Threat Model for MOCCA Component Environment

http://virolab.cyfronet.pl

Objective

This model has been created to asses potential vulnerabilities in MOCCA and to find a secure solution for integration of MOCCA and Shibboleth.

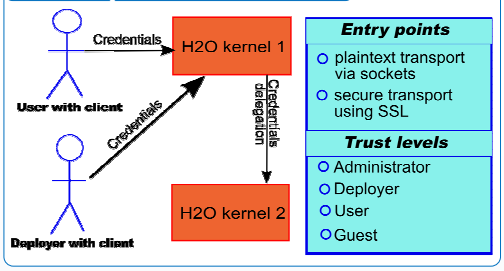
MOCCA Security Requirements

1. **Authentication** – identity verification, Single Sign On to access kernels distributed on various nodes in different locations.
2. **Authorization** – need to map user attributes to H2O role-based users to check permissions to e.g. deploy or run software.
3. **Credential delegation** – to enable component running in one container to deploy or run code in another container.
4. **Integrity** – it is crucial to protect both user data (input and output) as well as code from tempering or destruction.
5. **Confidentiality** – to protect data and code, that might contain classified information.
6. **Availability** – of the security infrastructure and protection H2O kernels from attack, spare nodes in case of DDoS attack

Protected Assets

Computer resources	Network resources	Privacy	Users data	Users software
Description	***Description***	***Description***	***Description***	***Description***
Software and hardware resources on the node running H2O kernels; Temporary unauthorized access to the node or exceeding predefined limits; Attacker should not be able to hold them for a long time.	Nodes' network connectivity; Malicious code with network access might allow to perform a Distributed Denial of Service attack; Potential for the attacker to compromise other hosts or send SPAM.	All personal data being transmitted in process of authentication and authorization; Very limited to user name in addition to real life name and user's email.	Most important assets in the system; Input data supplied by the user, might contain secrets like drugs formulas; Output data shouldn't be destroyed or stolen (might also be classified).	Another extremely important assets type; Contain all types of code being run by users in containers; Might contain unique secrets (e.g. algorithms) that should be protected from being stolen by the attacker.
Value for the attacker	***Value for the attacker***	***Value for the attacker***	***Value for the attacker***	***Value for the attacker***
Quite low - resources might be held just temporary for short time before detection; Even temporary access to computing power might be valuable (e.g. for passwords cracking).	Moderate - also temporary access to assets, even quite short, but well timed massive DDoS might be beneficial to the attacker (e.g. blocking competitors' systems).	Moderate here – in general highly valuable; In this case value is lower because there is not much personal data in the system.	High – access to assets is permanent (unless we track the perpetrator); Might contain confidential information (e.g. technological).	High - access to assets is permanent (unless we track the perpetrator); Might contain proprietary algorithms valuable for the competitor.
Cost of recovery	***Cost of recovery***	***Cost of recovery***	***Cost of recovery***	***Cost of recovery***
Low due to distributed nature of the system; even overloading a few nodes shouldn't immobilize the whole; High if massive attack block most important calculations.	Low – in most cases if staff reacts fast to complains from other (attacked) networks; High - if attacker causes serious damages to other systems using our network.	Low - unless we keep a very detailed personal information in the system.	Very high in the case of leakage of important data (fines specified by the contract); Lower in the case of data destruction (restore cost and fines for delays).	Very high if leaked code contains classified information; Moderate or high in other cases.

Sample Use Case



Threats to the System

STRIDE Classification Categories		
	Name	**Description**
S	Spoofing	pretending to be someone you aren't
T	Tampering	causing corruption of data
R	Repudiation	claiming that you did not agree on something, but in fact you have
I	Information disclosure	leakage of user's data or code
D	Denial of Service	system becomes unusable
E	Elevation of privilege	gaining bigger privileges

Threats to the system

Name	EP	Cat.	Description	Mitigation
Sniffing	Plain	STIE	Non-encrypted data could be easily eavesdropped	Do not use plaintext connection for production instalation
Man-in-the--middle	Plain SSL	STIE	Encrypted data eavesdropped	Make sure to use SSL with strong, valid certificates
Privilege escalation	Plain SSL	STIE	Gaining higher trust level than a user has	Check software for security bugs
Resources overstepping	Plain SSL	DE	Using more resources then a user is allowed	Check software for security bugs
Distributed Denial of Service	Plain SSL	D	Massive external attack on the network	Have spare nodes in another network
Social engineering	Plain SSL	STIDE	Extracting information from users not the system itself	Do not to trust information unless you know it is legitimate

EP - Entry Point

Attack Scenarios

1. **Plaintext Transmission** eavesdropping of data including credentials, using simple sniffer. Severity: Critical
2. **SSL eavesdropping** – Man-in-the-middle attack if certificate is not validated properly, leading to (1). Severity: Critical
3. **Privilege escalation** – attacker with low privileges (e.g. Guest) might get higher trust level (Admin) by deploying malicious code Severity: Moderate - Critical
4. **Resources limit overstepping** – privileged user might exceed permited resources by deploying malicious code, in worst case causing container crash. Severity: Low - Moderate
5. **Social engineering** – user (in worst case – Administrator) might be tricked into giving his/her credentials (e.g. phishing) Severity: Moderate - Critical

Future Work

- To provide easy credential delegation from Shibboleth to GSI-based system
- To combine our client library for Shibboleth SSO with GridShib library that allows propagating Shibboleth assertions as part of non-critical extensions to X.509 GSI certificate.

Authors

Jan Meizner (1), Maciej Malawski (1), Syed Naqvi (3), Marian Bubak (1,2)

(1) Institute of Computer Science AGH, al. Mickiewicza 30, 30-059 Krakow, Poland
(2) ACC CYFRONET AGH, Krakow, ul. Nawojki 11, 30-950 Krakow, Poland
(3) CETIC, Rue des Freres Wright 29/3, B-6041 Charleroi, Belgium

References

1. Maciej Malawski, Dawid Kurzyniec, Vaidy Sunderam: MOCCA - towards a distributed CCA framework for metacomputing. In Proceedings of the 10th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS2005) in conjunction with International Parallel and Distributed Processing Symposium (IPDPS 2005). IEEE, 2005.
2. Amit D. Lakhani, Erica Yang, Brian Matthews, Ian Johnson, Syed Naqvi, Gheorghe C. Silaghi. Threat Analysis and Attacks on XtreemOS: a Grid–enabled Operating System. Towards Next Generation Grids, Proceedings of the CoreGRID Symposium 2007, p. 53-62, Springer, 2007
3. http://shibboleth.internet2.edu/ 4. http://gridshib.globus.org/

ViroLab EU-IST-027446

Coordinator: Prof. P.M.A. Sloot
Universiteit van Amsterdam
www.virolab.org