)</td>
<td></td>
</tr>
<tr>
<td>FireEye</td>
<td>Malware Analysis (AX series)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sandbox</th>
<th>Open Source</th>
<th>Cuckoo Sandbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juniper</td>
<td>Sky Advanced Threat Prevention</td>
<td></td>
</tr>
<tr>
<td>FireEye</td>
<td>Malware Analysis (AX series)</td>
<td>Windows and Mac applications analysis</td>
</tr>
<tr>
<td>Fortinet</td>
<td>FortiSandbox</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Anti-Spam</th>
<th>FireEye</th>
<th>eSPM</th>
</tr>
</thead>
</table>
Verify that your Linux host comply with the following STIG recommendations:

1. The system must not permit interactive boot. To disable the ability for users to perform interactive startups, edit the file `/etc/sysconfig/init`. Add or correct the line: “PROMPT=no”. The ‘PROMPT’ option allows the console user to perform an interactive system startup, in which it is possible to select the set of services which are started on boot.

2. All rsyslog-generated log files must be owned by ‘root’ and have mode 0600 or less permissive. The log files generated by ‘rsyslog’ contain valuable information regarding system configuration, user authentication, and other such information. Log files should be protected from unauthorized access. The owner of all log files written by ‘rsyslog’ should be ‘root’.

3. The system must set a maximum audit log file size. The total storage for audit log files must be large enough to retain log information over the period required. This is a function of the maximum log file size and the number of logs retained. Determine the amount of audit data (in megabytes - at least 6 Mb) that should be retained in each log file. Edit the file `/etc/audit/auditd.conf`. Add or modify the following line, substituting the correct value for [STOREMB]: "max_log_file = [STOREMB]".

4. The SSH daemon must set a timeout interval on idle sessions. Causing idle users to be automatically logged out guards against compromises one system leading trivially to compromises on another. SSH allows administrators to set an idle timeout interval. After this interval has passed, the idle user will be automatically logged out. To set an idle timeout interval, edit the following line in `/etc/ssh/sshd_config` as follows: 

```
ClientAliveInterval [interval]
```

The timeout [interval] is given in seconds. To have a timeout of 15 minutes, set [interval] to 900. If a shorter timeout has already been set for the login shell, that value will preempt any SSH setting made here. Keep in mind that some processes may stop SSH from correctly detecting that the user is idle.

2. The SSH daemon must not permit user environment settings. SSH environment options potentially allow users to bypass access restriction in some configurations. To ensure users are not able to present environment options to the SSH daemon, add or correct the following line in `/etc/ssh/sshd_config` as follows: "PermitUserEnvironment no".

3. The SNMP service must not use a default password. Presence of the default SNMP password enables querying of different system aspects and could result in unauthorized knowledge of the system. Edit `/etc/snmp/snmpd.conf`, remove default community string ‘public’. Upon doing that, restart the SNMP service.

4. The system default umask for the bash shell must be 077. The umask value influences the permissions assigned to files when they are created. A misconfigured umask value
could result in files with excessive permissions that can be read and/or written to by unauthorized users. To ensure the default umask for users of the Bash shell is set properly, add or correct the 'umask' setting in '/etc/bashrc' to read as follows: "umask 077".

5. The system default umask in /etc/profile must be 077. The umask value influences the permissions assigned to files when they are created. A misconfigured umask value could result in files with excessive permissions that can be read and/or written to by unauthorized users. To ensure the default umask controlled by '/etc/profile' is set properly, add or correct the 'umask' setting in '/etc/profile' to read as follows: "umask 077".

6. Auditing must be enabled at boot by setting a kernel parameter. Each process on the system carries an 'auditable' flag which indicates whether its activities can be audited. Although 'auditd' takes care of enabling this for all processes which launch after it does, adding the kernel argument ensures it is set for every process during boot. To ensure all processes can be audited, even those which start prior to the audit daemon, add the argument 'audit=1' to the kernel line in '/etc/grub.conf', as follows: "kernel /vmlinuz-version ro vga=ext root=/dev/VolGroup00/LogVol00 rhgb quiet audit=1".

7. The Bluetooth kernel module must be disabled preventing the kernel from loading the kernel module provides an additional safeguard against its activation. The kernel's module loading system can be configured to prevent loading of the Bluetooth module. Add the following to the appropriate '/etc/modprobe.d' configuration file to prevent the loading of the Bluetooth module: "install net-pf-31 /bin/false install bluetooth /bin/false".


See also:
- DISA STIGs documentation [http://iase.disa.mil/stigs/Documents/Forms/AllItems.aspx](http://iase.disa.mil/stigs/Documents/Forms/AllItems.aspx)

FedRamp recommendations

See the FedRamp guide for your security control level [https://www.fedramp.gov/resources/documents-2016/](https://www.fedramp.gov/resources/documents-2016/)

PCI DSS recommendations

Quick summary of PCI DSS recommendations are as follows:

1. Restrict the use of administrative functions to defined endpoint networks and devices, such as specific laptops or desktops that have been approved for such access.
2. Require multi-factor authentication for all administrative functions.
3. Ensure that all changes are implemented and tested properly. Consider requiring additional management oversight, above and beyond that which is required through the normal change-management process.
4. Separate administrative functions such that hypervisor administrators do not have the ability to modify, delete, or disable hypervisor audit logs.
5. Send hypervisor logs to physically separate, secured storage as close to real-time as possible.
6. Monitor audit logs to identify activities that could indicate a breach in the integrity of segmentation, security controls, or communication channels between workloads. Implement an automatic log analysis solution and develop scripts notifying of all potentially harmful actions, according to company security policy.
7. Separate duties for administrative functions, such that authentication credentials for the hypervisor do not have access to applications, data, or individual virtual components.
Examples

Example of incident investigation and analysis of CozyDuke APT

Backdoors, as well as APTs communicate with C&C servers to get commands from attackers and send back collected information. The network traffic is usually encrypted and sent via a usual http port. This prevents it from being detected by Network Data Leakage Prevention (DLP) or antivirus systems. However, every APT generates a specific traffic that contains unique network IoCs. Based on the analysis of network behavior of recent APTs it is possible to reveal patterns of targeted attacks to detect still unknown cyber espionage campaign.

For example, let us take a look at the communication protocol of the CozyDuke ("Office Monkeys") APT.

It sends http requests to www.sanjosemaristas.com C&C server, which replies with the specific for this victim set of payload modules. The transmitted data is Base64 encoded. Once we take Base64 off we still see some binary content that seems to be still encrypted.

If we open the backdoor in a disassembler we can find out that it uses RC4 symmetric encryption.

```assembly
.text:1001C630 push eax ; phKey
.text:1001C631 push 800001h ; dwFlags
.text:1001C636 push CALG_RC4 ; Algid
.text:1001C638 push [ebp+hProv] ; hProv
.text:1001C63E call ds:CryptGenKey
.text:1001C644 test eax, eax
```

Moreover, we can even find the RC4 key in the memory when the backdoor exports it in a BLOB format to be stored as a header before the encrypted data. The key is 16 bytes and generated every time a new session is created:
The network packet before being encoded with Base64 contains the key at the beginning, so the server can use it to decrypt the message:

The same key is used to encrypt local configuration file of the backdoor called racss.dat.
The XML config file shows the C&C server addresses. According to the information in the `<Servers>` section we see two URLs used to establish the connection with C&C server.

These seems to be the new ones as they are not mentioned in the CozyDuke APT report by Kaspersky Lab.

If we open the second server main page we will see that this is a legal sport website “CIF Southern Section”, not blacklisted:
As you can see, after the short analysis of the threat we found two new network IoCs that can be blacklisted in your cloud network to detect the malicious activity of “Office Monkeys”:

www.sanjosemaristas.com/app/index.php
www.cifss.org/product_thumb/index.php

Now you can add these network IoCs to the blacklists of NGFW/IDPS solutions deployed in your cloud. Once blacklists are updated you can trace if there are more infected machines in a subnet or within a target project.

For incident response it is important to reveal a security breach and fix it to prevent the attack in future. In case of “Office Monkeys” no exploits were used, but spear phishing and social engineering methods. In this case you needed to pay attention to SMTP traffic and create rules to block suspicious attached files according to the corporate security policy.

To prevent this type of attack in future you can:

- deploy antivirus on a mail server or gateway to filter attached files in SMTP traffic
- open attached documents in a sandbox to check their behaviour for being malicious
- mirror documents excluding active content such as embedded scripts
Example of threat modeling for Ceph RBD

The example highlights a threat modeling process for Ceph RBD. You can use Microsoft Threat Modeling Tool to draw a data flow diagram (DFD) and run modeling using the STRIDE threat model described previously.

After modeling is completed, the tool suggests a list of potential threats to be mitigated. The tool distributes the discovered threats between interactions. Therefore, the report suggests the list of threats by categories connected to a particular interaction (data flow) on the diagram. Below you can find modeling information by the Microsoft Threat Modeling Tool for each interaction.

Interaction: Monitor Data Flow
1. Spoofing of Destination Data Store Generic Data Store  [State: Not Started]  [Priority: High]

Category: Spoofing
Description: Ceph Monitor (controller) may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Ceph Monitor (controller). Consider using a standard authentication mechanism to identify the destination data store.

Justification: <no mitigation provided>

2. Data Store Inaccessible  [State: Not Started]  [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.

Justification: <no mitigation provided>


Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: <no mitigation provided>
4. Potential Excessive Resource Consumption for OS Process or Generic Data Store  [State: Not Started]  [Priority: High]

Category: Denial Of Service
Description: Does Qemu Process or Ceph Monitor (controller) take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.

Justification: <no mitigation provided>

5. Data Flow Sniffing  [State: Not Started]  [Priority: High]

Category: Information Disclosure
Description: Data flowing across Monitor Data Flow may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.

Justification: <no mitigation provided>

6. Data Store Denies Generic Data Store Potentially Writing Data  [State: Not Started]  [Priority: High]

Category: Repudiation
Description: Ceph Monitor (controller) claims that it did not write data received from an entity on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: <no mitigation provided>
7. The Generic Data Store Data Store Could Be Corrupted [State: Not Started] [Priority: High]

Category: Tampering
Description: Data flowing across Monitor Data Flow may be tampered with by an attacker. This may lead to corruption of Ceph Monitor (controller). Ensure the integrity of the data flow to the data store.

Justification: <no mitigation provided>

8. Authenticated Data Flow Compromised [State: Not Started] [Priority: High]

Category: Tampering
Description: An attacker can read or modify data transmitted over an authenticated dataflow.

Justification: <no mitigation provided>

Interaction: Monitor Data Flow

9. Weak Access Control for a Resource [State: Not Started] [Priority: High]

Category: Information Disclosure
Description: Improper data protection of Ceph Monitor (controller) can allow an attacker to read information not intended for disclosure. Review authorization settings.

Justification: <no mitigation provided>
10. Potential Data Repudiation by OS Process  [State: Not Started]  [Priority: High]

Category: Repudiation
Description: Qemu Process claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: <no mitigation provided>

11. Spoofing of Source Data Store Generic Data Store  [State: Not Started]  [Priority: High]

Category: Spoofing
Description: Ceph Monitor (controller) may be spoofed by an attacker and this may lead to incorrect data delivered to Qemu Process. Consider using a standard authentication mechanism to identify the source data store.

Justification: <no mitigation provided>

12. Spoofing the OS Process Process  [State: Not Started]  [Priority: High]

Category: Spoofing
Description: Qemu Process may be spoofed by an attacker and this may lead to information disclosure by Ceph Monitor (controller). Consider using a standard authentication mechanism to identify the destination process.

Justification: <no mitigation provided>

13. Potential Process Crash or Stop for OS Process  [State: Not Started]  [Priority: High]

Category: Denial Of Service
Description: Qemu Process crashes, halts, stops or runs slowly; in all cases violating an availability metric.

Justification: <no mitigation provided>

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

15. Data Store Inaccessible [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.
Justification: <no mitigation provided>

16. OS Process May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Ceph Monitor (controller) may be able to remotely execute code for Qemu Process.
Justification: <no mitigation provided>

17. Elevation by Changing the Execution Flow in OS Process [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Qemu Process in order to change the flow of program execution within Qemu Process to the attacker's choosing.
Justification: <no mitigation provided>

18. Authenticated Data Flow Compromised [State: Not Started] [Priority: High]

Category: Tampering
Description: An attacker can read or modify data transmitted over an authenticated dataflow.

Justification: <no mitigation provided>

Interaction: Monitor Data Flow

19. Spoofing of Source Data Store Ceph RBD Data Store (OSD) [State: Not Started] [Priority: High]

Category: Spoofing
Description: Ceph RBD Data Store (OSD) may be spoofed by an attacker and this may lead to incorrect data delivered to Ceph Monitor (controller). Consider using a standard authentication mechanism to identify the source data store.

Justification: <no mitigation provided>

20. Spoofing of Destination Data Store Ceph Monitor (controller) [State: Not Started] [Priority: High]

Category: Spoofing
Description: Ceph Monitor (controller) may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Ceph Monitor (controller). Consider using a standard authentication mechanism to identify the destination data store.

Justification: <no mitigation provided>

Category: Tampering
Description: An attacker can read or modify data transmitted over an authenticated dataflow.
Justification: <no mitigation provided>

Interaction: Monitor Data Flow

22. Spoofing of Source Data Store Ceph Monitor (controller)  [State: Not Started]  [Priority: High]

Category: Spoofing
Description: Ceph Monitor (controller) may be spoofed by an attacker and this may lead to incorrect data delivered to Ceph RBD Data Store (OSD). Consider using a standard authentication mechanism to identify the source data store.
Justification: <no mitigation provided>

23. Spoofing of Destination Data Store Ceph RBD Data Store (OSD)  [State: Not Started]  [Priority: High]

Category: Spoofing
Description: Ceph RBD Data Store (OSD) may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Ceph
RBD Data Store (OSD). Consider using a standard authentication mechanism to identify the destination data store.

Justification: <no mitigation provided>


Category: Tampering
Description: An attacker can read or modify data transmitted over an authenticated dataflow.

Justification: <no mitigation provided>

Interaction: RBD Data Flow

25. Spoofing the OS Process  [State: Not Started]  [Priority: High]

Category: Spoofing
Description: Qemu Process may be spoofed by an attacker and this may lead to information disclosure by Ceph RBD Data Store (OSD). Consider using a standard authentication mechanism to identify the destination process.

Justification: <no mitigation provided>

26. Spoofing of Source Data Store Generic Data Store  [State: Not Started]  [Priority: High]

Category: Spoofing
Description: Ceph RBD Data Store (OSD) may be spoofed by an attacker and this may lead to incorrect data delivered to Qemu Process. Consider using a standard authentication mechanism to identify the source data store.

Justification: <no mitigation provided>

27. Potential Data Repudiation by OS Process  [State: Not Started]  [Priority: High]

Category: Repudiation
Description: Qemu Process claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: <no mitigation provided>

28. Weak Access Control for a Resource  [State: Not Started]  [Priority: High]

Category: Information Disclosure
Description: Improper data protection of Ceph RBD Data Store (OSD) can allow an attacker to read information not intended for disclosure. Review authorization settings.

Justification: <no mitigation provided>

29. Potential Process Crash or Stop for OS Process  [State: Not Started]  [Priority: High]

Category: Denial Of Service
Description: Qemu Process crashes, halts, stops or runs slowly; in all cases violating an availability metric.

Justification: <no mitigation provided>


Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: <no mitigation provided>

31. Data Store Inaccessible [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.

Justification: <no mitigation provided>

32. OS Process May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Ceph RBD Data Store (OSD) may be able to remotely execute code for Qemu Process.

Justification: <no mitigation provided>

33. Elevation by Changing the Execution Flow in OS Process [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Qemu Process in order to change the flow of program execution within Qemu Process to the attacker's choosing.

Justification: <no mitigation provided>

34. Risks from Logging [State: Not Started] [Priority: High]

Category: Tampering
Description: Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log
35. Authenticated Data Flow Compromised  [State: Not Started]  [Priority: High]

Category: Tampering
Description: An attacker can read or modify data transmitted over an authenticated dataflow.
Justification: <no mitigation provided>

Interaction: RBD Data Flow

36. Risks from Logging  [State: Not Started]  [Priority: High]

Category: Tampering
Description: Log readers can come under attack via log files. Consider ways to canonicalize data in all logs. Implement a single reader for the logs, if possible, in order to reduce attack surface area. Be sure to understand and document log file elements which come from untrusted sources.
Justification: <no mitigation provided>

37. Spoofing of Destination Data Store Generic Data Store  [State: Not Started]  [Priority: High]
Category: Spoofing
Description: Ceph RBD Data Store (OSD) may be spoofed by an attacker and this may lead to data being written to the attacker’s target instead of Ceph RBD Data Store (OSD). Consider using a standard authentication mechanism to identify the destination data store.

Justification: <no mitigation provided>

38. The Generic Data Store Data Store Could Be Corrupted  [State: Not Started]  [Priority: High]
Category: Tampering
Description: Data flowing across RBD Data Flow may be tampered with by an attacker. This may lead to corruption of Ceph RBD Data Store (OSD). Ensure the integrity of the data flow to the data store.

Justification: <no mitigation provided>

39. Data Store Denies Generic Data Store Potentially Writing Data  [State: Not Started]  [Priority: High]
Category: Repudiation
Description: Ceph RBD Data Store (OSD) claims that it did not write data received from an entity on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: <no mitigation provided>

40. Data Flow Sniffing  [State: Not Started]  [Priority: High]
Category: Information Disclosure
Description: Data flowing across RBD Data Flow may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or
simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.

Justification: <no mitigation provided>

41. Potential Excessive Resource Consumption for OS Process or Generic Data Store  [State: Not Started]  [Priority: High]

Category: Denial Of Service
Description: Does Qemu Process or Ceph RBD Data Store (OSD) take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.

Justification: <no mitigation provided>

42. Data Flow Generic Data Flow Is Potentially Interrupted  [State: Not Started]  [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: <no mitigation provided>

43. Data Store Inaccessible  [State: Not Started]  [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.

Justification: <no mitigation provided>

44. Weak Credential Storage  [State: Not Started]  [Priority: High]

Category: Information Disclosure
Description: Credentials held at the server are often disclosed or tampered with and credentials
stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they’re stored.

Justification: <no mitigation provided>

45. Lower Trusted Subject Updates Logs  [State: Not Started]  [Priority: High]

Category: Repudiation
Description: If you have trust levels, is anyone other outside of the highest trust level allowed to log? Letting everyone write to your logs can lead to repudiation problems. Only allow trusted code to log.

Justification: <no mitigation provided>

46. Data Logs from an Unknown Source  [State: Not Started]  [Priority: High]

Category: Repudiation
Description: Do you accept logs from unknown or weakly authenticated users or systems? Identify and authenticate the source of the logs before accepting them.

Justification: <no mitigation provided>

47. Insufficient Auditing  [State: Not Started]  [Priority: High]

Category: Repudiation
Description: Does the log capture enough data to understand what happened in the past? Do your logs capture enough data to understand
an incident after the fact? Is such capture lightweight enough to be left on all the time? Do you have enough data to deal with repudiation claims? Make sure you log sufficient and appropriate data to handle a repudiation claims. You might want to talk to an audit expert as well as a privacy expert about your choice of data.

Justification: <no mitigation provided>

48. Potential Weak Protections for Audit Data  [State: Not Started]  [Priority: High]

Category: Repudiation
Description: Consider what happens when the audit mechanism comes under attack, including attempts to destroy the logs, or attack log analysis programs. Ensure access to the log is through a reference monitor, which controls read and write separately. Document what filters, if any, readers can rely on, or writers should expect

Justification: <no mitigation provided>

49. Authenticated Data Flow Compromised  [State: Not Started]  [Priority: High]

Category: Tampering
Description: An attacker can read or modify data transmitted over an authenticated dataflow.

Justification: <no mitigation provided>