2015 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
VEHICLE ELECTRONICS AND ARCHITECTURE (VEA) TECHNICAL SESSION
AUGUST 4-6, 2015 – NOVI, MICHIGAN

IMPLEMENTING THE VICTORY ACCESS CONTROL FRAMEWORK IN A MILITARY GROUND VEHICLE

Leonard Elliott
Vehicle Electronics and Architecture
TARDEC
Warren, MI

Kim Woodward
Vehicle Software Department
DCS Corporation
Alexandria, VA

Alex LaBerge
Vehicle Systems Department
DCS Corporation
Warren, MI

ABSTRACT

The Vehicular Integration for Command, Control, Communications, Computers, Intelligence, and Surveillance/Electronic Warfare (C4ISR/EW) Interoperability (VICTORY) Standard provides an open architecture and technical specifications to promote sharing and reuse of resources within the military ground vehicle (MGV). The VICTORY Access Control Framework (VACF) provides services and mechanisms for protecting many of these shared-resources through the adoption of standards such as Security Attribute Markup Language (SAML) and eXtensible Access Control Markup Language (XACML). These technologies are typically used for securing an Enterprise Architecture and no fundamental issues appear to preclude their successful use within a MGV. However, despite consistent demand and pressure from Program Managers, and the successful deployment of many other VICTORY components, there has been no successful demonstration of these security components in an integrated vehicular environment. This paper presents a brief overview of the VACF and possible solutions for overcoming the practical challenges associated with implementing it in a MGV.

INTRODUCTION

The Vehicular Integration for Command, Control, Communications, Computers, Intelligence, and Surveillance/Electronic Warfare (C4ISR/EW) Interoperability (VICTORY) standard provides building-blocks for creating a Service-Oriented Architecture (SOA) within a MGV and the infrastructure created by combining these building-blocks is known as the VICTORY Data Bus (VDB). A SOA can provide a fabric through which vehicle subsystems can share resources and new capabilities can be inserted with relative ease. There is an increased need to protect these increasingly connected resources.

The VICTORY Access Control Framework (VACF) prescribes a mechanism for securing web-services connected to the VDB. The VACF components are a recommended, not required, feature of the VDB and consists of the Authentication Service and the VICTORY Authorization Framework (VAF). The VAF is composed of the Policy

Enforcement Service (PES), Policy Decision Service (PDS), Policy Store (PS), and Attribute Store (AS) Services. Variations of this framework are used in many enterprise systems (e.g. banking and healthcare). There are no identified issues which should prevent the successful implementation of this model in a MGV environment. Although the VICTORY ACF is not required by the VICTORY specification, most integrators find that protection is necessary to secure other services that are required to be available on the VDB. For example, if fully implemented, the VICTORY Shared-Processing Unit (SPU) Service exposes functionality to remotely reconfigure the host hardware and this is a function that must be protected. Despite the clear need to protect these distributed resources with some sort of access control, there has yet to be a successful deployment of the VACF in an integrated vehicular environment. Without a functioning VACF and security policy, integrators are choosing to omit functionality from required services, leading to failed VICTORY compliance tests.

In this paper we will briefly describe the VACF and conceptual operation within an MGV. We then discuss implementation and evaluation of the VACF by the VICTORY Standards Maturation (VSM) team at the Tank and Automotive Research Development and Engineering Center (TARDEC), and difficulties associated with deploying the VACF. Finally, we discuss possible solutions for providing access control to secure VICTORY services.

VICTORY ACCESS CONTROL FRAMEWORK

The VACF components leverage the SAML 2.0 and XACML 2.0 standards which are general purpose languages for describing and communicating security related information and transactions in an arbitrary security architecture. SAML and XACML provide comprehensive language to support security in complex systems and so the message sets and processing semantics are complex relative to the rest of the VICTORY specification (e.g. the SAML and XACML 2.0 core documents alone are several hundred pages) [1, 2]. Figure 1 illustrates a typical configuration of the VACF and the communication sequence for a successful access control request on the VICTORY Position Service. When a client attempts to perform an action on the Position Service, the security handler sends the client's credentials (i.e. username/password, X509, or SAML token) to the Authentication Service to verify authenticity. Upon notification of successful authentication, the security handler constructs and sends a SAML authorization decision query to the PES. The PES forwards the query to the PDS which queries the AS and PS, evaluates the retrieved information, and returns the authorization decision to the PES. The PES then returns the authorization decision to the resource who either executes the request or returns a "Not Authorized" message to the client.

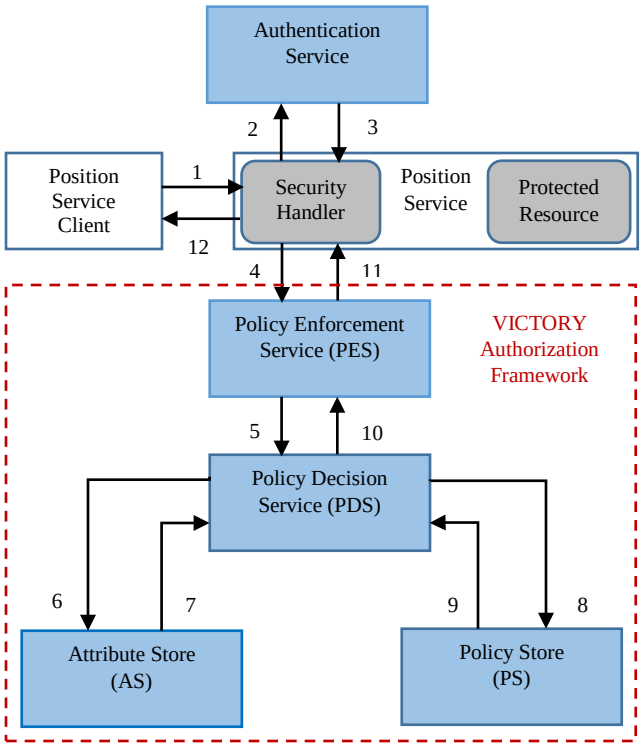

Figure 1: VACF Architecture and Communication Sequence

VACF PROTOTYPING AND VALIDATION

The VSM team at TARDEC has evaluated portions of the VICTORY specifications since version 0.7. The VSM team's primary goals are to ensure that the specifications are complete and unambiguous, but also to verify that the resulting components are practical logistically, and can operate in a MGV environment. The VSM team also verifies that software specifications can be implemented using a variety of programming languages and software tools, on a representative set of processing architectures and operating systems. When possible, open-source or commercial software packages are used to implement VICTORY components (e.g. the VICTORY Time Service was implemented using the open-source *ntpd* package). If no "turn-key" solutions exist then the VSM team designs and builds custom software to implement the component. In many cases, these prototypes are then integrated into a Government Open-Source Software VICTORY library called *libVictory*, which is available to other government organizations and contractors via https://software.forge.mil.

VACF Market Research

After studying the SAML and XACML specifications we suspected, given the relative complexity of these adopted standards, that sophisticated software would be needed to support the VACF. Market research was conducted to determine if existing software packages could be used to implement the VACF components. Many toolkits and products, primarily written in Java, supported both SAML 2.0 and XACML 2.0 to various degrees of conformance. In particular, the PDS was an item of concern due to the criticality of its role (i.e. interpreter of security policies), the complexity of XACML security policy processing rules, and the potential complexity of the security policies themselves. There were many PDS products available, both open-source and commercial. Many of the products identified were not fully compliant with the XACML specification and conformance test suite [3, 4]. The compliance status for a particular product was not always easy to assess and self-certification appeared to be standard. Table 1 lists a sample of XACML PDS engines including all of those that we identified as being fully XACML conformant.

Name	License	Compliant	Language
Axiomatics	Commercial	Yes	Java/C#
Heras-AF	Open-Source	Yes	Java
IBM Tivoli	Commercial	Yes	Java
Jericho	Commercial	Yes	Java
SunXACML	Open-Source	No	Java
XEngine	Open-Source	No	Java

Table 1: Sample of XACML Policy Decision Engines

These products all use Java and a Security Technical Implementation Guide (STIG) for Java does exist, but whether Java applications could be supported by MGV platforms was unknown. Given the high-frequency of zero-day vulnerabilities reported and the large attack surface provided by the Java core libraries [5], versus the comparatively slow patching cycle for vehicle systems, it was unclear whether Java would be prohibited for security reasons. Various organizations, including the VICTORY Information Assurance Working Group (IAWG), were queried to determine whether Java was definitely supported in a vehicular environment, but we were unable to identify a source to confirm this. Given this information, and the VSMs normal activity of verifying that VICTORY services could be supported by a wide variety of programming languages, the decision was made to build and evaluate VACF component prototypes written in C++ and using gSOAP for web-service support.

Prototyping VACF Components using C++

Prototypes for the PES, PDS, PS, and AS were built using C++ with the Genivia gSOAP toolkit being used to generate the appropriate C++ bindings from the VICTORY, SAML, and XACML schemas. The software prototypes were built to implement the behavior described in the VICTORY specification associated with the interfaces defined in the following Web-Service Definition Language (WSDL) files:

- Authentication.wsdl
- PolicyEnforcement.wsdl
- PolicyDecision.wsdl
- PolicyStore.wsdl
- AttributeStore.wsdl

The VACF services were implemented and evaluated, and detailed reports are available in the restricted section of the Defense Technical Information Center. Although these C++ prototypes adequately demonstrated basic operation of the VACF, problems were encountered when they were evaluated for compliance. The SAML and XACML specifications define languages in themselves and the resulting messages can be constructed in complex and varied forms. The subset of messages, formats, and options supported by the C++ prototype were adequate for providing basic access control functionality, but were nowhere near conformant with the SAML and XACML specifications. For example, the SAML specification requires that the VACF components have the ability to encrypt individual XML elements within a message. The C++ prototypes provided connection encryption via Transport Layer Security (TLS), and application-layer encryption via WS-Security encryption of the entire Simple Object Access Protocol (SOAP) message body, but lacked the ability to encrypt individual XML elements within the SOAP body. Several attempts were made to augment the C++ prototypes with additional resources like the OpenSAML library. After several months of continued effort it became apparent that it is likely not cost-effective to implement the VACF components in C or C++. The complexity of the VACF message sets and the preponderance of existing open-source support for SAML and XACML are built into languages like Java and the re-use of these tools is the most feasible option for implementing the VACF. These solutions are often deployed as Java servlets so the use of web-servers like Apache and an accompanying web-service stack like AXIS2 may also be required.

The VSM has conducted performance testing on the VICTORY services that are written in C++ with gSOAP support [6] and these services (contained within *libVictory*) have been tested on a variety of architectures including Intel, PowerPC, and ARM. The *libVictory* software is supported

by Linux and it has been tested on several variations including Debian, Embedded, Red Hat, Ubuntu, and Wind River Linux. The *libVictory* web-services supported by gSOAP are generally an order of magnitude faster than Java based web-servers [7]. There is a possibility that integrators may be more restricted in their choice of target hardware for hosting the VACF components, but this may not be an issue given the ever-increasing computational power provided by military hardware.

VACF Validation Conclusion

The VSM team's process of prototyping and evaluating VACF components using C++ showed that VACF components could be used to protect web-service on the vehicle. We also found that it would probably require sophisticated software and that, compared to our experience with other VICTORY services, there are a different set of issues impeding the deployment and configuration of the VACF. To summarize, the following problems were identified as significant obstacles to successfully deploying the VACF in an integrated vehicular environment:

- Complexity of the SAML/XACML specifications and availability of supporting software precludes the cost-effective implementation of VACF components in languages such as C/C++
- Unknown support for Java in mobile MGV environments
- Development of XACML policies for distributed systems can be complicated and may require special tools and/or the adoption of a reduced set of language constructs
- VACF components need to be running on each vehicle rather than a centralized location, as is common for enterprise systems
- Potential large cost impact on MGV programs if commercial enterprise software is procured on a per vehicle basis
- Use of servers and web-service stacks such as Apache and AXIS2 require significantly more memory and processing time than other web servers such as gSOAP
- Use of servers and web-service stacks such as Apache and AXIS2 may restrict choice of embedded targets for hosting VACF
- VACF is a potential bottleneck and single-point of failure for message traffic on the vehicle

VACF COMPONENT DEPLOYMENT OPTIONS

In this section we discuss several deployment options available to MGV integrators wishing to secure their web-services. For many VICTORY Services, including the Position, Orientation, Direction-of-Travel, SPU, Threat Detection and Reporting, Remote Weapon Services, and VDB Manager, the VSM prototypes have been incorporated into *libVictory*, a freely-available option for vehicle integrators wanting to deploy VICTORY components. There are also other reference code packages provided by the VICTORY Standards Support Office (VSSO). These alternatives provide sample code which leverage either the Qt framework or Java. However, TARDEC's *libVictory* is the only reference software which provides a well-documented C application programming interface (API), which significantly reduces the integration burden placed on software developers. The *libVictory* software also has additional support features such as formal bug reporting and tracking. The *libVictory* software is written in C++ and does not support the VACF components.

MGV integrators will most likely have to use Java to implement the VACF (either by leveraging the VSSO reference code, or other COTS Java packages). This means that vehicular software for Java is a prerequisite for successful deployment of the VACF. We decided to expand our search to definitively ascertain whether Java applications would be permitted to run on the vehicle. We were able to contact individuals from Project Manager Mission Command (PM-MC) who were able to verify that Java was in fact being used on the vehicle and that a product called Tactical Services Security System (TS3), which is written in Java, is currently fielded and provides web-service security within the MGV. This means that there is a precedent for using Java on the MGV and we assume that VICTORY applications may be developed in Java in accordance with the applicable STIGs. In the subsequent sections we further explore several options integrators have for providing web-service security including leveraging the VICTORY reference software, developing the VACF components from scratch, and adopting TS3 which, as we will show, provides similar interfaces and functionality to what the VACF currently attempts to provide.

VICTORY Reference Software

Since there is a precedent to assume that Java software may be used within the vehicle, it can be assumed that the VICTORY Reference Code provided by the VSSO is a viable option for integrators wishing to implement the VACF components. This software was intended to be a general purpose reference so integrators wishing to leverage it will have to address the following challenges:

- No API for integrating platform specific logic (e.g. XACML policies are hard-coded)
- Not fully SAML and XACML conformant

- Licensing appears to be more restrictive than other government software licenses (an applicant must petition the VSSO for a license)
- Ability to handle parallel access is not well understood
- No persistent storage for attributes and policies

Limitations such as these are common for any software that is originally used for proof-of-concept. This reference software is useful for validating the VICTORY specification however integrators may face significant challenges and costs correcting the SAML/XACML conformance deficiencies and inserting platform-specific features such as persistent storage.

VACF Components from Scratch

VICTORY provides web-service interface definitions and some degree of specification for communication semantics. Since this information is readily available, integrators and vendors may choose to implement VACF components without leveraging reference code. We do not think that it would be cost-effective to build SAML 2.0 and XACML 2.0 conformant VACF components without using existing open-source or commercial libraries and tools, but vendors may choose to implement the VICTORY and other vehicle specific logic.

Building VACF components "from scratch" would allow developers to have more control over the licensing of their product, but they will likely incur significant development costs to realize the same level of base functionality that is already provided by the VSSO reference code. Additionally, they will have to address the same obstacles that were identified for deploying the VSSO reference code including handling of parallel access, persistent storage, and SAML/XACML conformance. There may also be significant costs associated with implementing other non-essential features of VICTORY (e.g. Auto-discovery), VICTORY compliance testing, and other types of accreditation and certification efforts.

Tactical Services Security System (TS3)

TS3 software was initially identified during the effort to check the assumption that Java can be used to implement VACF components on the vehicle. TS3 is a government off-the-shelf (GOTS) software package that has been under development for over a decade and is managed by Communications-Electronics Command (CECOM) Software Engineering Center. TS3 was designed to provide security to web-services in a tactical environment and was developed in accordance with, and largely conforms to the Net-Centric Enterprise Services (NCES) security model, developed by the Defense Information Systems Agency (DISA) [11]. The Army has already invested approximately $10M developing this software and the PM estimates that approximately $700,000 - $800,000 per year is spent on maintaining and updating this product to be compliant with the STIGs [8]. We were informed that this yearly maintenance cost is comparable to what vendors typically charge for annual maintenance on similar COTS products [9]. TS3 has been accredited for use in the field and is used by the Army's Data Dissemination Services (DDS), Army Common Software services, and Global Command & Control System Army (GCCS-A) v4.3 widgets and services [10]**.** We were able to obtain a copy of the software with code samples and developer's guide.

The TS3 developer's guide provides extensive documentation describing the theory of operation and specifics of configuring and integrating with the TS3 software. Figure 2 depicts TS3 architectural components with a sample communication sequence for a successful access control request.

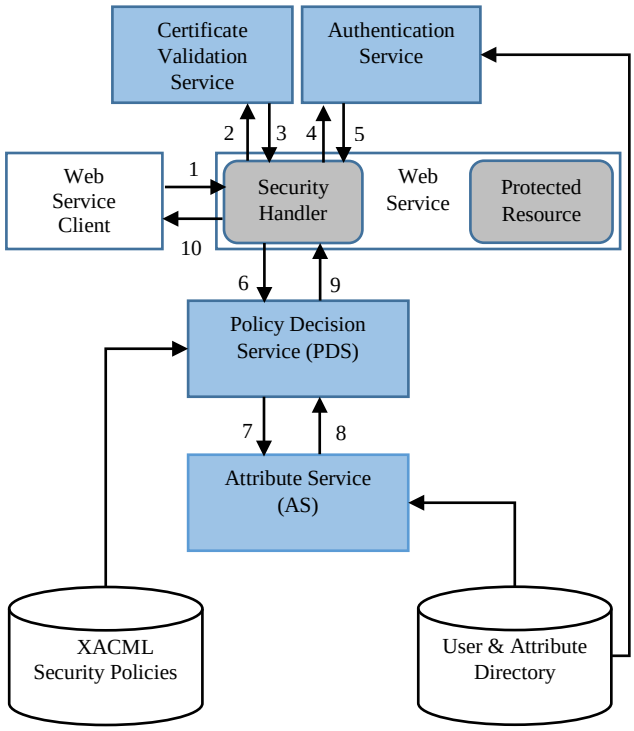

Figure 2: TS3 Architecture and Communication Sequence

When a client attempts to perform an action on the web-service, the security handler optionally sends the client's X509 certificate to the Certificate Validation Service. Upon successful certificate validation, the security handler sends the client's credentials (i.e. username/password, X509, or SAML token) to the Authentication Service to verify authenticity. Upon notification of successful authentication, the security handler constructs and sends a SAML

authorization decision query to the PDS. The PDS queries the AS and evaluates the retrieved information against its policy set, and returns the authorization decision to the resource who either executes the request or returns a "Not Authorized" message to the client.

The similarities between the TS3 architecture and the VACF are significant. Both use SAML 2.0 and XACML 2.0 and in many cases use the same messages. Many of the components within each framework are the same including the Authentication Service, Policy Decision Service, and Attribute Service and both use SOAP and provide freely available WSDLs with similarly defined operations. TS3 even leverages the same XACML engine that is used within the VSSO reference code. TS3 additionally provides security handlers that can be easily integrated into Java or .NET based web-services. The software also provides other functionality such as the ability to check certificate revocation status via the Online Certificate Status Protocol (OCSP), auditing, and graphical configuration tools. Given the similarities between each system's functionality, interfaces, and communication formats, TS3 actually seems well positioned to provide "turn-key" access control functionality to the VDB even though it is not technically VICTORY compliant. Vehicle integrators should consider TS3 interfaces as a viable "free" option for implementing security for their web-service interfaces. It is surprising given the capabilities, maturity, and the Army's investment in TS3 that the VICTORY Work Groups (WG) did not consider adopting TS3 interfaces.

CONCLUSION

In this paper we provided a basic overview of the VACF and described the VSM team's effort to prototype and evaluate VACF components and the results of this effort. We provided evidence that suggests that it is not cost-effective to implement VACF components without leveraging existing Java resources or products. We have shown that VACF components (and any software conformant to the full SAML and XACML specifications) must be sophisticated and therefore potentially costly. We explored the limitations of the VSSO reference software and concluded that any attempt to implement and deploy production grade VACF components could place a large software development burden on vehicle programs, particularly if efforts are duplicative. Finally, we provided an overview of TS3, a freely available GOTS product that has existed for over decade, has been fielded in MGV environments, and has actively been supported with over $10M in investment through the CECOM Software Engineering Center. We examined the TS3 architecture and showed that it provides functions and interfaces similar to those of the VACF including use of the same protocols and message sets.

Moving forward, we believe that the VICTORY community has a tremendous opportunity to provide value to the MGV community by: 1) considering the adoption of TS3 software interfaces and 2) leveraging the significant investment that the Army has already made in this software. This would enable the reuse of the proven TS3 software while still decoupling interfaces and implementation; allowing integrators to choose different software if desired. This information is timely, due to the fact that minimal investment in VACF software needs to be discarded, given that little work on production-grade VACF components has occurred. We have brought this to the attention of the VICTORY IAWG group and are actively collaborating with them to explore this potential solution. In addition, the VSM team is researching the integration of *libVictory* services with TS3. Our expectation is to help guide MGV integrators in the near future towards a relatively low-risk, low-cost solution, for securing their VICTORY web interfaces.

REFERENCES

[1] OASIS, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf March 2005.

[2] OASIS, "eXtensible Access Control Markup Language (XACML) Version 2.0", OASIS Standard, [Online]. Available: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf February 2005.

[3] OASIS, "XACML 2.0 Conformance Tests (Draft)", Oct. 2005. [Online]. Available: https://www.oasis-open.org/committees/download.php/14877/ConformanceTests.html October 2005.

[4] N. Li, J. Hwang, and T. Xie, "Multiple-Implementation Testing for XACML Implementation", [Online]. Available: http://web.engr.illinois.edu/~taoxie/publications/tavweb08xacml.pdf November 2008.

[5] D. Svoboda, "Java Zero Day Vulnerabilities", [Online]. Available: https://blog.sei.cmu.edu/post.cfm/java-zero-day-vulnerabilities December 2014.

[6] M. Russell, "Performance of an Embedded Platform Aggregating and Executing Core VICTORY Services", Defense Technical Information Center, August 2012.

[7] M. Govindaraju et al., "Toward Characterizing the Performance of SOAP Toolkits", Fifth IEEE/ACM International Workshop on Grid Computing, pages 365-372, November 2004.

[8] J.A. Landmesser, private communication, May 2015.

[9] J.A. Landmesser, private communication, April 2015.

[10] C. Smith, "Security First…Protecting the Army's Web Services", The Link, pages 20-21, [Online]. Available: http://cecom.army.mil/THE-LINK/2014/spring/files/assets/common/downloads/CECOMLink-Mar14-WEB.pdf March 2014.

[11] DISA, "A Security Architecture for Net-Centric Enterprise Services (NCES)", Version 0.3, March 2004.