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
Virtuous User Environment (VirtUE)

VirtUE seeks to leverage the federal government's impending migration to commercial cloud-based 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 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

World-Leading Research with Real-World Impact!
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.
• 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.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Better Forensics/Faster Recovery after an attack</td>
<td>Compromised machines can be cloned in compromised state for forensic analysis. After cloning, VM can be restored to known good state.</td>
</tr>
<tr>
<td>Control the default state of VM</td>
<td>Patches can be automatically incorporated. Failed upgrades can be reverted faster.</td>
</tr>
<tr>
<td>Cost effective</td>
<td>Scale infrastructure devices accordingly instead of one size fits all.</td>
</tr>
<tr>
<td>Leveraging Virtualization to provide better security</td>
<td>View interactions and contexts.</td>
</tr>
</tbody>
</table>
• 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.
A small, hardened, de-privileged Linux OS VM

Containers for easy packaging and security configuration

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

© James Benson
Galalahad Use Case

AWS Immutable Infrastructure

© James Benson
AWS Servers…

World-Leading Research with Real-World Impact!
VM’s shut down
VM’s turn on
Bad actor, probing…
Attack detected.
Aggressive Attack

World-Leading Research with Real-World Impact!
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

World-Leading Research with Real-World Impact!
Current System Architecture

Step: Login Mechanism

User/Admin

Login Attempt

Token received upon successful login

Excaliber

Virtue-Security API
Virtue-Admin API
Virtue-User API
Current System
Administrative Architecture

Excalibur

Virtue-Security API
Virtue-Admin API

Admin

Login Attempt
Token received upon successful login
Other cli commands

Admin task

Virtue lifecycle management

Sensor, Actuator and Logging Configuration

Valor lifecycle management

Admin task

Other CLI commands
Current System Operational Architecture

User

Canvas Commands (GUI clicks)
Login Attempt
Token received upon successful login

Galahad’s Canvas

Virtue User API Commands

User API Commands

Role/Virtue

Launch application
Initiate workloads and virtue to virtue sharing by shared drive (copy/paste)
### System Functions: Spec Function to User API Call Mapping

<table>
<thead>
<tr>
<th>ID</th>
<th>Spec Function</th>
<th>API Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CheckAccessVirtue</td>
<td>virtue get</td>
<td>Get information about a given Virtue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>virtue launch</td>
<td>Launches a virtue for the current user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>virtue reload state</td>
<td>Reload the virtue’s state and IP address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>virtue stop</td>
<td>Stops a user’s virtue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user virtue list</td>
<td>Lists the current Virtues for the given user</td>
</tr>
<tr>
<td>2</td>
<td>CheckAccessApplication</td>
<td>virtue application launch</td>
<td>Launches an application in a running Virtue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>virtue application stop</td>
<td>Stops an application in a running Virtue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>application get</td>
<td>Get list of available applications</td>
</tr>
<tr>
<td>3</td>
<td>CheckAccessGalahadResourceGroup</td>
<td>role get</td>
<td>Get information about a role</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user role list</td>
<td>List roles available to the given user</td>
</tr>
<tr>
<td>Functions</td>
<td>Conditions</td>
<td>Result</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>----------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>System functions: User level operations.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| CheckAccessVirtue                             | $v \in V$  
$u \in U$  
$vop \in OPS$                                                  | $result = v \in user_virtues(u)$                  |
| (u, v, vop: NAME, out result: BOOLEAN)        |                                                                 |                                                  |
| CheckAccessApplication                        | $v \in V$  
$u \in U$  
$appop \in OPS$  
$app \in A$                                             | $result = v \in user_virtues(u)$  
∧ (app, virtue_grgroup(v)) \in AA                      |
| (u, v, app, appop: NAME, out result: BOOLEAN) |                                                                 |                                                  |
| CheckAccessGalahadResourceGroup               | $gr \in GRESGROUP$  
$u \in U$                                                   | $result = (u, gr) \in UGRGA$                      |
| (u, gr: NAME, out result: BOOLEAN)            |                                                                 |                                                  |
## Administrative Functions: Specific Function to Admin API Call Mapping

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

<table>
<thead>
<tr>
<th>Functions</th>
<th>Conditions</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateGRGroup</td>
<td>grgroup (\not\in) GRESGROUP&lt;br&gt;Apps (\subseteq) A&lt;br&gt;Res (\subseteq) R&lt;br&gt;Tds (\subseteq) T</td>
<td>GRESGROUP' = GRESGROUP \cup {grgroup}&lt;br&gt;A' = A \cup {grgroup \times Apps}&lt;br&gt;T' = T \cup {grgroup \times Tds}&lt;br&gt;RA' = RA \cup {grgroup \times Res}&lt;br&gt;assigned users' = assigned users \cup {grgroup \mapsto \phi}</td>
</tr>
<tr>
<td>DestroyGRGroup (grgroup : NAME)</td>
<td>grgroup (\in) GRESGROUP&lt;br&gt;assigned_virtues(grgroup) = \phi&lt;br&gt;assigned_grgroupusers(grgroup) = \phi</td>
<td>Ax' = Ax \setminus {a : a \mapsto grgroup}&lt;br&gt;T'A = T \setminus {t : T \times t \mapsto grgroup}&lt;br&gt;RA' = RA \setminus {r : R \times r \mapsto grgroup}&lt;br&gt;assigned_users' = assigned users \setminus {grgroup \mapsto assigned_users}</td>
</tr>
<tr>
<td>CreateVirtue (v, u, grgroup : NAME)</td>
<td>v (\notin) VIRTUES&lt;br&gt;u (\in) USERS&lt;br&gt;u (\in) assigned_users(grgroup)&lt;br&gt;grgroup (\not\in) GRESGROUP&lt;br&gt;gr = virtue_grgroup(v) \land v (\in) user_virtues(u)</td>
<td>VIRTUES' = VIRTUES \cup {v}&lt;br&gt;V'A = VA \cup {v \mapsto grgroup}&lt;br&gt;UV'A = UV'A \cup {u \mapsto v}&lt;br&gt;assigned_virtues' = assigned_virtues \cup {grgroup \mapsto \phi}&lt;br&gt;user_virtues' = user_virtues \setminus {u \mapsto user_virtues} \cup {u \mapsto (user_virtues(u) \cup {v})}&lt;br&gt;virtue_grgroup' = virtue_grgroup \cup {v \mapsto grgroup}</td>
</tr>
<tr>
<td>DestroyVirtue (v : NAME)</td>
<td>v (\in) V</td>
<td>V'A = VA \setminus {v}&lt;br&gt;VIRTUES' = VIRTUES \setminus {v}&lt;br&gt;assigned_virtues' = assigned_virtues \setminus {grgroup \mapsto assigned_virtues}&lt;br&gt;user_virtues' = user_virtues \setminus {u \mapsto user_virtues} \cup {u \mapsto (user_virtues(u) \setminus {v})}&lt;br&gt;virtue_grgroup' = virtue_grgroup \setminus {v \mapsto virtue_grgroup}</td>
</tr>
<tr>
<td>AuthorizeUser (u, grgroup : NAME)</td>
<td>u (\in) U&lt;br&gt;grgroup (\in) GRESGROUP&lt;br&gt;u (\not\in) assigned_users(grgroup)</td>
<td>UA' = UA \cup {u \mapsto grgroup}&lt;br&gt;assigned_users' = assigned_users \cup {grgroup \mapsto assigned_users} \cup {grgroup \mapsto assigned_users(grgroup) \cup {u}}</td>
</tr>
<tr>
<td>UnauthorizeUser (u, grgroup : NAME)</td>
<td>u (\in) U&lt;br&gt;grgroup (\in) GALAHADROLES&lt;br&gt;u (\in) assigned_users(grrole)&lt;br&gt;grgroup (\not\in) GRESGROUP&lt;br&gt;gr = virtue_grgroup(v) \land v (\in) user_virtues(u)</td>
<td>UA' = UA \setminus {u \mapsto grgroup}&lt;br&gt;assigned_users' = assigned_users \setminus {grgroup \mapsto assigned_users} \cup {grgroup \mapsto (assigned_users(grgroup) \setminus {u})}</td>
</tr>
</tbody>
</table>

World-Leading Research with Real-World Impact!
<table>
<thead>
<tr>
<th>Function</th>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateResource (res: NAME)</td>
<td>res ( \not \in R )</td>
<td>( R' = R \cup { \text{res} } )</td>
</tr>
<tr>
<td>DestroyResource (res: NAME)</td>
<td>res ( \in R )</td>
<td>( R' = R \setminus { \text{res} } )</td>
</tr>
<tr>
<td>AddApplication (app: NAME)</td>
<td>app ( \not \in A )</td>
<td>( A' = A \cup { \text{app} } )</td>
</tr>
<tr>
<td>AttachResource (res, v: NAME)</td>
<td>res ( \in R ) &lt;br&gt; v ( \in V ) &lt;br&gt; res ( \not \in \text{virtue_resources}(v) )</td>
<td>( RVA' = RVA \setminus { \text{res} \mapsto v } )&lt;br&gt;( \text{virtue_resources}' = \text{virtue_resources} )&lt;br&gt;( { v \mapsto \text{virtue_resources}(v) } \cup { v \mapsto (\text{virtue_resources}(v) \setminus { \text{res} }) } )</td>
</tr>
<tr>
<td>DetachResource (res, v: NAME)</td>
<td>res ( \in R ) &lt;br&gt; v ( \in V ) &lt;br&gt; res ( \in \text{virtue_resources}(v) )</td>
<td>( RVA' = RVA \setminus { \text{res} \mapsto v } )&lt;br&gt;( \text{virtue_resources}' = \text{virtue_resources} )&lt;br&gt;( { v \mapsto \text{virtue_resources}(v) } \cup { v \mapsto (\text{virtue_resources}(v) \setminus { \text{res} }) } )</td>
</tr>
<tr>
<td>EnableTransducer (td, v: NAME; transducerconfig: STRING)</td>
<td>td ( \in T ) &lt;br&gt; v ( \in V ) &lt;br&gt; td ( \not \in \text{virtue_transducers}(v) )</td>
<td>( TVA' = TVA \cup { td \mapsto v } )&lt;br&gt;( \text{virtue_transducers}' = \text{virtue_transducers} )&lt;br&gt;( { v \mapsto \text{virtue_transducers}(v) } \cup { v \mapsto (\text{virtue_transducers}(v) \cup { td }) } )</td>
</tr>
<tr>
<td>DisableTransducer (td, v: NAME)</td>
<td>td ( \in T ) &lt;br&gt; v ( \in V ) &lt;br&gt; td ( \in \text{virtue_transducers}(v) )</td>
<td>TVA' = TVA \setminus { td \mapsto v } &lt;br&gt;( \text{virtue_transducers}' = \text{virtue_transducers} )&lt;br&gt;( { v \mapsto \text{virtue_transducers}(v) } \cup { v \mapsto (\text{virtue_transducers}(v) \setminus { td }) } )</td>
</tr>
<tr>
<td>EnableAllTransducer (td: NAME, ldomconfig: STRING)</td>
<td>td ( \in T ) &lt;br&gt; V ( \neq \emptyset ) &lt;br&gt; td ( \not \in \text{virtue_transducers}(v : V) )</td>
<td>TVA' = TVA \cup { v : V \cdot td \mapsto v } &lt;br&gt;( \text{virtue_transducers}' = \text{virtue_transducers} )&lt;br&gt;( { v \mapsto \text{virtue_transducers}(v) } \cup { v \mapsto (\text{virtue_transducers}(v) \cup { td }) } )</td>
</tr>
<tr>
<td>DisableAllTransducer (td: NAME)</td>
<td>td ( \in T ) &lt;br&gt; V ( \neq \emptyset ) &lt;br&gt; td ( \in \text{virtue_transducers}(v : V) )</td>
<td>TVA' = TVA \setminus { v : V \cdot td \mapsto v } &lt;br&gt;( \text{virtue_transducers}' = \text{virtue_transducers} )&lt;br&gt;( { v \mapsto \text{virtue_transducers}(v) } \cup { v \mapsto (\text{virtue_transducers}(v) \setminus { td }) } )</td>
</tr>
</tbody>
</table>
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

World-Leading Research with Real-World Impact!
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==