2018 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
VEHICLE ELECTRONICS AND ARCHITECTURE (VEA) & GROUND SYSTEMS CYBER ENGINEERING (GSCE) TECHNICAL SESSION
AUGUST 7-9, 2018 - NOVI, MICHIGAN

AUTHENTICATION AND AUTHORIZATION USING THE VICTORY ACCESS CONTROL SOFTWARE

Andy Rodriguez
ASRC Federal
Aberdeen, MD

Leonard Elliott
TARDEC
Warren, MI

Dina Keane
Jason Broczkowski
John Skrletts
Mark Moerdyk
ASRC Federal
Aberdeen, MD

Disclaimer: Reference herein to any specific commercial company, product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the Department of the Army (DoA). The opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or the DoA, and shall not be used for advertising or product endorsement purposes.

ABSTRACT

The Vehicular Integration for Command, Control, Communication, Computers, Intelligence, Surveillance and Reconnaissance / Electronic Warfare (C4ISR/EW) Interoperability (VICTORY) standards is an open architecture that defines how software and hardware are shared as common resources among services that make up a platform's capabilities such as Ethernet switches and routers, end nodes, processing units, as well as functionality such as position and navigation systems, radios, health monitoring, and automotive. The VICTORY standard enables reducing the total Size, Weight, and Power (SWaP), and Costs (SWaP-C) on a platform. As part of the Information Assurance (IA) capabilities of the VICTORY standard, the VICTORY Access Control Framework (VACF) provides protection to these shared resources in the form of an Attribute-Based Access Control (ABAC) system. The VACF is composed of five VICTORY component types: Authentication, Attribute Store, Policy Store, Policy Decision, and Policy Enforcement Services. This paper will discuss how the VICTORY Access Control Software (VACS), an implementation of VACF developed by the U.S. Army Tank Automotive Research, Development, and Engineering Center (TARDEC), can enable authentication and authorization on ground combat vehicles.

INTRODUCTION: VICTORY, VACF, ETC.

The Vehicular Integration for Command, Control, Communication, Computers, Intelligence, Surveillance, and Reconnaissance/Electronic Warfare (C4ISR/EW) Interoperability (VICTORY) standard is an open architecture and specification for C4ISR/EW integration on ground vehicles and any other platform. The VICTORY standard is a framework that is composed of an architecture, a set of standard specifications, and a set of reference designs. The architecture established by VICTORY defines common terminology, systems, components, and interfaces. The VICTORY architecture also defines an infrastructure called the VICTORY Data Bus (VDB), which provides the foundation for sharing resources and data within the vehicle. The standard specifications provide the technical details that aim to be used as reference for the system acquisition and science and technology communities to employ new practices, modernize current activities, and draft engineering change proposals. Finally, the VICTORY reference designs demonstrate how to combine the VICTORY standard specifications into an example implementation.

The VDB system is a collection of component types that make up an intra-vehicle network. An instance of a VDB is composed of hardware and software that implements the specific VICTORY component types selected by the implementer to accomplish the desired capabilities. Since the VDB provides access to many critical components in the vehicle network, it is imperative to protect and secure those components by controlling the access to those components. VICTORY adopted an Attribute-Based Access Control (ABAC) model in conjunction with the Information Assurance component types that helps maintain a secure, protected environment. The VICTORY Access Control Framework (VACF) is a subset of the VICTORY standard which provides an ABAC system. The ABAC system may be used to protect the network's management interfaces or any other arbitrary resource residing on the VDB. The VACF provides security features such as authentication, authorization, Transport Layer Security (TLS), Data-At-Rest, and Data-Signing/Signed-Data-Verification, and is comprised of five VICTORY component types: Authentication, Attribute Store, Policy Store, Policy Decision, and Policy Enforcement Services. The Authentication Service verifies multiple forms of credentials, while the other four services make up the ABAC portion of the authorization mechanism. Figure 1 represents the components in the VACF and how they interact with each other to provide authentication and authorization.

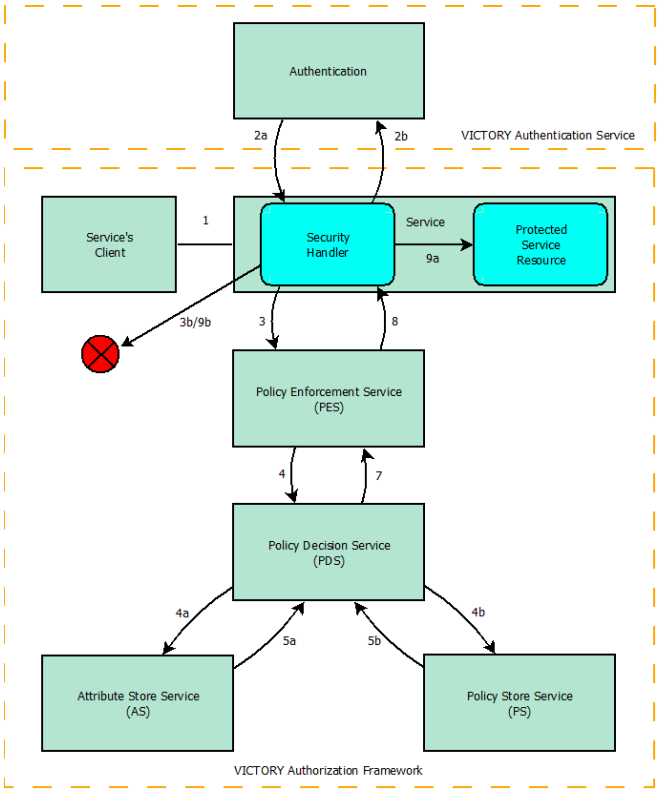

Figure 1: VACF component's Structure and Workflow [1].

When a client requests an action (i.e., a web method) on a resource, the resource extracts the client's credentials (e.g., username/password, X.509 certificate, or SAML token) and queries the Authentication Service for verification. If the client

is authenticated, then the authorization process takes place. The same credentials are used to identify the requester as a subject, and along with the requested action on the resource, an authorization query is constructed. This query is processed by an ABAC implementation, resulting in either a PERMIT or DENY response based on the client's attributes and the policies that have been defined.

PREVIOUS WORK ON VACF

A paper written in 2015 entitled "Implementing the VICTORY Access Control Framework in a Military Ground Vehicle" [2], explored the VACF specification, supporting technologies, and presented many obstacles that had been encountered during efforts to deploy the system in a relevant Military Ground Vehicle (MGV) environment. The paper also noted that the authoring of the VACF specification through the VICTORY Standards Body had been done without due consideration of commercial off-the-shelf (COTS) and government off-the-shelf (GOTS) solutions. Since the VICTORY Standards Body purportedly leverages an "adopt, adapt, author" methodology, this apparent deviation was noted and several potential COTS and GOTS candidates for adoption were proposed including the U.S. Army's *Tactical Service Security Software (TS3).* TS3, the paper asserted, seemed to be an ideal alternative due to its target environment, maturity, and the similarity to the existing VACF and protocols. The paper concluded with the recommendation that the VICTORY Standards Body should explore the COTS/GOTS alternatives and adopt the most applicable.

In the years since the 2015 VACF paper, participants in the VICTORY community have performed due diligence by examining many COTS and GOTS solutions as candidates for adoption by VICTORY. These solutions include TS3, DISA's OS-ABAC, and Axiomatic's suite of ABAC solutions. These products were examined through experimentation, when possible, and review of documentation. All the products were found to have deficiencies which substantially hindered the ability for MGV programs to deploy their components in a modular and interoperable manner. The primary deficiencies fell into two broad categories:

1. The product leveraged SAML, XACML, and a modular architecture, but had made implementation decisions which did not properly support these protocols and/or constrained modularity.
2. The product provided services and interfaces which were only intended to be used by software clients/handlers developed by the service developer, limiting interoperability.

Given the nature of the deficiencies identified, the reluctance of the vendors to address them, and VICTORY's prioritization of interoperability, it was decided that adoption of a current COTS/GOTS solution would not be beneficial and work supporting the existing VACF would provide better value for the MGV community.

The VICTORY Access Control Software

The VICTORY Access Control Software (VACS) is a VICTORY-compliant implementation of the VICTORY Access Control Framework that provides a set of specifications for authentication and authorization security on a platform. VACS is a deployable-ready attribute-based access control software, practical for any type of platform including ground combat vehicles. VACS's capabilities are accomplished by implementing the same VICTORY component types that comprise the VACF: the Authentication, Attribute Store, Policy Store, Policy Decision, and Policy Enforcement Services.

Environment, Licensing, & Distribution

VACS utilizes servlet containers that are written in Java and deployed within an Apache Tomcat server allowing the VACS distribution to be somewhat platform independent. Since VACF components leverage the Security Assertion

Markup Language (SAML) 2.0 standard and the eXtensible Access Control Markup Language (XACML) 2.0 standard, VACS also makes use of SAML and XACML to carry out its access control capabilities. SAML and XACML are technologies used for security applications in other industries such as banking and healthcare. VACS is released under the Department of Defense (DoD) Community Source Usage Agreement Version 1.1 by U.S. Army Tank Automotive Research, Development, and Engineering Command (TARDEC). The distribution of VACS is authorized to DoD and U.S. DoD contractors only. Other requests for VACS shall be referred to TARDEC.

Authentication Service

The Authentication Service responds to credential verification requests asserting whether the credentials are valid or not. The VACF authentication service supports username/ password and PKI authentication. For example, determining whether a username and password combination is valid, or if an X.509 Certificate is part of a trusted certificate chain.

Attribute Store Service

The Attribute Store (AS) Service plays a key role in the implementation of ABAC that VACS provides. The AS stores attributes of a subject. The Policy Decision Service uses the AS in conjunction with the Policy Store to provide authorization decisions based on the principal's attributes and policies. The AS provides a fully-featured management interface for querying, adding, and removing subject attributes.

Policy Store Service

The Policy Store (PS) Service plays a similar significant role as the Attribute Store. The PS stores the XACML policies for the platform. In an authorization request, the Policy Decision Service evaluates the request against the attributes in the AS and the policies stored in the PS. The XACML 2.0 standard is used to store and transfer policies. The PS provides a fully-featured management interface for querying, adding, and removing security policies.

Policy Decision Service

The Policy Decision Service (PDS) is the main agent that makes the authorization decision based upon the principal's attributes, which the PDS gathers from the AS, and the system policies fetched from the PS. The PDS provides to the Policy Enforcement Service the information necessary to provide the final decision to grant or deny the request. In a typical PDS request, the Policy Enforcement Service would receive the authorization request containing the client's credentials, the resource involved in the request, and the action that the client wants to perform upon the resource.

The Policy Enforcement Service passes the request to the PDS, which fetches the client's attributes from the AS and the applicable policies from the PS. The PDS then computes a decision based on the information received from the AS and PS, along with the authorization request. The decision is passed on to the Policy Enforcement Service as a SAML Response message, which, accordingly, turns the decision into a PERMIT or DENY.

Policy Enforcement Service

The Policy Enforcement Service (PES) plays an intermediary role between the security handler of a resource and the Policy Decision Service. The PES receives the information as an authorization request, which contains the client's information, a subject, the resource involved in the request, and the action that the client wants to perform upon the resource. That information is then passed to the PDS to evaluate.

After the evaluation, the PDS sends an authorization decision to the PES. This decision can take many forms, such as: PERMIT, DENY, INDETERMINATE, and NOT APPLICABLE.

The VICTORY standard allows for any specific implementation of the VACF to supplement a set of instructions and logic functions, for making the final decision upon receiving the authorization decision from the PDS. For instance, VACS maps every authorization decision that is not a PERMIT to a final DENY decision. It is up to the implementer to provide additional logic for determining the final decision. The security handler, which is part of the protected resource, receives the final decision from the PES and grants or denies the principal access to the protected resource.

libVictory/VACS Interoperability

One of the ways TARDEC has supported the MGV community's adoption of VICTORY is through the development and support of libVictory. libVictory is a C++ library that provides integrators with the ability to rapidly implement VICTORY clients and services by creating, managing, and pushing/pulling data through a C API which allows binding to many other programming languages. libVictory does not support the VICTORY IA components, instead it focuses on platform services such as Automotive, Position, Navigation, Direction-of-Travel, Threat, etc. Although libVictory clients and services can be configured with a course level of security and access control via XML Digital Signing and Transport Layer Security (TLS), libVictory services can also leverage VACS to provide finer-grained access control. libVictory services implement the security handler shown in Figure 1 and so, with proper configuration, are interoperable with VACS.

VACS-RELATED PRODUCTS

VACS provides a number of services, each with various configuration parameters, and it is not uncommon for an average user to encounter difficulty when deploying a distributed security solution of this nature. One of the products that has been developed to help mitigate the difficulties with deploying VACS is the VICTORY Access Control Software Management Tool (VACS-MT). The main objective of VACS-MT is to create a user interface that allows the user to manage attributes in a SAML Attribute Store and policies in a XACML Policy Store [3].

One frequently encountered difficulty with deploying VACS is managing the attribute and policy stores. Both attributes and policies are written in XML-based technologies such as SAML and XACML, so users who want to create attributes or policies must have a deep understanding of the SAML and XACML specifications. According to the SAML specification, a subject is the main actor whose attributes are being specified. An attribute is a descriptive characteristic of a subject. In creating an attribute, the user must have extensive knowledge of SAML constructs. Since SAML has many ways to create the needed attributes, and different rules that can be applied to the attribute itself, creating attributes and knowing if they are being created correctly is crucial. The challenges are the same, or greater, when working with policies and the complexity of XACML. VACS-MT helps overcome these difficulties with a custom graphical user interface (GUI), shown in Figure 2, that helps the user create attributes and policies, abstracting away from some of the hurdles that SAML and XACML present.

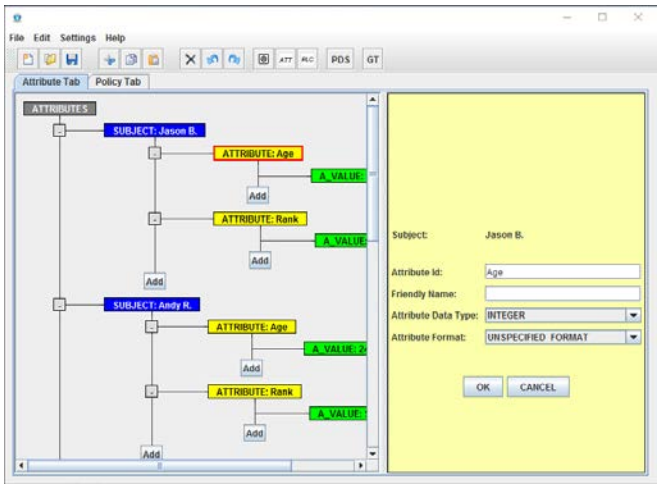

Figure 2: VACS-MT Main GUI

For example, let's assume that the user wants to create a new attribute. In the VACS-MT, the user clicks on the add button shown in the GUI, which then pulls up a panel where the user can enter the necessary information for the subject. By having areas where the user can put in known information, the user has a better understanding of what is going on with the attribute they are creating. When the user hits the OK button, VACS-MT adds the corresponding components and the item is then added to the local attribute set. This process makes the creation of attributes and polices easier, as the selections that the user has made will be converted into the appropriate XML format. The process eliminates guesswork by making sure that even complex attributes and polices are in the right format. Additionally, the local sets of attributes and policies can be applied to the destination attribute and/or policy stores via the VICTORY defined methods for remote management.

Another difficulty with VACS is when the user is creating attributes and polices, maintaining consistency within the created attribute or policy can get very complicated. With VACS-MT, there is input validation to assure the user that attributes and policies will remain persistent according to the rules set in place from other nodes. For example, if the user is creating an attribute that is of type integer, then VACS-MT will alert the user of the data mismatch if they try to enter in data that is not an integer as shown in Figure 3.

It is often desirable to verify the evaluation results for a newly created set of attributes and policies to make sure that what was created results in the authorization decisions that the user expects. VACS-MT has a utility that assists with the verification by taking the newly created attributes and polices and creating a list of tests that can be run automatically and provide the user with the results of those tests. With those results, the user can make sure that what they created for attributes and polices acts the way that they had intended.

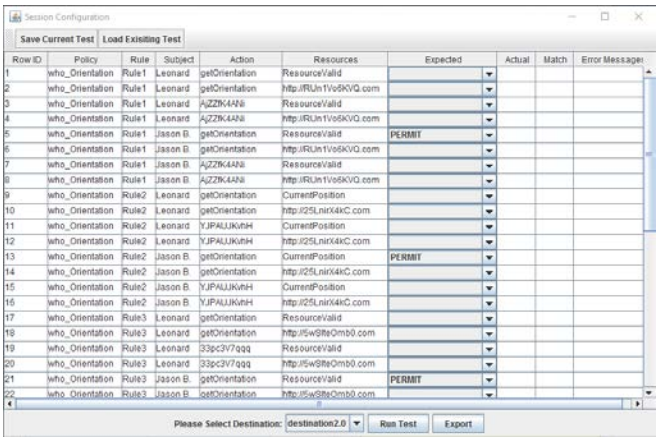

Figure 3: Input Validation Test Table

CONCLUSION

Authentication and authorization using the VACF is a viable approach to implementing interoperable, distributed access control. Although potential alternatives were identified and examined, they were deficient when it came to interoperability, which is a key objective of the VICTORY initiative. VACS, which is an implementation of the VACF, leverages established methods and protocols for implementing authentication and authorization using common standards such as SAML and XACML. Although there are challenges associated with deploying a distributed security solution such as VACS in a tactical environment, products such as libVictory and VACS-MT are available and can assist with configuration and deployment.

REFERENCES

[1] TARDEC-VEA, "VICTORY Access Control Software (VACS) version 1.0," Warren, Michigan, Available: https://confluence.di2e.net/pages/viewpage.action?spaceKey=VACS&title=VICTORY+Access+Control+Software+Home, December 2016.

[2] L. Elliott, "Implementing the VICTORY Access Control Framework in a Military Ground

Vehicle," Defense Technical Information Center, August 2015.

[3] J. Broczkowski, "VICTORY Access Control Software (VACS) Management Tool (VACS-MT) Initial Design Document Version 1.0," August 2017.