I·C·S
The Institute for Cyber Security

C·SPECC
Center for Security and Privacy Enhanced Cloud Computing

Institute for Cyber Security: The Galahad Project

A Secure User Computing Environment for the Cloud

James Benson
Technology Research Analyst II

Bushra Zahed

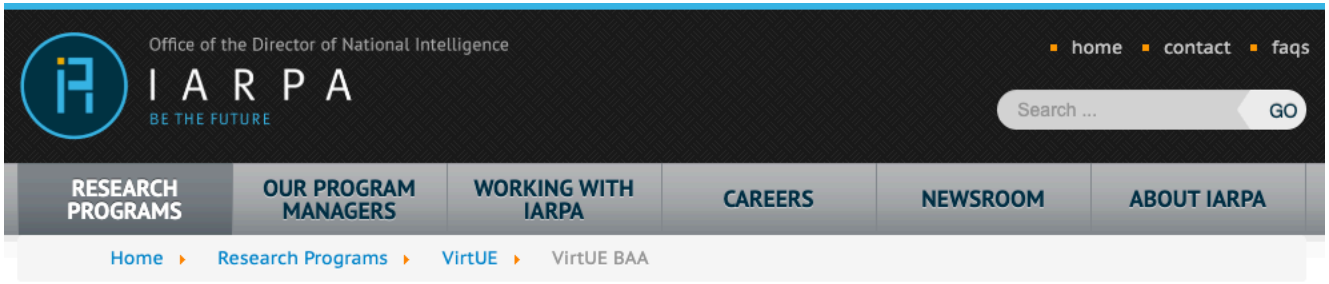
March 2020

James.Benson@utsa.edu
www.ics.utsa.edu
https://gitlab.com/utsa-ics/galahad

World-Leading Research with Real-World Impact!

UTSA Computer Science

Original IARPA Solicitation



Virtuous User Environment (VirtUE)

VirtUE seeks to leverage the federal government's impending migration to commercial cloudbased information Technology (IT) infrastructures and the current explosion of new virtualization and operating system (OS) concepts to create and demonstrate a more secure interactive user computing environment (UCE) than the government has had in the past or likely to have in the near future. Currently the government UCE is represented by a general purpose Windows desktop OS running multiple installed applications hosted on either a dedicated physical computer or on a shared virtualized platform. When a desktop OS is hosted on a shared virtualized platform, it is called a virtualized desktop interface or VDI.

In Phase 1, VirtUE seeks to deliver an interactive UCE designed from the outset to be a more secure, capable sensor and defender in the cloud environment than the current government UCE solution. To be acceptable to potential government consumers, the new UCE must still offer functionality and performance characteristics comparable to the current government UCE. Phase 1 performers shall create a UCE that mitigates the exploitation of legacy and cloud-based vulnerabilities and/or provides numerous logging and protection options for future external security logic to do so.

In Phase 2, performers shall take the technologies and/or concepts developed in Phase 1 and create novel external analytics and security controls that leverage them. The purpose of this analytics/control effort is to create dynamic detection and protection capabilities that make the VirtUE user environment more resistant

Solicitation Status: CLOSED

IARPA-BAA-16-12
Proposers' Day Date: July 19, 2016
BAA Release Date: October 18, 2016
BAA Question Period: October 18, 2016 - November 10, 2016
Proposal Due Date: December 12, 2016

Additional Information

Program Description

IARPA-BAA-16-12 Q&A (round one)

Proposers' Day Briefings

VirtUE Proposers' Day Briefing

AIS (presentation)

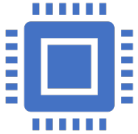
Columbia University (presentation)

Concurrent Technologies Corporation (presentation)

Galahad's History

Galahad was Star Lab's solution for Intelligence Advanced Research Project Activity (IARPA) VirtUE program - Virtuous User Environment (VirtUE) proposed in 2016.

Galahad is unique in that it was transitioned from Star Labs to ICS; We have open-sourced it. To create a turn-key opensource deployment tool to share it with others.



Utilizes:
Server, Desktop, Application, & Nested Virtualization

Overview of Galahad System

- Star Lab and Raytheon BBN Technologies proposes to conduct an applied research and development project to create and transition **Galahad.**

- Galahad – a revolutionary User Computer Environment (**UCE**) for the Amazon Cloud that is designed to be highly interactive while mitigating legacy and cloud specific threats.

- UCE – An environment for computation that a user interacts with to launch needed applications and access required informational resources containing one or more **Virtues** and a **presentation interface**.

Goals & Motivations

- Objective: Detection and mitigation of threats attempting to exploit, collect, and/or effect user computing environments (UCE) within public clouds

- Cloud service providers have not offered any game changing security solutions
  - Adversaries can leverage an arsenal of capabilities used to succeed
  - Providers cannot necessarily be trusted

- Current end-point security solutions and analytical approaches are not tuned for cloud environments

Security Advantages of Virt.

Better Forensics/Faster Recovery after an attack

Compromised machines can be cloned in compromised state for forensic analysis

After cloning, VM can be restored to known good state.

Control the default state of VM

Patches can be automatically incorporated

Failed upgrades can be reverted faster

Cost effective

Scale infrastructure devices accordingly instead of one size fits all.

Leveraging Virtualization to provide better security

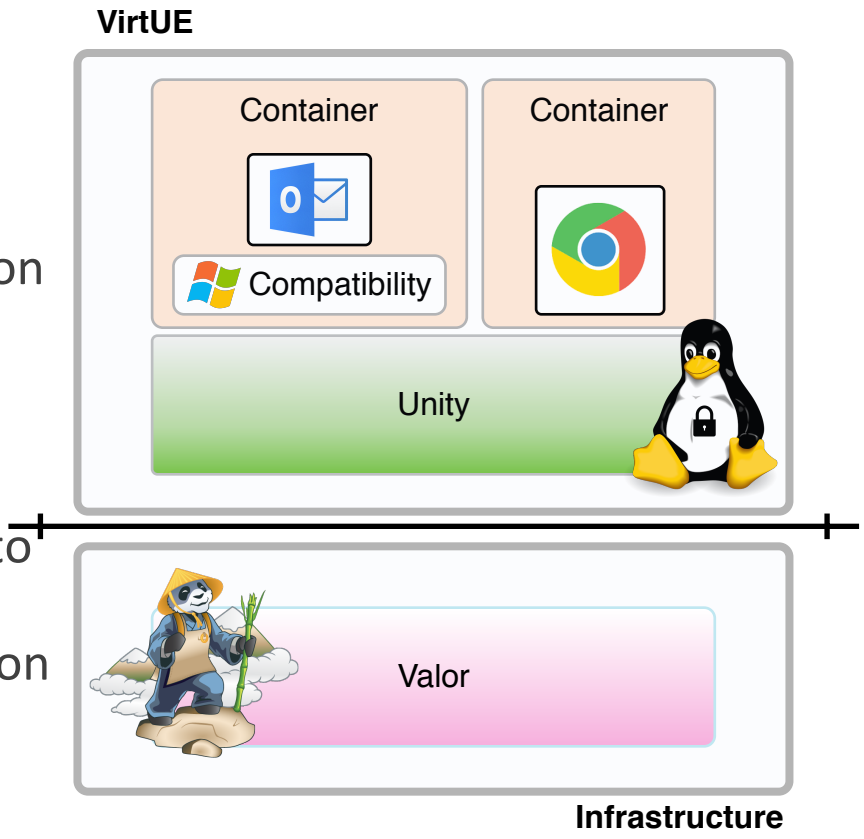
View interactions and contexts

Galahad Approach

- To combat threats in a public cloud, isolate, protect what is controlled, and maneuver
  - Do not attempt to establish trust (Rowhammer, Spectre, Meltdown,…)
  - Do not require special cloud services, e.g., dedicated servers
  - Impede the ability of adversaries to operate within AWS by making it more difficult to co-locate
  - Force adversaries to consume more resources thereby increasing the accuracy, rate, and speed with which threats maybe detected
  - Facilitate the creation of role-enabled security models
    - Universal Role: Email and intranet
    - Administrative: “Universal Role”, internet, word, excel
    - Programmer: “Universal Role”, internet, PowerShell, SSH, notepad++
    - Guest: Internet
  - Reduce attack surface area, hardened kernel, real-time sensing, limit resources.

Galahad VirtUE

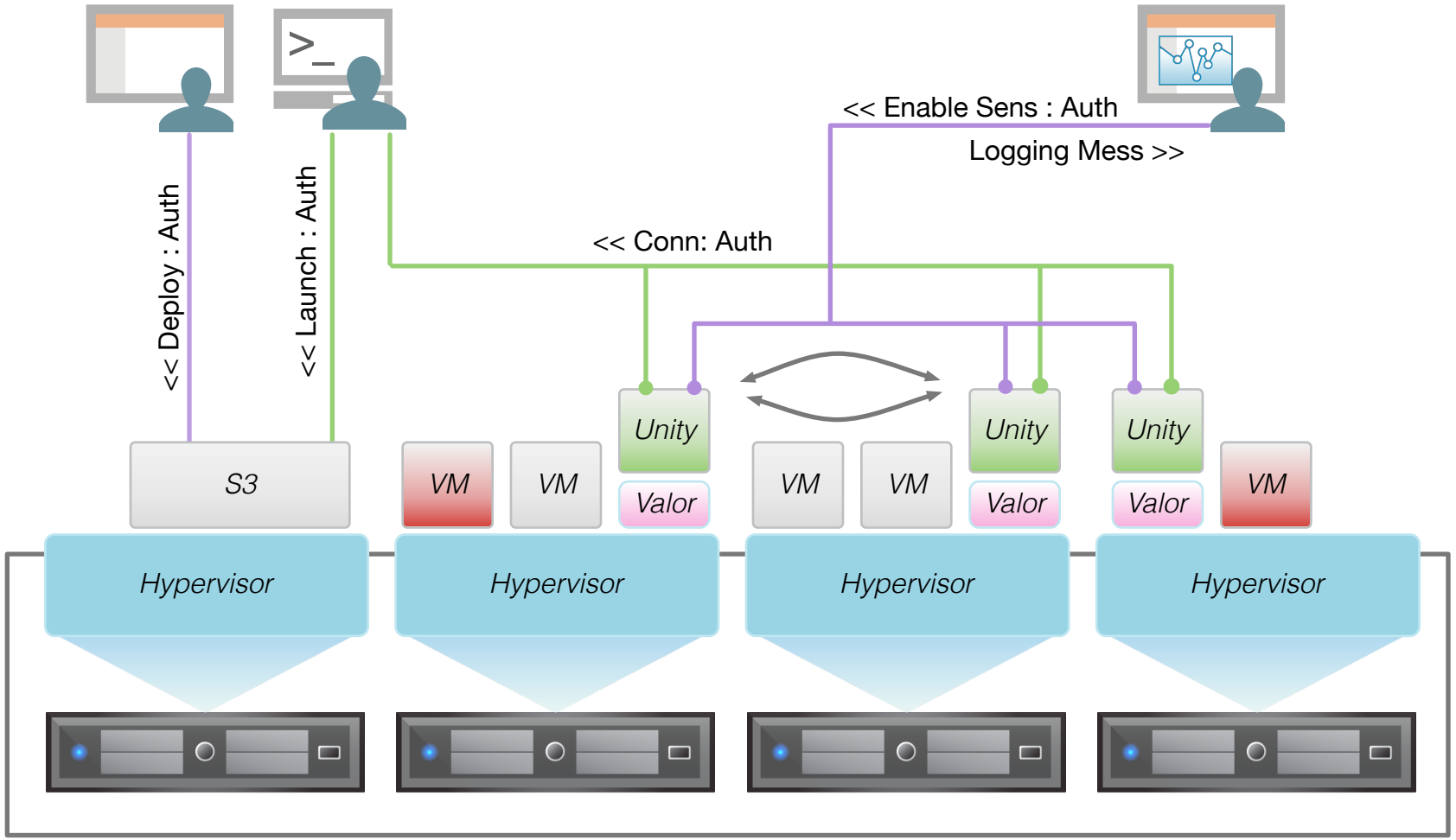
Containers for easy packaging and security configuration



A small, hardened, de-privileged Linux OS VM

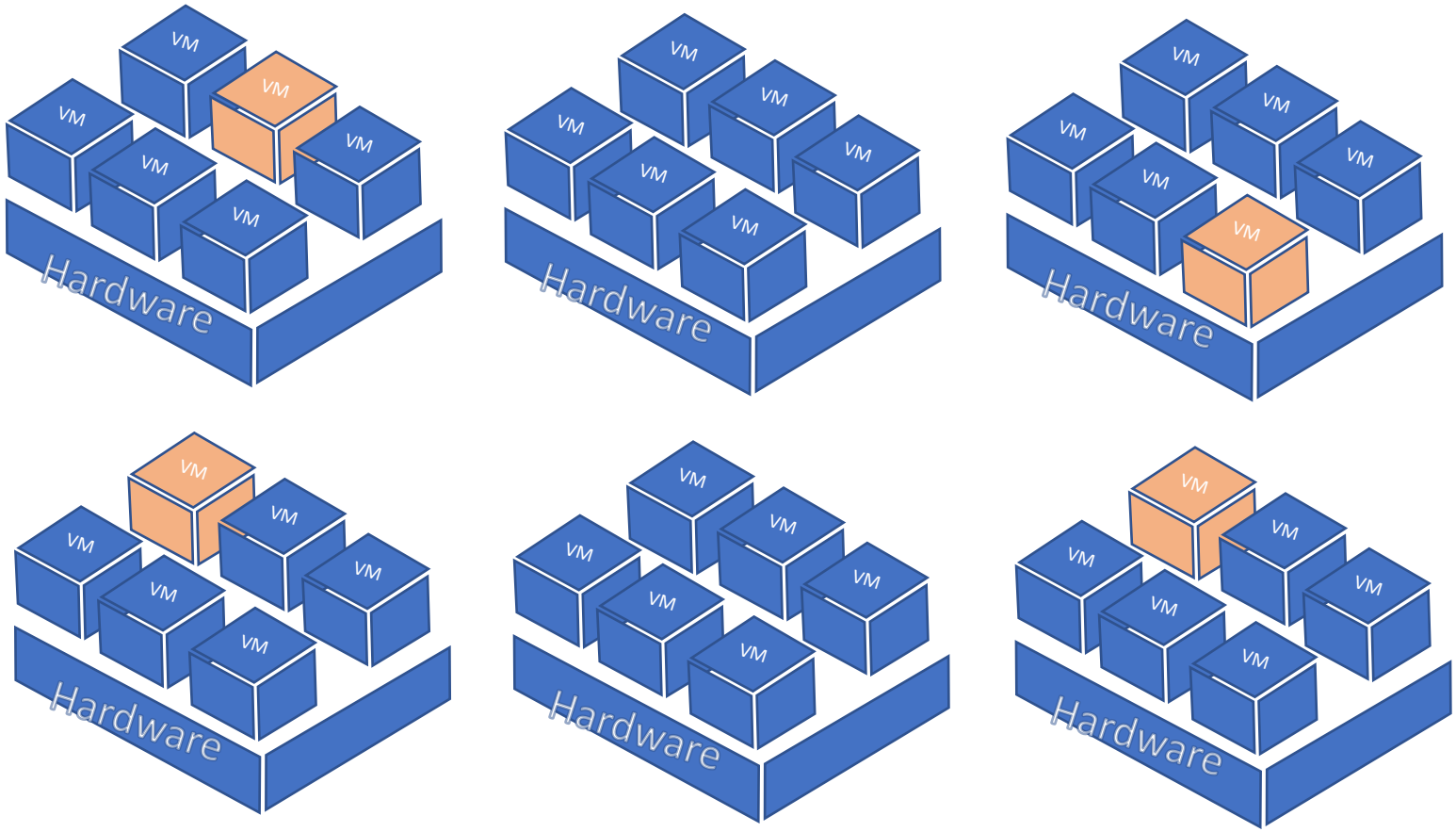
A nested hypervisor to facilitate regular, recurring live migration of Unity VMs inside AWS

Galahad Use Case

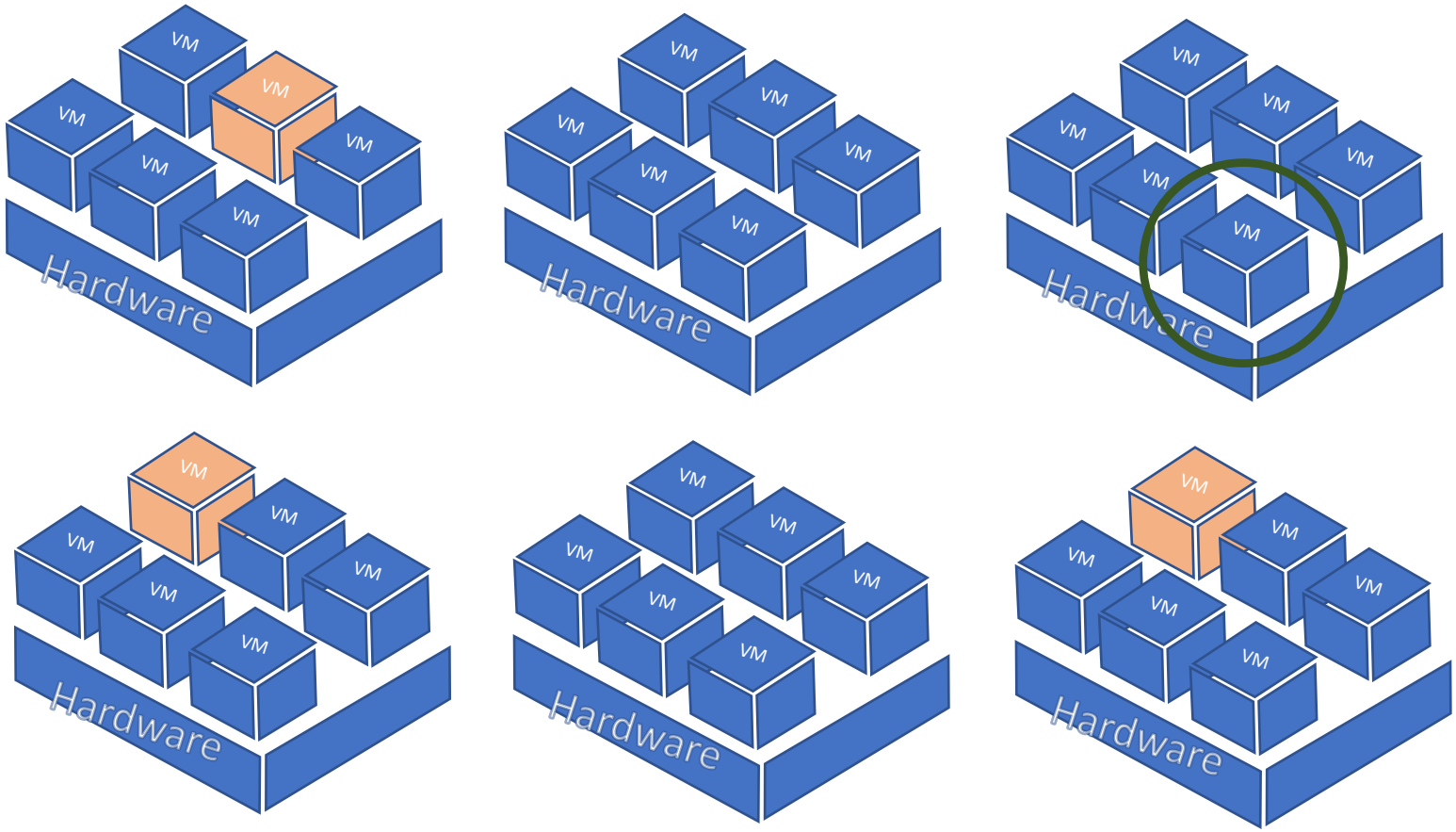


AWS Immutable Infrastructure

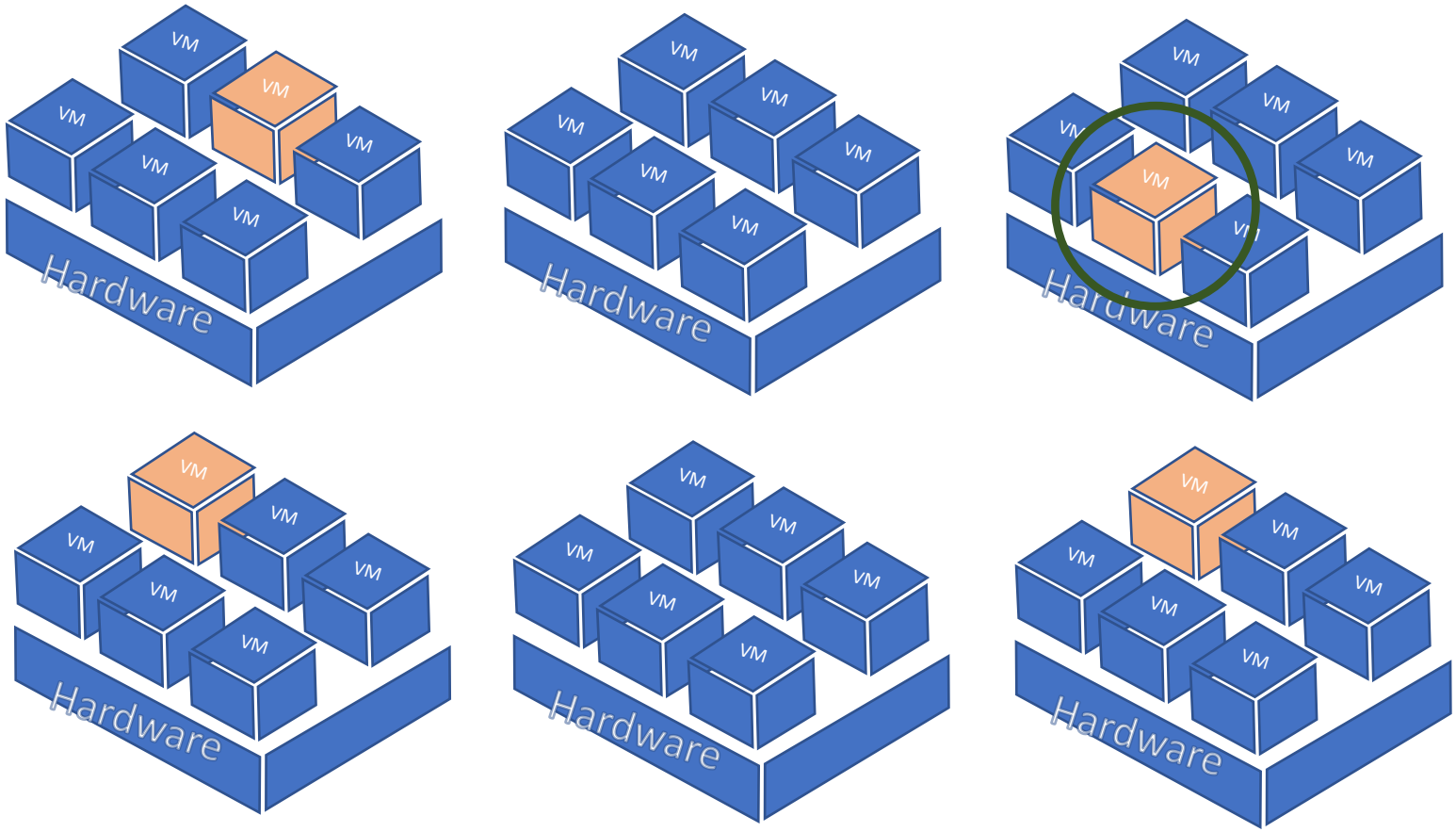
AWS Servers…



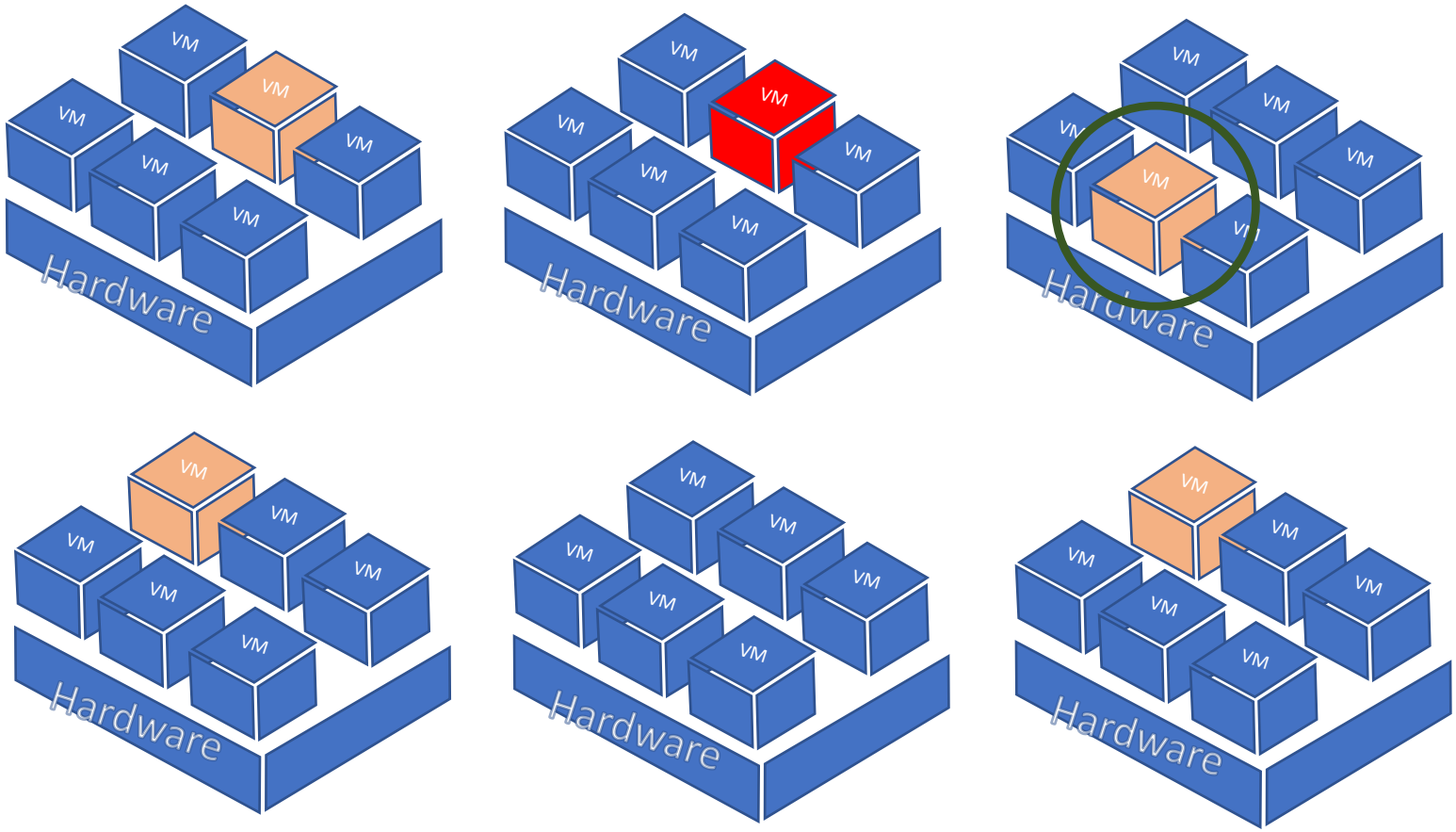
VM's shut down



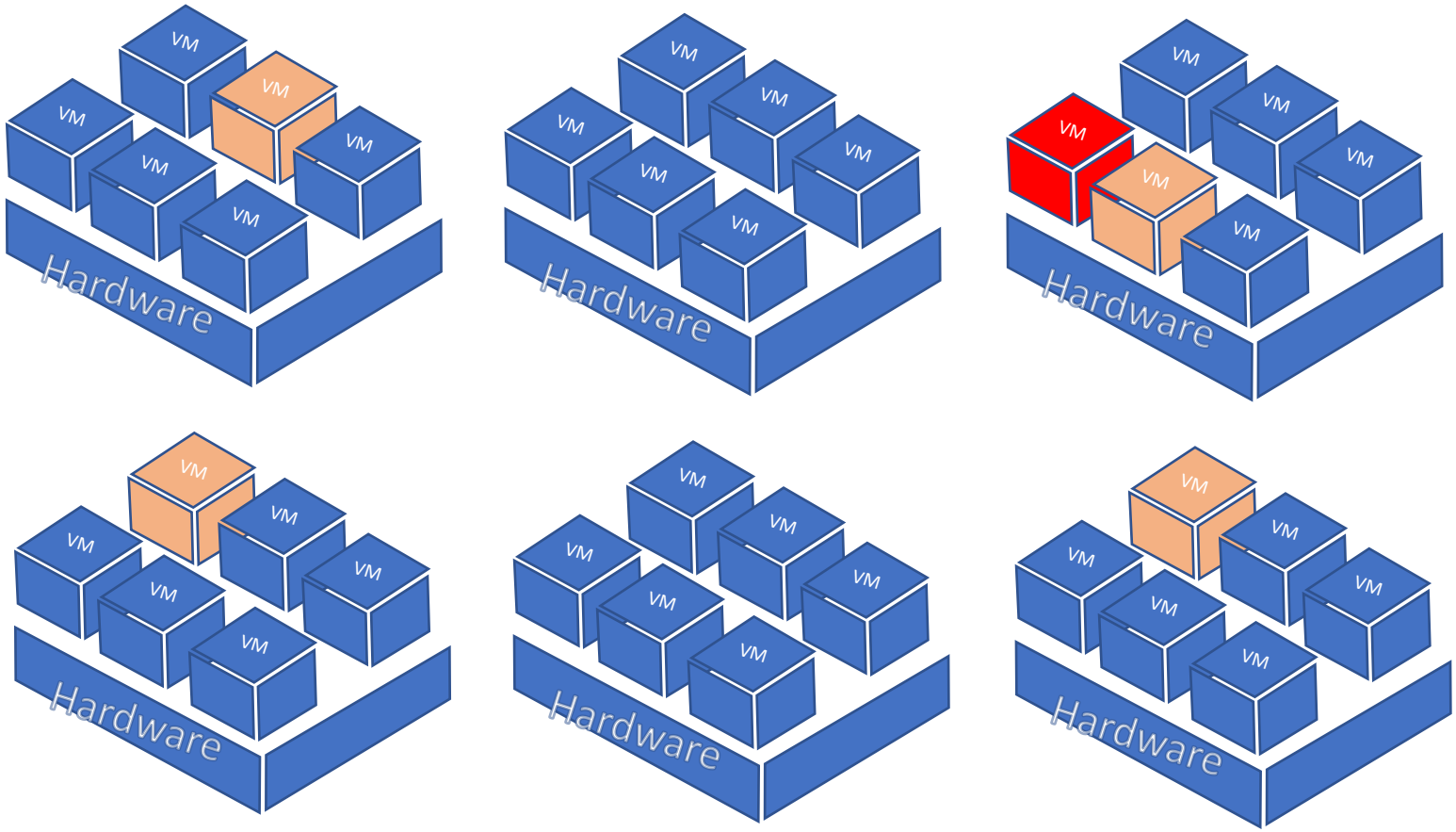
VM's turn on



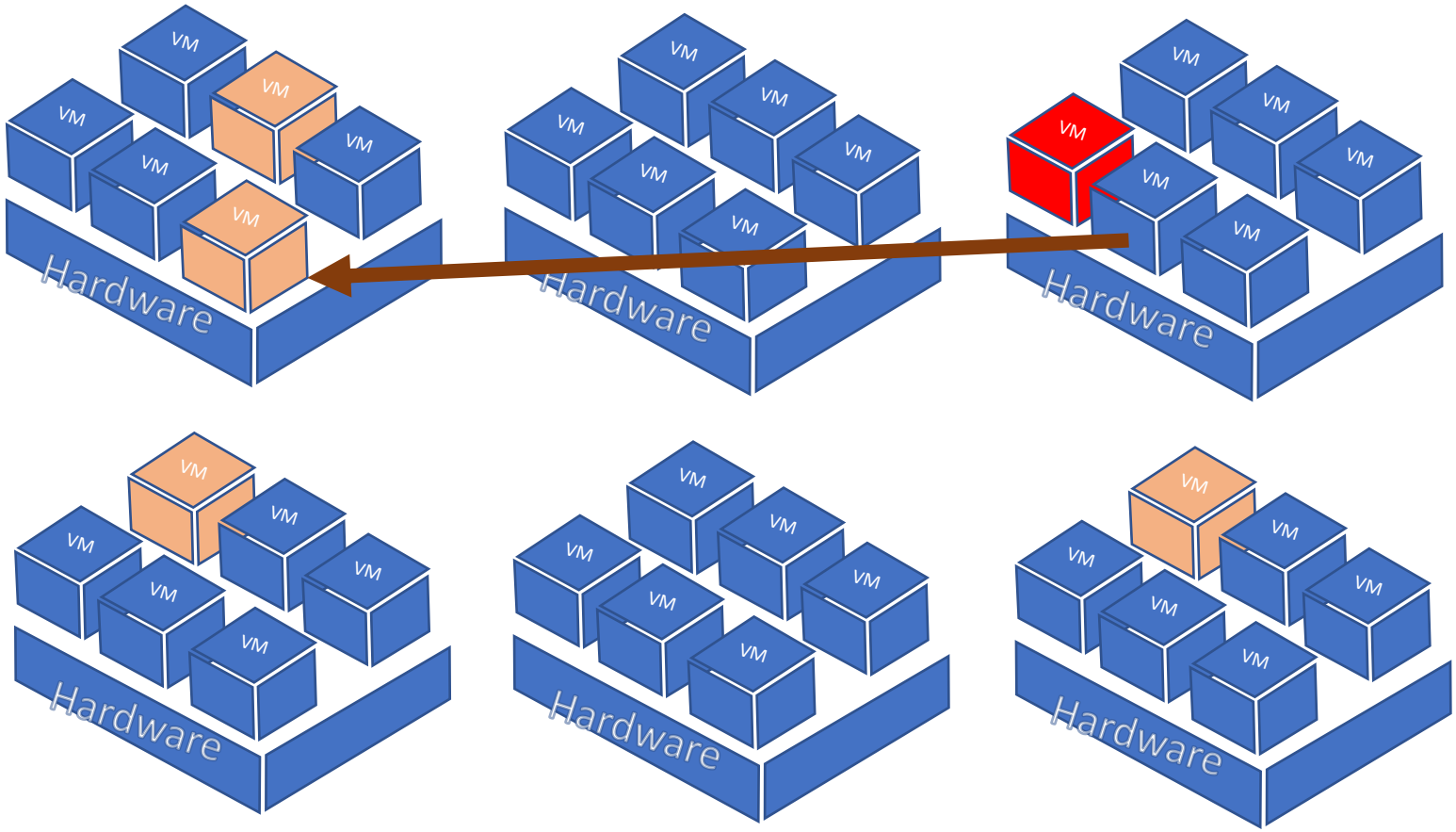
Bad actor, probing…



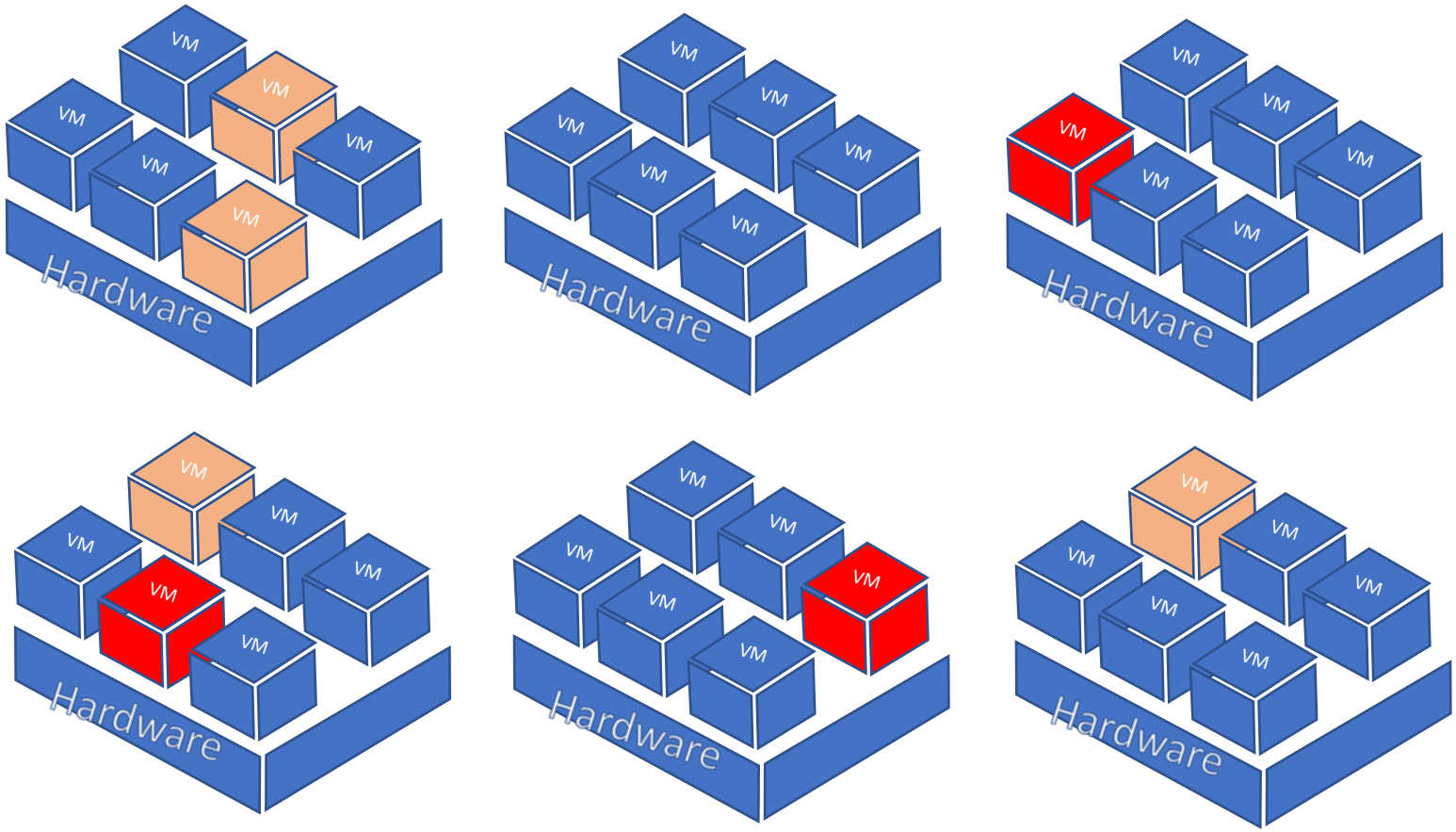
Found something?



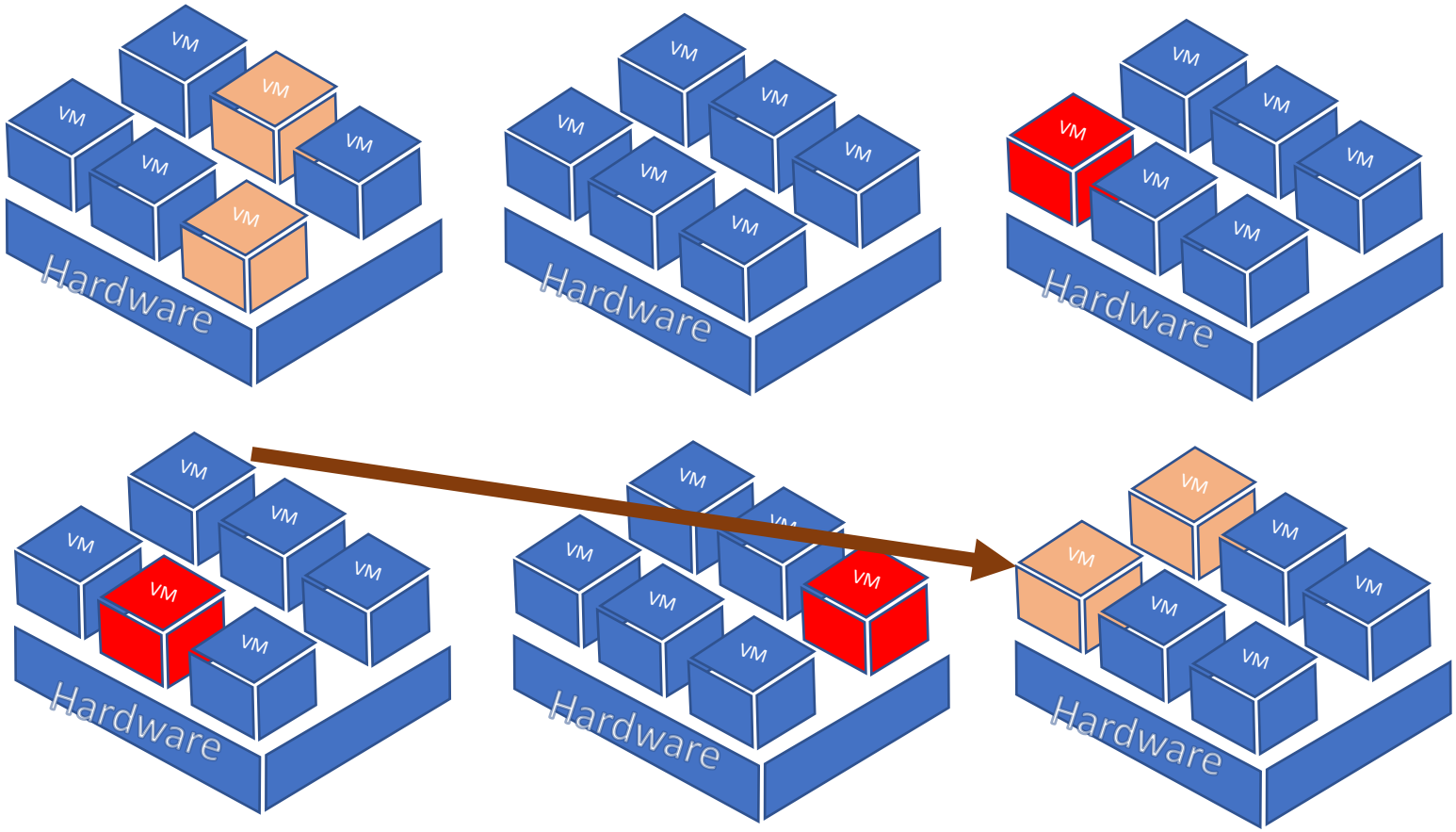
Attack detected.



Aggressive Attack



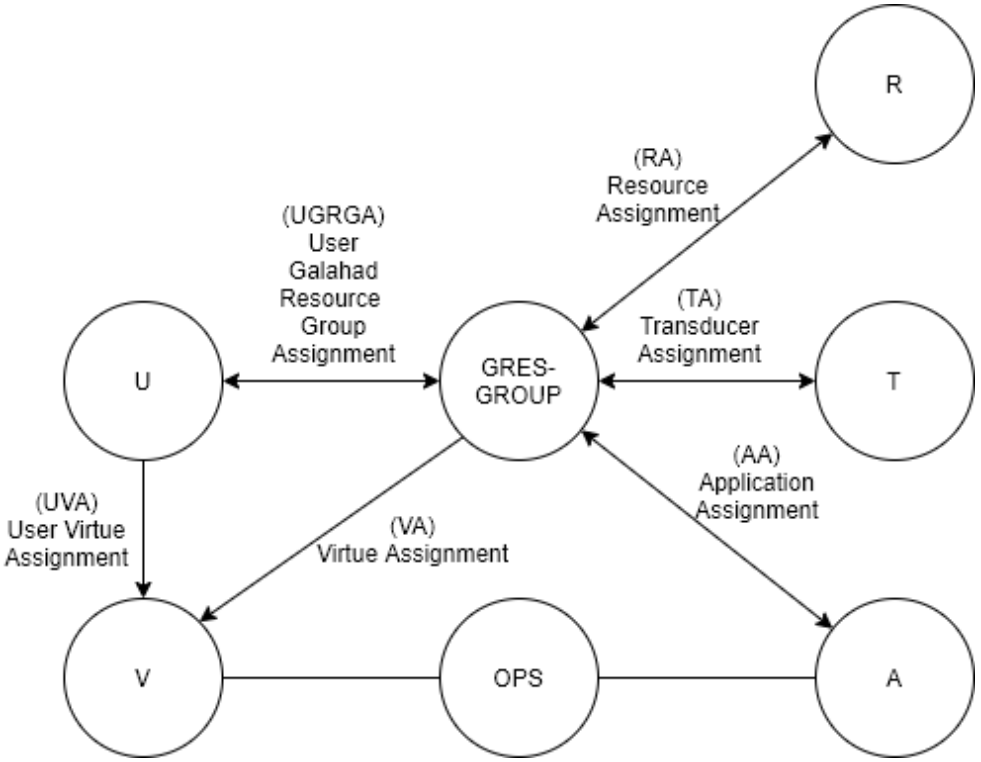
Aggressive Migration



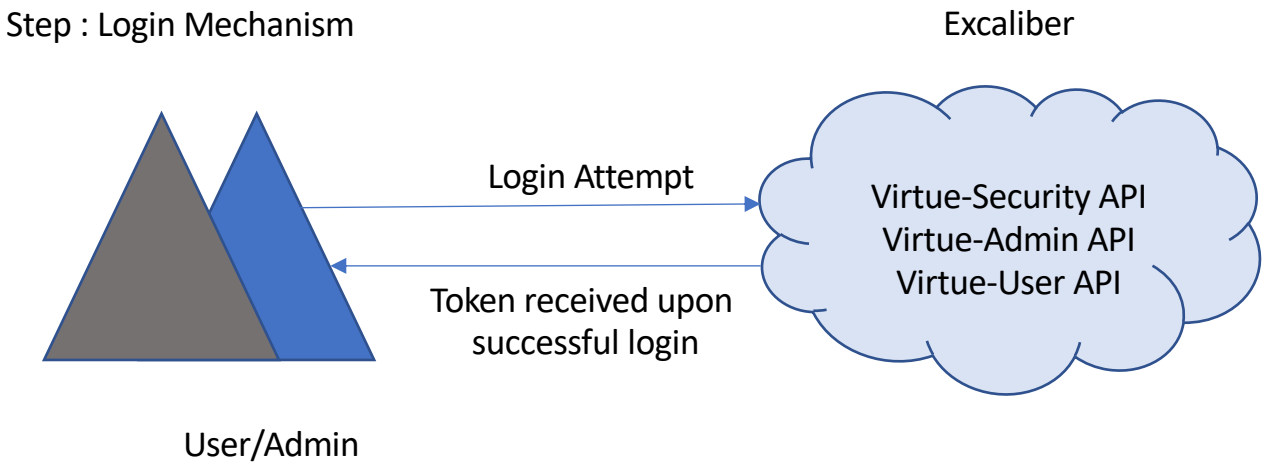
Existing System Framework

Definition

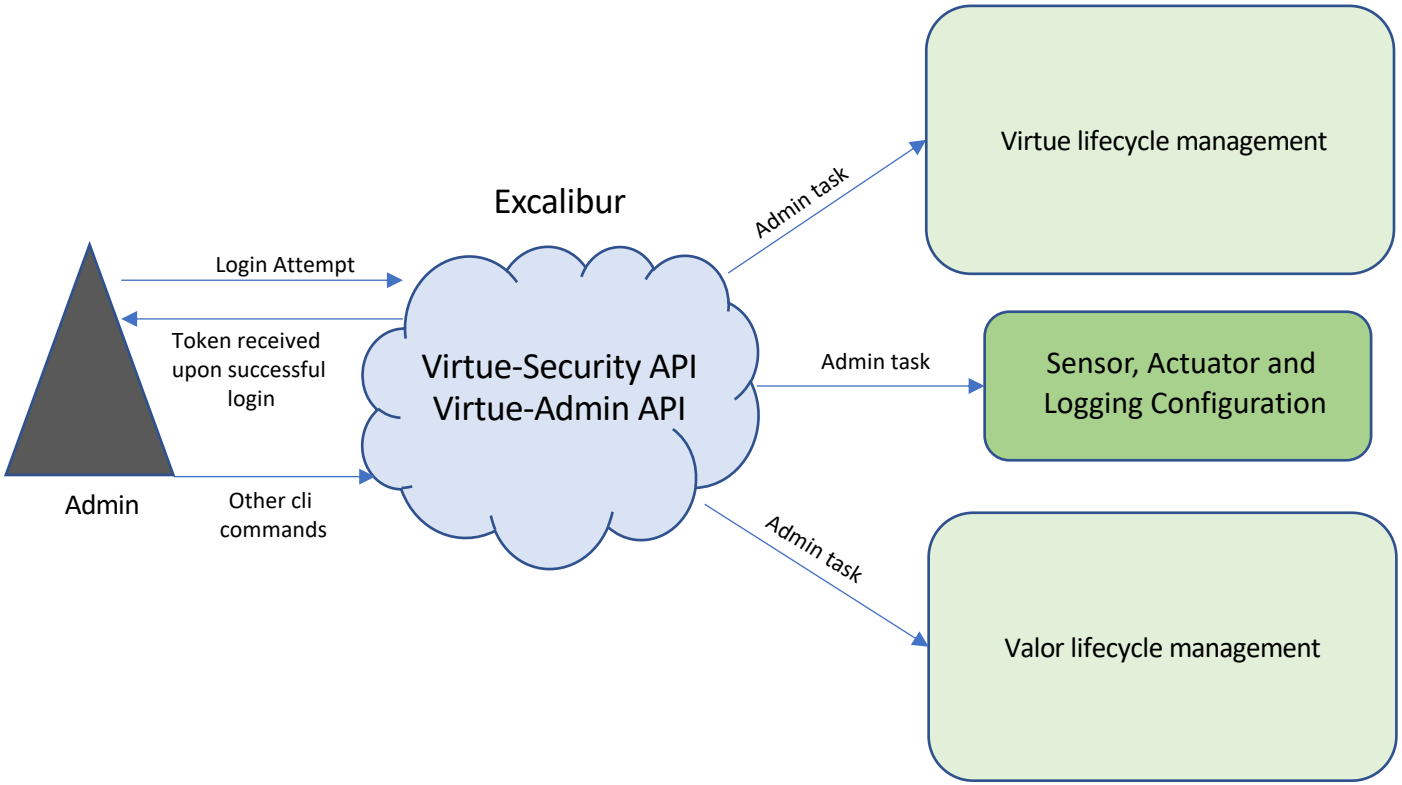
- U, V, GRESGROUP, A, R, T and OPS are Users, Virtues, Galahad Resource Group, Applications, Resources, Transducers and operations respectively
- OPS is a set of operations on VIRTUES and APPLICATIONS respectively
- UGRGA is a many to many user to galahad resource group assignment relation
  - UGRGA ⊆ USERS×UGRGA
- UVA is a one to many user to virtue assignment relation,
  - UVA ⊆ USERS × VIRTUES
- VA is a many to one virtue to galahad resource group assignment relation
  - VA ⊆ VIRTUES × GRESGROUP
- TA is a many to many transducer to galahad resource group assignment relation
  - TA ⊆ TRANSDUCERS × GRESGROUP
- RA is a many to many resource to galahad resource group assignment relation
  - RA ⊆ RESOURCES × GRESGROUP
- AA is a many to many application to galahad resource group assignment relation
  - AA ⊆ APPLICATIONS × GRESGROUP



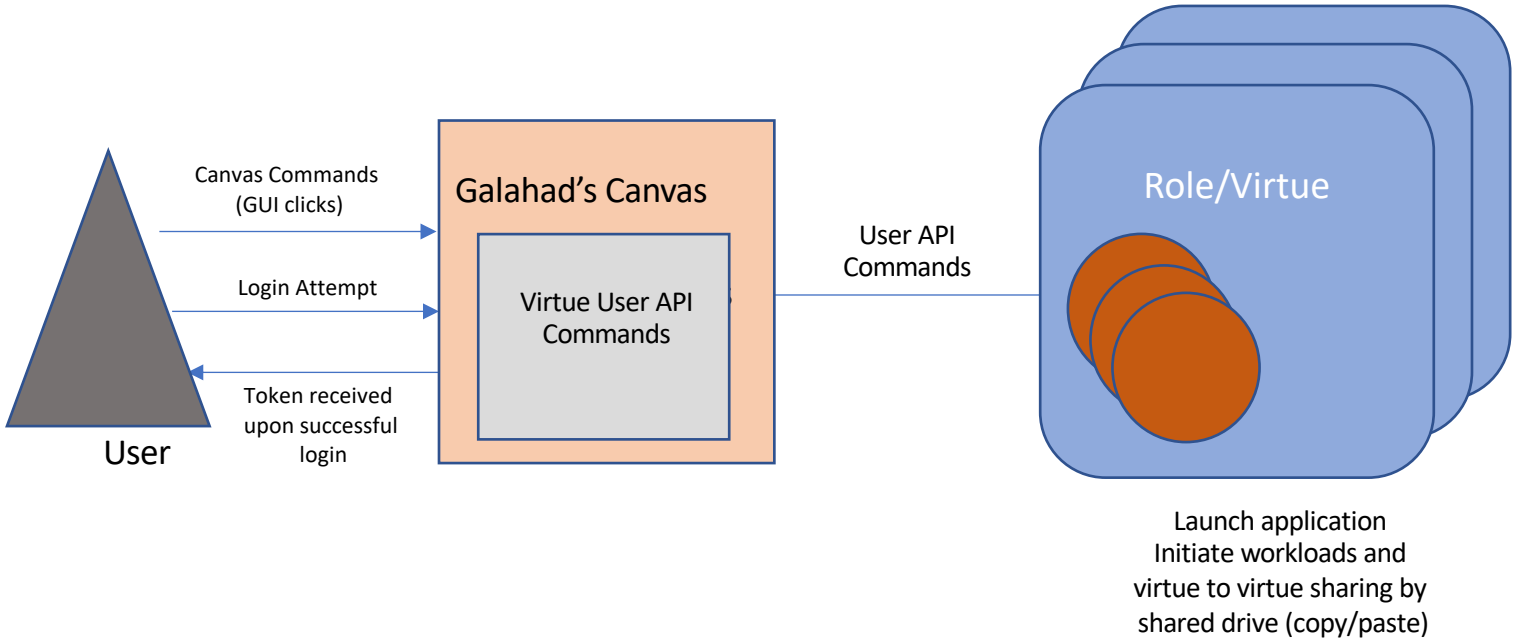
Current System Architecture



Current System Administrative Architecture



Current System Operational Architecture



System Functions: Spec Function to User API Call Mapping

ID	Spec Function	API Call	Description
1	CheckAccessVirtue	virtue get	Get information about a given Virtue
		virtue launch	Launches a virtue for the current user
		virtue reload state	Reload the virtue's state and IP address
		virtue stop	Stops a user's virtue
		user virtue list	Lists the current Virtues for the given user
2	CheckAccessApplication	virtue application launch	Launches an application in a running Virtue
		virtue application stop	Stops an application in a running Virtue
		application get	Get list of available applications
3	CheckAccessGalahadResourceGroup	role get	Get information about a role
		user role list	List roles available to the given user

System Functions: Galahad Functional Specification User API

Functions	Conditions	Result
System functions: User level operations.		
CheckAccessVirtue *(u, v, vop: NAME, out result: BOOLEAN)*	$v \in V$ $u \in U$ $vop \in OPS$	$result = v \in user\_virtues(u)$
CheckAccessApplication *(u, v, app, appop: NAME, out result: BOOLEAN)*	$v \in V$ $u \in U$ $appop \in OPS$ $app \in A$	$result = v \in user\_virtues(u)$ $\wedge$ (app, virtue_grgroup(v)) $\in AA$
CheckAccessGalahadResourceGroup *(u, gr: NAME, out result: BOOLEAN)*	$gr \in GRESGROUP$ $u \in U$	$result = (u, gr) \in UGRGA$

Administrative Functions: Spec Function to Admin API Call Mapping

ID	Spec Function	API Call	Description
1	CreateGRGroup	role create	Creates a Role with the given parameters
2	DestroyGRGroup	role destroy	Destroys a role
3	CreateVirtue	virtue create	Create a Virtue instance for a User/Role combination
4	DestroyVirtue	virtue destroy	Destroy a Virtue
5	AuthorizeUser	user role authorize	Authorizes the indicated Role for the given User
6	UnauthorizeUser	user role unauthorize	Removes authorization for a Role for a User
7	CreateResource	resource create	Creates a Resource with the given parameters
8	DestroyResource	resource destroy	Destroys a resource
9	AddApplication	application add	Add an application to the system
10	AttachResource	resource attach	Attach the indicated resource to the indicated Virtue
11	DetachResource	resource detach	Detaches indicated resource from the indicated Virtue
12	EnableTransducer	transducer enable	Enable a transducer on a Virtue
13	DisableTransducer	transducer disable	Disable a transducer on a Virtue
14	EnableAllTransducer	transducer enable all	Enable a transducer on all Virtues
15	DisableAllTransducer	transducer disable all	Disable a transducer on all Virtues

Galahad Functional Specification (Administrative Functions)

Functions	Conditions	Updates
Administrative functions: Creation and maintenance of the system elements performed by Admin.		
CreateGRGroup *(Apps, Res, Tds: 2^{NAME}, nwrules: STRING, grgroup: NAME)*	$grgroup \notin GRESGROUP$ $Apps \subseteq A$ $Res \subseteq R$ $Tds \subseteq T$	$GRESGROUP' = GRESGROUP \cup \{grgroup\}$ $AA' = AA \cup (\{grgroup\} \times Apps)$ $TA' = TA \cup (\{grgroup\} \times Tds)$ $RA' = RA \cup (\{grgroup\} \times Res)$ $assigned\_users' = assigned\_users \cup \{grgroup \longmapsto \phi\}$
DestroyGRGroup *(grgroup : NAME)*	$grgroup \in GRESGROUP$ $assigned\_virtues(grgroup) = \phi$ $assigned\_grgroupusers(grgroup) = \phi$	$AA' = AA \setminus \{a : A \cdot a \longmapsto grgroup\}$ $TA' = TA \setminus \{t : T \cdot t \longmapsto grgroup\}$ $RA' = RA \setminus \{r : R \cdot r \longmapsto grgroup\}$ $assigned\_users' = assigned\_users \setminus \{grgroup \longmapsto assigned\_users(grgroup)\}$ $GRESGROUP' = GRESGROUP \setminus \{grgroup\}$
CreateVirtue *(v, u, grgroup: NAME)*	$v \notin VIRTUES$ $u \in USERS$ $u \in assigned\_users(grgroup)$ $grgroup \notin \{gr \in GRESGROUP \mid gr = virtue\_grgroup(v) \wedge v \in user\_virtues(u)\}$	$VIRTUES' = VIRTUES \cup \{v\}$ $VA' = VA \cup \{v \longmapsto grgroup\}$ $UVA' = UVA \cup \{u \longmapsto v\}$ $assigned\_virtues' = assigned\_virtues \cup \{grgroup \longmapsto \phi\}$ $user\_virtues' = user\_virtues \setminus \{u \longmapsto user\_virtues(u)\} \cup \{u \longmapsto (user\_virtues(u) \cup \{v\})\}$ virtue_grgroup' = virtue_grgroup $\cup$ { v $\longmapsto$ grgroup}
DestroyVirtue *(v: NAME)*	$v \in V$	$VA' = VA \setminus \{v \longmapsto grgroup\}$ $UVA' = UVA \setminus \{u \longmapsto v\}$ $VIRTUES' = VIRTUES \setminus \{v\}$ $assigned\_virtues' = assigned\_virtues \setminus \{grgroup \longmapsto assigned\_virtues(grgroup)\}$ $user\_virtues' = user\_virtues \setminus \{u \longmapsto user\_virtues(u)\} \cup \{u \longmapsto (user\_virtues(u) \setminus \{v\})\}$ virtue_grgroup' = virtue_grgroup \ { v $\longmapsto$ virtue_grgroup(v)}
AuthorizeUser *(u, grgroup: NAME)*	$u \in U$ $grgroup \in GRESGROUP$ $u \notin assigned\_users(grgroup)$	$UA' = UA \cup \{u \longmapsto grgroup\}$ $assigned\_users' = assigned\_users \setminus \{grgroup \longmapsto assigned\_users(grgroup)\} \cup \{grgroup \longmapsto (assigned\_users(grgroup) \cup \{u\})\}$
UnauthorizeUser *(u, grgroup: NAME)*	$u \in U$ $grgroup \in GALAHADROLES$ $u \in assigned\_users(grole)$ $grgroup \notin \{gr \in GRESGROUP \mid gr = virtue\_grgroup(v) \wedge v \in user\_virtues(u)\}$	$UA' = UA \setminus \{u \longmapsto grgroup\}$ $assigned\_users' = assigned\_users \setminus \{grgroup \longmapsto assigned\_users(grgroup)\} \cup \{grgroup \longmapsto (assigned\_users(grgroup) \setminus \{u\})\}$

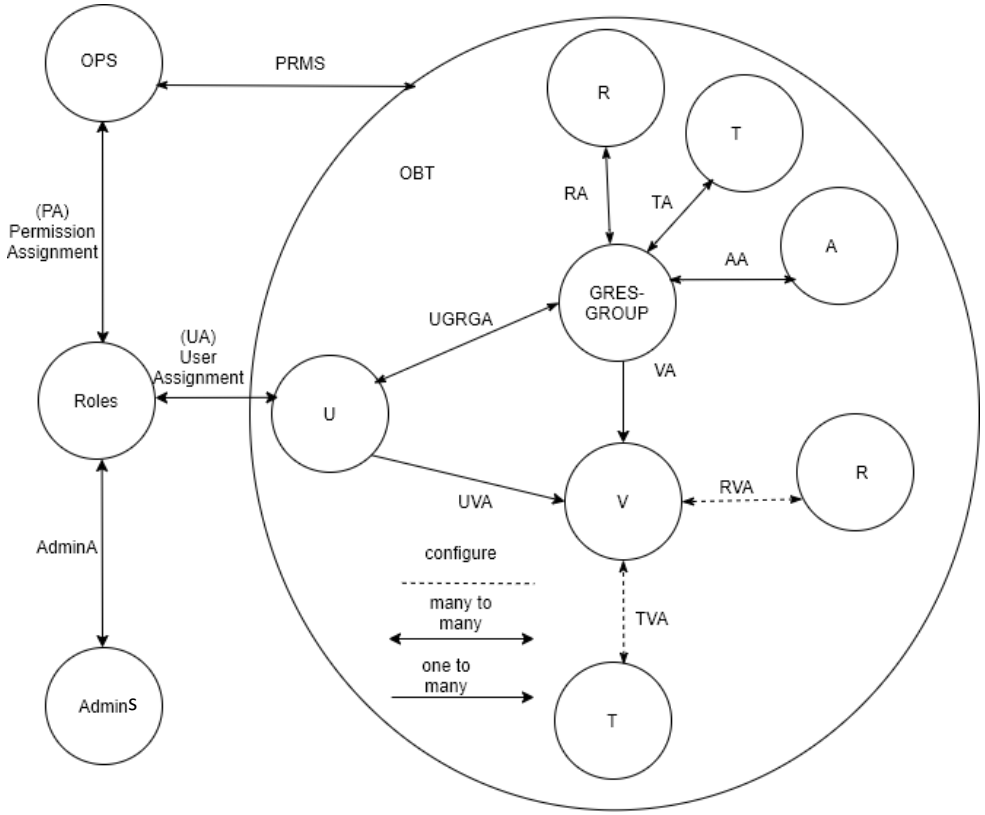
Galahad Functional Specification (Administrative Functions)

CreateResource *(res: NAME)*	$res \notin R$	$R' = R \cup \{res\}$
DestroyResource *(res: NAME)*	$res \in R$	$R' = R \backslash \{res\}$
AddApplication *(app: NAME)*	$app \notin A$	$A' = A \cup \{app\}$
AttachResource *(res,v: NAME)*	$res \in R$ $v \in V$ $res \notin virtue\_resources(v)$	$RVA' = RVA \cup \{res \longmapsto v\}$ $virtue\_resources' = virtue\_resources \backslash$ $\{v \longmapsto virtue\_resources(v)\} \cup$ $\{v \longmapsto (virtue\_resources(v) \cup \{res\})\}$
DetachResource *(res, v: NAME)*	$res \in R$ $v \in V$ $res \in virtue\_resources(v)$	$RVA' = RVA \backslash \{res \longmapsto v\}$ $virtue\_resources' = virtue\_resources \backslash$ $\{v \longmapsto virtue\_resources(v)\} \cup$ $\{v \longmapsto (virtue\_resources(v) \backslash \{res\})\}$
EnableTransducer *(td, v: NAME; transduceconfig: STRING)*	$td \in T$ $v \in V$ $td \notin virtue\_transducers(v)$	$TVA' = TVA \cup \{td \longmapsto v\}$ $virtue\_transducers' = virtue\_transducers \backslash$ $\{v \longmapsto virtue\_transducers(v)\} \cup$ $\{v \longmapsto (virtue\_transducers(v) \cup \{td\})\}$
DisableTransducer *(td, v: NAME)*	$td \in T$ $v \in V$ $td \in virtue\_transducers(v)$	$TVA' = TVA \backslash \{td \longmapsto v\}$ $virtue\_transducers' = virtue\_transducers \backslash$ $\{v \longmapsto virtue\_transducers(v)\} \cup$ $\{v \longmapsto (virtue\_transducers(v) \backslash \{td\})\}$
EnableAllTransducer *(td: NAME, tdconfig: STRING)*	$td \in T$ $V \neq \phi$ $td \notin virtue\_transducers(v : V)$	$TVA' = TVA \cup \{v : V \cdot td \longmapsto v\}$ $virtue\_transducers' = virtue\_transducers \backslash$ $\{v \longmapsto virtue\_transducers(v)\} \cup$ $\{v \longmapsto (virtue\_transducers(v) \cup \{td\})\}$
DisableAllTransducer *(td: NAME)*	$td \in T$ $V \neq \phi$ $td \in virtue\_transducers(v : V)$	$TVA' = TVA \backslash \{v : V \cdot td \longmapsto v\}$ $virtue\_transducers' = virtue\_transducers \backslash$ $\{v \longmapsto virtue\_transducers(v)\} \cup$ $\{v \longmapsto (virtue\_transducers(v) \backslash \{td\})\}$

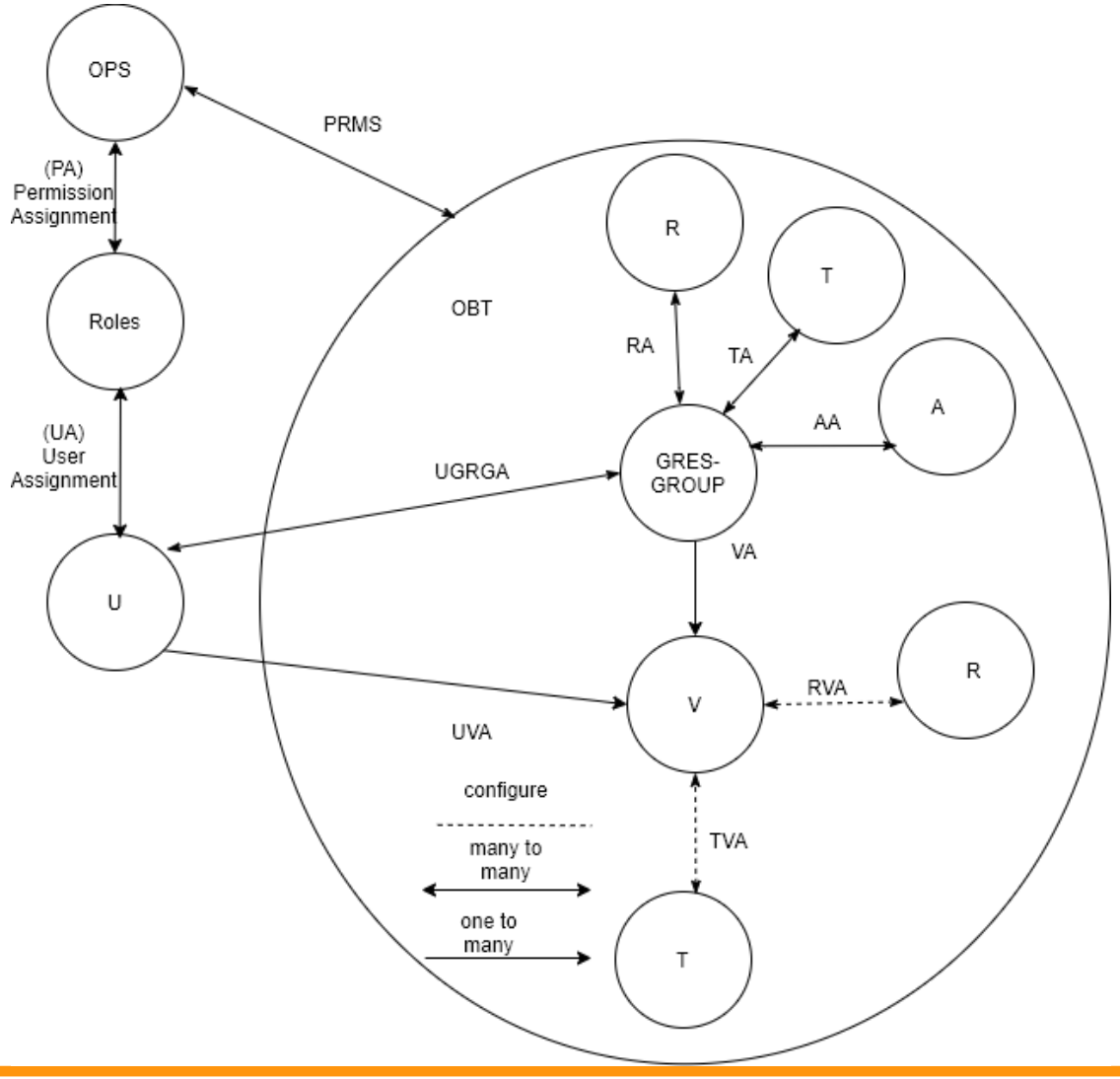
Proposed Administrative Access Control Model

Additional Definition

- Roles, Admins and OPS stands for proposed roles, administrative users and operations
- UA is a many to many user to roles assignment relation
  - UA ⊆ USERS X ROLES
- PA is a many-to-many permission to role assignment relation
  - PA ⊆ PRMS X ROLES
- PRMS is the set of permissions relation
  - PRMS ⊆ OPS OBT
- AdminA is a many to many Admin and Roles assignment relation
  - AdminA ⊆ Admin X Roles
- RVA is a many to many resource to virtue assignment relation
  - RVA ⊆ RESOURCES X VIRTUES
- TVA is a many to many transducer to virtue assignment relation
  - TVA ⊆ TRANSDUCERS X VIRTUES



Proposed Operational Access Control Model



Possible Future Directions

- Additional access control models
- Apply machine learning to characterize application use:
  - Detect abnormal behavior and automate a response
  - Help increase detection of zero-day attacks
  - Help increase detection of unknown viruses
- Develop models/white-papers on securing hypervisors better
- Apply SDN technology to increase performance on the backend.

Questions?

Resources

https://gitlab.com/utsa-ics/galahad/galahad

https://www.iarpa.gov/index.php/research-programs/virtue/virtue-baa?highlight=WyJ2aXJ0dWUiXQ==