MiCADO Security Enablers Implementation Reference Guide

There are 7 user-visible security functions implemented by WP7 within the scope of the MiCADO framework to enhance the overall security of the product. These provide services to the end user and developers that implement industry-standard best practices, while minimizing the need of user-supplied configuration. The security user experience is crucial in terms of the actual security level of MiCADO deployments, as end users will only use functions, that are easily enabled and configured.

Besides the features that provide direct user value, there are also 3 support functions that aid to implement internal security and maintainability of the MiCADO platform. It is a design decision in the COLA project to build every aspect of the software in a pluggable architecture. By the nature of security enables, they tightly integrate with the actual implementations of the various MiCADO functions. The high-level support functions have been implemented in a standalone component, that is responsible for executing security workflows while hiding the actual implementation from other MiCADO components. This way the security enablers may be changed or replaced without touching other pieces of the framework relying on implementation details.

The deliverables “D7.2 MiCADO security architecture specification”, “D7.3 Design of application level security classification formats and principles” and “D7.4 Security policy formats specification” describe the designed security enablers in detail, also providing an open specification for implementing the various security functions.

Out of the identified and designed security features, the enablers that provide the most efficient security functions – based on feedback from WP8 use-case partners – have been selected for implementation.

The following user-visible functions have been implemented within the scope of the COLA project:

- Verifying OS images that are running the application containers
- Storing cloud credentials in an encrypted way
- Packet filtering firewall for the MiCADO master node
- Encryption, authentication and authorization on web access to the MiCADO master node
- Secure communication between MiCADO master and worker nodes
- Application-level firewalling for MiCADO applications
- Application-related secret handling and distribution

The above functions are addressing security problems, that are present in all current infrastructure, container and application orchestration solutions and have not been solved in any of the mainstream cloud solutions. While some cloud providers supply tools for workarounds, these all present an administrative overhead and are often neglected by cloud administrators.

Verifying OS images that are running the application containers

This security enabler aims to provide means for administrators to verify if their deployed operating system image has been modified by 3rd parties. As administrators are handing off responsibility to cloud providers for running their images, MiCADO would like to provide a way to enhance this trust by implementing detective controls. The current implementation is included in version 0.7.2 but needs to be simplified to be of use to end-users.

Storing cloud credentials in an encrypted way

This security enabler aims to provide a means for administrators to store sensitive cloud credential information an encrypted way and audit access to the credentials. While the information to access the credentials needs to be available for MiCADO to drive automatic workflow execution, "data-at-rest" encryption and audit logging increase the security of credentials storage and provide detective controls for unauthorized access. The current implementation is based on the industry-standard open source credential store Hashicorp Vault and included in version 0.7.4. Integration with other MiCADO component needs to be improved.

Packet filtering firewall for the MiCADO master node

This security enabler aims to provide a means for administrators to restrict network-level access to the MiCADO master node in a cloud-agnostic way. This area is painfully neglected in most mainstream cloud solutions and while there are proprietary tools on each cloud platform, they cannot be maintained in a uniform way. This enabler provides both preventive and detective controls by implementing network access control and audit logging for incoming connections. The current implementation is based on Netfilter and included in version 0.6.0.

Encryption, authentication and authorization on web access to the MiCADO master node

This security enabler aims to provide a means for administrators to ensure that all web management connections to the master node are properly encrypted in transit and make sure that only authorized users have access to MiCADO resources, while providing easy access for machine accounts to perform automated tasks. This enabler is implemented as proprietary tools in the portfolio of some cloud providers, sometimes also dubbed as "Identity-aware proxy", but no cloud-agnostic solution is included in mainstream cloud solutions. The current implementation is based on Balasys Zorp and included in version 0.6.0.

Secure communication between MiCADO master and worker nodes

This security enabler aims to provide a means for administrators to ensure that all network traffic between the master and worker nodes is encrypted in-transit and access to sensitive master node resources is restricted. This enabler makes it possible to separate management and production functions into separate runtime environments, while providing the same level of security for transit connections as if they were running in an environment controlled entirely by the end user. This removes the need of trusting the cloud provider not to listen in on internal connections. The current implementation is based in the StrongSWAN IPSEC daemon and included in version 0.7.2.

Application-level firewalling for MiCADO applications

This security enabler aims to provide a means for administrators to provide application-level filtering and enforcement capabilities for exposed MiCADO applications based on the NetworkSecurityPolicy descriptions in the ADT. This enabler brings in the networking and network security aspects of the application integrated into the same application descriptor while providing a cloud-agnostic implementation. The current implementation is based on Balasys Zorp and the Kubernetes Ingress Controller framework and is included in version 0.8.0.

Application-related secret handling and distribution

This security enabler aims to provide a means for administrators to ensure that all application-related sensitive information is stored and transmitted in a secure way when being distributed within the MiCADO framework. Current mainstream solutions do not use encryption by default. By implementing encryption not only application-related information, but also systems secrets are protected against eavesdropping and modification. The current implementation is based on etcd and included in version 0.7.4.

As stated before, several support functions have also been implemented to facilitate pluggability and overall systems security:

- Security workflow director and common interface to security components
- Safely store and verify user accounts for accessing the MiCADO framework
- Providing cryptographic functions to other components within MiCADO

These functions are highly technical, and their specification can be found in deliverable "D7.3 Design of application level security classification formats and principles".

The current status and technical information of the enablers is described in Appendix … FIXME

User-visible Features:

- Verifying OS images that are running the application containers
  - Technical name: Image Integrity Verifier
  - Rationale: ensure that the image running in the cloud infrastructure has not been modified by 3rd parties
  - Current status: integrated and running in MiCADO, but no mechanism to call the verification method, only works on Intel CPUs
  - Initial availability: v0.7.2
  - 3rd Party: no
  - Improvement plans:
    - provide an automated mechanism for image or file verification of the underlying operating system (WP6 assistance needed)
    - simplify verification
    - make implementation CPU-agnostic
    - research whether cloud providers expose a way of retrieving image hashes, VM orchestrator should provide this function to the IIVR
    - integrate the verifier with the VM orchestrator
    - provide CLI tool (micadoctl) to add new hashes to the image database
  - GitHub repository: https://github.com/micado-scale/component-iivr
  - Docker Hub repository: https://hub.docker.com/r/micado/iivr/
  - Docker Hub autobuild: yes
  - Travis: no
  - Configuration: none
  - Relation to other components:
    - Deployment via ansible-micado, affected files:

        - https://github.com/micado-scale/ansible-micado/blob/master/roles/build-micado-master/tasks/pull-docker-images.yml
        - https://github.com/micado-scale/ansible-micado/blob/master/roles/start-micado-master/templates/micado/micado-manifest.yml.j2
    - CLI configuration via security-policy-manager:
        - https://github.com/micado-scale/component-security-policy-manager/blob/master/bin/micadoctl

- Storing cloud credentials in an encrypted way
    - Technical name: Credential Store
    - Rationale: ensure that cloud credentials are not accessible in plain text when 'at rest' and access to the credentials are audited
    - Current status: integrated and running in MiCADO, but cloud orchestrator still uses plain text to cache the credentials
    - Initial availability: v0.7.2, reimplemented in v0.7.4
    - 3rd Party: yes, Hashicorp Vault
    - Improvement plans:
        - extend VM orchestrator to retrieve the credentials on a per-use bases and discard safely after use (WP6 assistance needed)
        - provide CLI tool to change credentials after deployment
    - GitHub repository: https://github.com/hashicorp/vault
    - Docker Hub repository: https://hub.docker.com/_/vault
    - Docker Hub autobuild: yes
    - Travis: CircleCI
    - Configuration: https://github.com/micado-scale/ansible-micado/blob/master/roles/build-micado-master/files/vault/vault.hcl
    - Relation to other components:
        - Deployment via ansible-micado, affected files:
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/build-micado-master/tasks/pull-docker-images.yml
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/start-micado-master/templates/micado/micado-manifest.yml.j2
        - The API functions are called by the Security Policy Manager component that hides the actual implementation of storing the cloud credentials, details in https://github.com/micado-scale/component-security-policy-manager/blob/master/app/secrets.py

- Packet filtering firewall for the MiCADO master node
    - Technical name: iptables
    - Rationale: ensure that only those services are exposed to the outside world that are needed for the operation of MiCADO
    - Current status: integrated and running
    - Initial availability: v0.6.0
    - 3rd Party: yes, iptables
    - Improvement plans:

        - limit the accessibility of MiCADO management ports to a set of pre-defined IP addresses / subnets
    - GitHub repository: https://github.com/micado-scale/ansible-micado
    - Docker Hub repository: N/A
    - Docker Hub autobuild: N/A
    - Travis: N/A
    - Configuration:
        - https://github.com/micado-scale/ansible-micado/blob/master/micado-master.yml#L98
        - SystemD service files: https://github.com/micado-scale/ansible-micado/tree/master/roles/build-micado-master/files/iptables
        - config templates: https://github.com/micado-scale/ansible-micado/tree/master/roles/start-micado-master/templates/iptables
    - Relation to other components:
        - Deployment via ansible-micado, see above
        - The set of rules needs to be re-evaluated every time a component is upgraded or replaced in MiCADO to ensure that their operation is not affected by firewalling

- Encryption, authentication and authorization on web access to the MiCADO master node
    - Technical name: Zorp Master Node
    - Rationale: ensure that all web management connections to the master node are properly encrypted in transit and make sure that only authorized users have access to MiCADO resources. Supports form-based authentication for users and HTTP Basic authentication for machine accounts.
    - Current status: integrated and running in MiCADO
    - Initial availability: v0.6.0
    - 3rd Party: yes, Zorp GPL
    - Improvement plans:
        - Certificate generation via letsencrypt
    - GitHub repository: https://github.com/Balasys/zorp
    - Docker Hub repository: https://hub.docker.com/r/micado/zorpgpl
    - Docker Hub autobuild: no
    - Travis: https://travis-ci.org/Balasys/zorp
    - Configuration:
        - Web listening port configuration:
            - https://github.com/micado-scale/ansible-micado/blob/develop/micado-master.yml#L73
            - https://github.com/micado-scale/ansible-micado/blob/develop/roles/start-micado-master/templates/micado/micado-manifest.yml.j2#L923
        - Certificate generation:
            - https://github.com/micado-scale/ansible-micado/blob/master/sample-credentials-micado.yml
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/build-micado-master/files/zorp/zorp-entrypoint.sh

        - Authentication web form:
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/build-micado-master/files/zorp/authform.html
        - Policy template:
            - https://github.com/micado-scale/ansible-micado/blob/develop/roles/start-micado-master/templates/zorp/policy.py.j2
    - Relation to other components:
        - Deployment via ansible-micado, affected files:
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/build-micado-master/tasks/pull-docker-images.yml
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/start-micado-master/templates/micado/micado-manifest.yml.j2
        - Performs authorization by checking users against the Credential Manager and fetches group membership as well

- Secure communication between MiCADO master and worker nodes
    - Technical name: IPSEC
    - Rationale: ensure that all network traffic between the MiCADO master and worker are encrypted authenticated.
    - Current status: integrated and running in MiCADO
    - Initial availability: 0.7.2
    - 3rd Party: yes, StrongSwan
    - Improvement plans:
        - Encrypted overlay network to protect worker-worker communication
    - GitHub repository: https://git.strongswan.org/?p=strongswan.git;a=summary
    - Docker Hub repository: N/A
    - Docker Hub autobuild: N/A
    - Travis: https://travis-ci.org/strongswan/strongswan/
    - Configuration:
        - Master node certificate generation:
            - https://github.com/micado-scale/ansible-micado/blob/develop/roles/start-micado-master/tasks/micado-start.yml
        - Master node configuration:
            - https://github.com/micado-scale/ansible-micado/blob/develop/roles/start-micado-master/templates/ipsec/ipsec.conf
        - Installation on master node:
            - https://github.com/micado-scale/ansible-micado/blob/develop/roles/build-micado-master/tasks/other-install.yml
        - Worker node certificate generation:
            - https://github.com/micado-scale/ansible-micado/blob/develop/roles/start-micado-master/templates/worker_node/cloud_init_worker.yaml.j2#L69
        - Worker node configuration:
            - https://github.com/micado-scale/ansible-micado/blob/develop/roles/start-micado-master/templates/worker_node/cloud_init_worker.yaml.j2#L45
        - Installation on worker node:

        - https://github.com/micado-scale/ansible-micado/blob/develop/roles/start-micado-master/templates/worker_node/cloud_init_worker.yaml.j2#L117
  - Relation to other components:
    - Deployment via ansible-micado, see above
    - Certificate generation is done via the Security Policy Manager
    - If not operational, worker node cannot join the k8s cluster and/or provide metrics to Prometheus

- Application-level firewalling for MiCADO applications
  - Technical name: Zorp Ingress Director
  - Rationale: provide application-level filtering and enforcement capabilities for exposed MiCADO applications based on the NetworkSecurityPolicy descriptions in the ADT
  - Current status: under development
  - Initial availability: v0.7.4 / v0.7.5
  - 3rd Party: yes, Zorp Ingress Director
  - GitHub repository: https://github.com/Balasys/zorp-ingress-controller
  - Docker Hub repository: https://hub.docker.com/r/balasys/zorp-ingress
  - Docker Hub autobuild: via Travis
  - Travis: https://travis-ci.org/Balasys/zorp-ingress-controller
  - Configuration:
    - Worker node configuration:
      - https://github.com/micado-scale/ansible-micado/blob/develop/roles/start-micado-master/templates/worker_node/cloud_init_worker.yaml.j2#L45
    - TOSCA Submitter adapor:
      - https://github.com/micado-scale/component_submitter/pull/186/commits/55a2505737d7b53525af9b50e55901de9b6afcac#diff-716a6c6ec30ef95b7f0423a09c495c88
  - Relation to other components:
    - Part of the TOSCA Submitter as a Security Enforcer Adaptor
    - Part of the TOSCA Submitter as part of the K8s Adaptor
    - Applies configuration through the K8s API
    - Reads configuration from the K8s API
    - Configuration is generated based on the TOSCA descriptor available at: https://github.com/micado-scale/tosca/

- Application-related secret handling and distribution
  - Technical name: K8s Secret
  - Rationale: provide a way for TOSCA users to distribute secrets in a secure way to only affected application containers
  - Current status: integrated and running in MiCADO, no mechanism is provided to read the secrets on the application container side
  - Initial availability: v0.7.4
  - 3rd Party: yes, k8s Secret API
  - Improvement plans:

        - provide a mechanism to read the secrets on the application container side (WP6 assistance required)
        - avoid storing the secret in plain text in the ADT (symmetric encryption?)
    - GitHub repository:
    - Docker Hub repository:
    - Docker Hub autobuild:
    - Travis:
    - Configuration:
        - TOSCA Submitter adaptor:
            - https://github.com/micado-scale/component_submitter/pull/186/commits/55a2505737d7b53525af9b50e55901de9b6afcac#diff-716a6c6ec30ef95b7f0423a09c495c88
    - Relation to other components:
        - Part of the TOSCA Submitter as a Security Enforcer Adaptor
        - Stores secrets within the K8s Secret API
        - Configuration is generated based on the TOSCA descriptor available at: https://github.com/micado-scale/tosca/

Support functions:

- Security workflow director and common interface to security components
    - Technical name: Security Policy Manager
    - Rationale: provide a common interface to all MiCADO components to access security-related components and functions and run security-related workflows
    - Current status: integrated and running in MiCADO, used for several infrastructure-related functions
    - Initial availability: v0.7.2
    - 3rd Party: no
    - Improvement plans:
        - upgrade library for interacting with Hashicorp Vault (https://github.com/hvac/hvac) a version that is compatible with the latest Vault API
        - provide and interface to Credential Manager
    - GitHub repository: https://github.com/micado-scale/component-security-policy-manager
    - Docker Hub repository: https://hub.docker.com/r/micado/security-policy-manager
    - Docker Hub autobuild: yes
    - Travis: https://travis-ci.org/micado-scale/component-security-policy-manager
    - Configuration: N/A
    - Relation to other components:
        - Deployment via ansible-micado, affected files:
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/build-micado-master/tasks/pull-docker-images.yml
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/start-micado-master/templates/micado/micado-manifest.yml.j2
        - Provides implementation for storing cloud credentials

        - Provides implementation for storing application secrets
        - Provides a PKI to be used by worker-master secure communication
        - Provides an interface to CryptoEngine
        - Provides an interface to Image Integrity Verifier

- Safely store and verify user accounts for accessing the MiCADO framework
    - Technical name: Credential Manager
    - Rationale: provide a lightweight REST-based interface for storing and verifying user accounts to use with the MiCADO web interface
    - Current status: integrated and running in MiCADO, used by Zorp
    - Initial availability: v0.6.0
    - 3rd Party: no
    - Improvement plans:
        - incorporate the flask-users library for simplification of the implementation
        - support multiple roles besides the currently supported user/admin distinction
        - avoid using plain-text credentials for initial provisioning (use an API call from the ansible playbook to add credentials via the API without saving them to file)
        - run automated tests on Travis
    - GitHub repository: https://github.com/micado-scale/component-credential-manager
    - Docker Hub repository: https://hub.docker.com/r/micado/credential-manager
    - Docker Hub autobuild: yes
    - Travis: no
    - Configuration: https://github.com/micado-scale/ansible-micado/blob/master/sample-credentials-micado.yml
    - Relation to other components:
        - Deployment via ansible-micado, affected files:
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/build-micado-master/tasks/pull-docker-images.yml
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/start-micado-master/templates/micado/micado-manifest.yml.j2
        - The Security Policy Manager contains the micadoctl CLI tool, that can change users and passwords in Credential Manager
        - The Zorp component uses Credential Manager directly to authenticate and authorize web access to MiCADO. A direct connection is used to avoid DoS attacks on the Security Policy Manager

- Providing cryptographic functions to other components within MiCADO
    - Technical name: Crypto Engine
    - Rationale: provide abstraction for crypto functions so that the underlying implementation can be changed easily
    - Current status: integrated and running in MiCADO, but no other components use it
    - Initial availability: v0.7.2
    - 3rd Party: no
    - Improvement plans:

        - convert the component into a python library instead of the current REST-based application
        - identify MiCADO components that should use the CryptoEngine for cryptography, currently none identified
    - GitHub repository: https://github.com/micado-scale/component-crypto-engine
    - Docker Hub repository: https://hub.docker.com/r/micado/crypto-engine
    - Docker Hub autobuild: no
    - Travis: no
    - Configuration: none
    - Relation to other components:
        - Deployment via ansible-micado, affected files:
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/build-micado-master/tasks/pull-docker-images.yml
            - https://github.com/micado-scale/ansible-micado/blob/master/roles/start-micado-master/templates/micado/micado-manifest.yml.j2