**Threat modeling of Nova (High Level)**

**DRAFT - Work in Progress**

Nova is the project name for OpenStack Compute, a cloud computing fabric controller, the main part of an IaaS system.

The goal of this document is to present a High Level Threat Model analysis over view, discriminating main Security Objectives Assets, Adversaries, etc.

Following a top-down approach, this document describes High Level components, assets and main interactions. The idea is then to explode threat model analysis for relevant lower level components.

**1.1 System Overview**

|  |  |
| --- | --- |
| Application version | Nova Icehouse Release |
| Application  Description | OpenStack Compute service (nova) provides services to support the management of virtual machine instances at scale, instances that host multi-tiered applications, dev/test environments, "Big Data" crunching Hadoop clusters, and/or high performance computing. |
| Version | 0.1 |
| Participants | Cristian Fiorentino |
| Additional Info | Currently a Draft Analysis |

**1.2 Implementation Overview**

|  |  |
| --- | --- |
| Major Components | Nova API, Nova Controller, Nova Compute, Nova Scheduler, Nova Network, Nova Storage/Volume, Nova Conductor, Virtualization Drivers, DBMS, Messaging system, nova-client |
| Dependent components | Keystone, Neutron, Swift, Horizon, Cinder, Glance, OSLO, External Hypervisors |
| Description | Nova project implements Major Components described above. Several of them interact with external modules (Dependent components); some of them also implement alternative/subset functionality for these components. |
|  |  |

**1.3 System Assumptions (External Dependencies)**

|  |  |
| --- | --- |
| **ID** | **Description** |
| 1 | Nova components may be deployed in different nodes. As a general rule most components except Nova Compute could be in a single node; there could be an arbitrary number of Compute Nodes. |
| 2 | Internal Nova components persist information via an ORM mechanism (SQLAlchemy). |
| 3 | Nova interacts with different in Hypervisors via an extensible set of virtualization drivers (i.e. libvirt, VMWare). |
| 4 | Relevant nova configuration is persisted in nova.conf. Unless stated otherwise, the default values of the configuration file is used. |
| 5 | The communication channel among most Nova components and other OpenStack Service/modules contain sensitive information and should be protected (Cinder, Swift, Keytone, Neutron, etc.). |
| 6 | Hypervisors, Operating systems, databases, webservers, and physical machine are hardened. |
| 7 | The system admin follows Security best practice for managing compute resources. |
| 8 | Communication among internal Nova components is performed via messaging (OSLO/RabbitMQ), wrapped in the RPC call implementation. |
|  |  |

**2. Security Objectives**

|  |  |
| --- | --- |
| Cloud Computing fabric controller  Security objectives | 1. Authentication, Integrity and Confidentiality assured in communications among Nova components 2. Authentication, Integrity and Confidentiality assured among Nova components and external Nova modules/services 3. Authentication, Integrity and Confidentiality assured among Nova components and Hypervisors/VM instances 4. Isolated VM instances execution 5. Access restrictions for VMs/components/data in different tenants 6. Provide mechanisms for assuring Authorization, Integrity and Confidentiality for VM images/instances/snapshots while being managed/transmitted/persisted. 7. Provide Integrity, Confidentiality and Access Authorization for exposed Nova API/Compute 8. Assure Integrity, Confidentiality and Authentication access to persistence mechanisms. 9. Provide the means for configuring different public and private/management Network for VMs 10. Allow Network rules management configuration for VMs 11. Secure management (i.e. encryption, hashing) for sensitive data (i.e. VMs metadata, configuration, credentials) being transmitted/persisted. 12. Non-repudiation for transactions/sensitive calls 13. Assure less privilege for users/clients/agents/commands accessing Nova functionality 14. Assure Integrity, Confidentiality and Authorization for Image/Object/Storage repositories 15. In case of failure/error, fail intelligently 16. Not to expose sensitive information in Logs 17. Perform defend in depth strategies such as input validation/output sanitization for data coming from users/services or non-trusted sources. 18. Perform schema/data validation for incoming information in formats such as XML/JSON 19. Provide protections for insiders potentially accessing infrastructure without permissions. 20. Infrastructure files (code, configuration, repos, etc.) to be persisted with appropriate FS credentials/permissions. |

**3. 1 Component Interaction diagram**

High level threat-model diagram for Nova. Includes: main Nova internal components/processes, related external modules, their main interaction/data flows and the associated trust boundaries.

* *Reference*
  + *Red arcs: delimit trusted boundaries (among Nova components and external entities)*
  + *Blue arcs: delimit machine/host boundaries (as different deployment possible, only showed most common scenarios)*
  + *Brown arcs: delimit process boundaries (for main Nova processes/components)*

*(\*) RPC-call is accessed by most Nova components. Not showed in diagram for simplification only. Most interactions among Nova components in this diagram (arrows) are implemented via the messaging mechanism accessed thru this RPC-call.*

*(\*\*) ORM is accessed directly by several Nova components. Not showed in diagram for simplification only. Only Conductor interaction is showed given it’s one of the main purposes of this component.*



**3.2 Entry points**

The interfaces through which an attacker can request for resources or supply data

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Accessible To** |
| 1 | Nova Controller API’s | Either Nova/EC2 or S3 APIs |  |
| 2 | Conductor/Compute API’s | APIs generally deployed in different Hosts |  |
| 3 | Python-nova client | CLI for accessing Nova functionality |  |
| 4 | Nova-manage | CLI for managing Nova resoruces |  |
| 5 | DB | Via ORM, or directly via the DB |  |
| 6 | Messaging | Via OSLO/RabbitMQ interfaces |  |
| 7 | Nova Remote Desktop proxy | Such as VNC for accessing VMs instances |  |
| 8 | Virtualization Driver API | Such as libvirt API |  |
| 9 | Direct access to VMs | Such as telnet/SSH/port scans |  |
| 10 | Monitoring | Via monitoring/metering components such as Ceilometer/StackTach |  |
| 11 | Logging | Nova logs for different modules |  |

**3.3 Assets**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Details** | **Required Trust Levels** |
| 1 | User Data | Assets related to a user |  |
| 1.1 | Tenant user auth credentials (password) |  |  |
| 1.2 | Tenant User auth details | Information related to a user e.g., role, domain, username/id, groups and their relation |  |
| 1.3 | Tenant Admin auth credentials |  |  |
| 1.4 | Keystone Identity admin auth credentials |  |  |
| 1.5 | Personal data related to user | PII |  |
| 1.6 | Catalog data related to user and tenant |  |  |
| 1.7 | Auth Token related to a user |  |  |
|  |  |  |  |
| 2 | VMs and Nodes |  |  |
| 2.1 | VM Metadata |  |  |
| 2.2 | VM Credentials |  |  |
| 2.3 | Node Metadata |  |  |
| 2.4 | Node Credentials |  |  |
| 2.5 | Hypervisor credentials/API |  |  |
| 2.6 | VM persistence/files |  |  |
| 2.7 | VMs data in transit |  |  |
|  |  |  |  |
| 3. | Infrastructure/Network |  |  |
| 3.1 | Stack/infra Network interfaces/configuration |  |  |
| 3.2 | VMs Network interfaces/configuration |  |  |
| 3.3 | Security/FW rules |  |  |
| 3.4 | Load Balancer |  |  |
| 3.5 | Proxies |  |  |
| 3.6 | TLS/SSL Keys/Certs |  |  |
| 3.7 | Encryption Keys/Certs |  |  |
| 3.8 | Code Updates/upgrades |  |  |
|  |  |  |  |
| 4. | Web Apps |  |  |
| 4.1 | Dashboard login sessions |  |  |
| 4.2 | Controller API - Web Server/credentials |  |  |
| 4.3 | Compute API - Web Server/credentials |  |  |
|  |  |  |  |
| 5 | Transient information |  |  |
| 5.1 | Execution Memory | Sensitive information/keys |  |
| 5.2 | Ephemeral disks |  |  |
|  |  |  |  |
| 6 | Messaging |  |  |
| 6.1 | Messaging credentials |  |  |
| 6.2 | Messaging DB |  |  |
|  |  |  |  |
| 7 | Configuration |  |  |
| 7.1 | Nova Config |  |  |
| 7.2 | Scheduler Filters/rules |  |  |
|  |  |  |  |
| 8. | Logs |  |  |
| 8.1 | Log files |  |  |
| 8.2 | Logging API |  |  |
|  |  |  |  |
| 9. | Remote Access |  |  |
| 9.1 | Remote desktop |  |  |
| 9.2 | SSH |  |  |
|  |  |  |  |
| 10 | Nova DB |  |  |
| 10.1 | DB Credentials |  |  |
| 10.2 | DB persistence files |  |  |
|  |  |  |  |
| 11 | ObjectStore |  |  |
| 11.1 | ObjectStore persistence/files |  |  |
| 11.2 | ObjectStore Credentials |  |  |
| 11.3 | ObjectStore Data in transit |  |  |
|  |  |  |  |
| 12 | Monitoring/Metering  /Events/Alerts |  |  |
| 12.1 | M/M/E/A in transit |  |  |
| 12.2 | M/M/E/A Credentials |  |  |
| 12.3 | M/M/E/A Persistente |  |  |
|  |  |  |  |

**3.4 Security Profile of Nova (Existing Security Controls)**

Here we describe some model question to create a security profile for this application.

Ref: http://msdn.microsoft.com/en-us/library/ff648644.aspx

TBD

**4 Threat Identification (High Level)**

**4.1 Threat Agents**

|  |  |  |
| --- | --- | --- |
| ID | Name | Details |
| IA-U | Internet Attacker– Unauthorized |  |
| IA-A | Internet Attacker – Authorized |  |
| IN-I | Internal Attacker - Insider |  |

**4.2 Threat Identification and analysis**

Analysis result:

Following STRIDE Threat categorization: <http://en.wikipedia.org/wiki/STRIDE_(security)>

**TBD**

**4.3. Component based Threat Analysis**

**4.3.1 Identified Major Components / Proxies:**

**Components layering two trust boundaries and performing critical operations are interesting for this analysis**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Functionality** |
| 1 | *Client* | |
| 1.1 | Nova-client |  |
| 1.2 | Nova Dashboard |  |
| 1.3 | Remote Desktop proxy |  |
| 1.4 | Nova-manage |  |
|  |  | |
| 2 | *API/Web servers* | |
| 2.1 | Controller - Nova API |  |
| 2.2 | Controller - EC2 API |  |
| 2.3 | Compute APIs |  |
| 2.4 | S3 API |  |
| 2.5 | Compute/Conductor API |  |
|  |  |  |
| 3 | *Persistence/Storage* | |
| 3.1 | DBMS |  |
| 3.2 | ORM |  |
|  |  |  |
| 4 | *Infrastructure/System* | |
| 4.1 | Web server / WSGI server |  |
| 4.2 | SSL server/termination |  |
| 4.3 | Controller system/OS |  |
| 4.4 | Compute’s OS |  |
| 4.5 | Monitoring/Metering mechanism |  |
| 4.6 | Nova.conf |  |
|  |  |  |
| 5 | *External* |  |
| 5.1 | Keystone- External Auth system |  |
| 5.2 | Swift |  |
| 5.3 | Neutron |  |
| 5.4 | Cinder |  |
| 5.5 | Glance |  |
| 5.6 | OSLO |  |
| 5.7 | Orchestration (Heat) |  |
| 5.8 | Hypervisor |  |

**Result:**

**TBD**

**4.3.2 Use Cases:**

Common Nova usage scenarios to consider:

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Details** |
| 1 | User creates a VM Instance | Thru Dashboard or Nova Client, via EC2 or OpenStack API |
| 2 | User add Security Groups / Rules to Instance | Thru Dashboard or Nova Client, via EC2 or OpenStack API |
| 3 | Admin (Live) Migrates a VM instance | Thru Dashboard or Nova Client, via EC2 or OpenStack API |
| 4 | Admins Terminates Host | Thru Dashboard or Nova Client, via EC2 or OpenStack API |
| 5 | Heat Module request a VM Instance creation/termination thru API | Via Nova API |
| 6 | Admin instantiates Nova to use specific virtualization plugin and Hypervisors (i.e. Libvirt/KVM, Libvirt/Xen, VMWare, etc) | Infrastructure definitions/TrippleO |
| 7 | Admin instantiates new Compute node | Infrastructure definitions/TrippleO |