Paper Submission for the 23rd International Command and Control Research and Technology Symposium
(Pensacola, FL / 2018)

Paper 15

Track 10: Cyberspace Challenges

IT Asset Management for Cyber Defense

Jonathan R. Agre
Information Technology & Systems Division
Institute for Defense Analyses
4850 Mark Center Drive
Alexandria VA 22311
USA
jagre@ida.org

Karen D. Gordon
Information Technology & Systems Division
Institute for Defense Analyses
4850 Mark Center Drive
Alexandria VA 22311
USA
kgordon@ida.org

Ronald G. Bechtold
Information Technology & Systems Division
Institute for Defense Analyses
4850 Mark Center Drive
Alexandria VA 22311
USA
rbechtol@ida.org

Marius S. Vassiliou
Science and Technology Division
Institute for Defense Analyses
4850 Mark Center Drive
Alexandria VA 22311
USA
mvassili@ida.org

Keywords: Cyber; Cyber Defense; IT Asset Management; Software Inventory; Information Systems

Abstract

How can we defend what we don't know we have? At first glance, gathering information about our IT assets might appear straightforward. However, in a complex enterprise with multiple mission components and possibly multiple sites, as well as large numbers of users, the situation is more complicated. As an enterprise evolves over time, knowledge about applications in inventory, how they are implemented, and how they interact becomes dispersed and disorganized. If organizations always exercised proper configuration management and maintained perfect registries of necessary information capturing all dependencies, then cyber defense would be easier. However, in the real world, entities evolve haphazardly and the required information is not maintained, particularly since applications can be very complex and distributed across many resources. In some cases, periodic data calls might temporarily alleviate the problem, but the information gathered quickly goes out of date. One key to a more comprehensive and effective inventory is to use application-aware tools that go beyond identifying low-level hardware and software; they recognize and characterize high-level enterprise applications and their complex dependencies. Another key is appropriate human investigation and analysis facilitated by the tools. This is required when users run complicated custom applications that are not well known to tool developers. For such cases, we describe methods including the examination of communication patterns between software modules, pattern matching, and name analysis. Applying such methods to several large enterprises successfully identified running applications and yielded previously unknown potential vulnerabilities to attack.

1. Introduction

In today's information technology (IT) environment, cyber defense is an increasingly important and pervasive issue.[1,2] Effective cyber defense depends on many factors, including perimeter and endpoint defense, intrusion detection and prevention, malware detection and interdiction, training to support safe practices, and others. Fundamental to all cyber defense activity is a knowledge of one's IT assets; to defend something, we must first know we have it.

The United States National Institute of Standards and Technology (NIST) has formulated a Cyber Security Framework[3] consisting of five core functions: Identify, Protect, Detect, Respond, and Recover. It is worth quoting the description of the Identify function in its entirety:

> "Develop an organizational understanding to manage cybersecurity risk to systems, people, assets, data, and capabilities. The activities in the Identify Function are foundational for

[1] U.S. Department of Homeland Security, 2018. U.S. Department of Homeland Security Cyber-Security Strategy. Washington, D.C.
https://www.dhs.gov/sites/default/files/publications/DHS-Cybersecurity-Strategy_1.pdf
[2] U. S. Department of Defense, 2015. The DoD Cyber Strategy. Washington, D. C.
https://www.defense.gov/Portals/1/features/2015/0415_cyber-strategy/Final_2015_DoD_CYBER_STRATEGY_for_web.pdf
[3] National Institute of Standards and Technology, 2018. Framework for Improving Critical Infrastructure Cybersecurity (Version 1.1). Gaithersburg, MD.
https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf

effective use of the Framework. Understanding the business context, the resources that support critical functions, and the related cybersecurity risks enables an organization to focus and prioritize its efforts, consistent with its risk management strategy and business needs. Examples of outcome Categories within this Function include: Asset Management; Business Environment; Governance; Risk Assessment; and Risk Management Strategy."

This paper focuses on one aspect of the Identify function—effectively identifying IT assets and characterizing their dependencies in large, complex enterprises.

It would seem that gathering information about our IT assets is straightforward. However, in a complex enterprise there are several reasons that the situation becomes very complicated. Among these are:

- multiple, distributed command centers
- multiple computing sites
- multiple programs, applications, systems
- a long history of software—based application development
- a large numbers of users

As an enterprise evolves over time, knowledge about the enterprise applications in the large data centers, in small racks of computers, on the desktops, and in the field becomes outdated and diffuse. The knowledge of how applications are implemented, what resources are required, and how they interact can become dispersed and disorganized as the developers and project managers change jobs, retire, or fail to document the systems properly. In the enterprises we studied, there were nearly 100,000 host computing devices and software assets in the millions. This illustrates the scale of the problem, and the need for automation.

If organizations always exercised proper configuration management and maintained perfect registries of necessary information, capturing all dependencies among the systems and applications, then cyber defense would be substantially easier. However, in the real world, the required information is generally not adequately maintained, particularly since applications can be very complex and distributed across many resources. In some cases, periodic data calls might temporarily alleviate the problem, but the information gathered through data calls is not especially accurate and quickly goes out of date

Better automation of the process of creating and maintaining an IT asset inventory that includes capturing the resources used by complex applications is possible through the use of tools we term application-aware discovery tools. A typical IT asset inventory tool will scan and report on the systems on an enterprise network and automatically discover and characterize all of the basic IT assets (hardware and software) found. Examples of such tools include Microsoft System Center Configuration Manager (SCCM), [4] and Tenable's Nessus Professional Vulnerability Scanner,[5] among others. In contrast to such tools, an *application-aware* IT inventory tool will also

[4] https://www.microsoft.com/en-us/cloud-platform/system-center-configuration-manager
[5] https://www.tenable.com/products/nessus/nessus-professional

report information about complex applications and map the basic assets that they use. A complex application consists of a collection of basic IT assets such as a web server, a database, a load balancer, etc.

There are several application-aware tools available in the commercial marketplace that can accomplish this task with varying degrees of manual intervention.[6,7,8,9] In general, such tools work well when the systems/applications being run consist mainly of popular, commercial software. The tool vendors supply libraries that recognize the well-known commercial software. But, when users are running complex custom or Government off-the-shelf (GOTS)[10] applications that are not well known to commercial tool developers, the situation becomes much more difficult, and requires human investigation and analysis. For such cases, we outline a method to characterize the applications that uses discovered information including the examination of communication patterns between software modules, pattern matching, and name analysis.

In this paper we will describe the capabilities of a typical application-aware discovery tool. We will then discuss application discovery and mapping using the information gathered by the tool. Lastly, we will make some recommendations on how an organization can use this type of analysis to improve its cyber defense as well as to satisfy other demands on IT organizations.

2. Application-Aware Discovery Tools

An application is defined as a collection of hardware and software organized for a specific function. An application may be government-off-the-shelf (GOTS) software and hardware, specifically configured commercial off-the-shelf (COTS) products, or a mix of both. A complex application may span several host computers and use various combinations of software products and custom software.

An IT asset inventory tool collects information about the hardware characteristics and software loaded onto the devices found in a network enclave. An enclave is defined as the networks and attached devices in a subset of the enterprise network that is under consideration. The inventory typically includes a detailed description of the hardware (host computers, network devices, workstations, laptops, printers, storage devices, and other equipment) and the software products and other lower-level software components (the installed software packages, file systems,

[6] BMC Discovery [http://www.bmc.com/it-solutions/discovery-dependency-mapping.html]
[7] IBM Tivoli Application Dependency Discovery Manager (TADDM) [https://www.ibm.com/support/knowledgecenter/en/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/welcome_page/kc_welcome-444.html]
[8] Micro Focus (former software division of Hewlett Packard Enterprise] Configuration Management System (CMS) Universal Discovery and CMDB [https://software.microfocus.com/en-us/products/configuration-management-system-database/overview
[9] ServiceNow Discovery [https://www.servicenow.com/products/discovery.html]
[10] For a discussion of issues related to GOTS software, see, e.g., ASD-NII, 2011, *Open Technology Development: Lessons Learned and Best Practices for Military Software*. Washington, DC: Report to the (then) Assistant Secretary of Defense (Networks & Information Integration) (NII) / DoD Chief Information Officer (CIO) and the Under Secretary of Defense for Acquisition, Technology, and Logistics (AT&L)).

processes, services, and other specific configuration data) found on the enclave. The tool will capture the inventory data and store it in an *inventory datastore*, typically organized as a database.

The inventory tools employ two basic methods: agent-based and agentless. Agent-based tools require the installation of a small piece of code (an agent) on every device that can report back to an inventory collection controller. The agentless tools do not require an agent to be installed and will scan all the devices found on a network and report back to the inventory collection controller. The differences in the two methods are primarily that an agent-based tool requires a priori identification of all devices that can be scanned, but can capture a history of processes that have run on a device over a period of time, while an agentless device can discover new or previously unknown devices that are present at the time of the scan, but is restricted to reporting on software elements that are running at the time of the scan. For discovery purposes, agentless schemes have an advantage.

Complex applications are difficult to track and manage. For example, a complex enterprise application may consist of web servers running web sites and web applications and employing a backend database, all of which may be running on different host computer servers with load balancers on the front-end. This knowledge of the application structure must be either supplied by someone familiar with the application, determined by inference mechanisms in the tool, determined by identifying the application components based on keyword searches, or determined by a combination of these methods.

Application discovery is the identification of the complex applications that are present in the enclave. The challenge of performing application discovery from IT inventory data is that often no specific knowledge of the locations or sometimes even the existence of the applications is available in the IT asset inventory data. Although many inventory tools are application-aware, in the sense that the tools provide functions and a visualization capability that assist an analyst in performing application discovery, they do not completely solve the application discovery problem.

Application mapping – a concept closely related to application discovery – is the identification of the IT resources, such as hosts and software products and services, which are being used by an application and graphically shown in diagram called a map. The information in application maps can be used for planning purposes and to assess the impact of cyberattacks on the specific resources, as well as determining the impact of hardware and software failures on users of the application. The goal of application mapping is to create a model (map) of an "application of interest" that describes the main parts of the application in terms of what software products and host computers are used to implement the application and also the relationships between these parts (e.g., "communicates with" or "is hosted on"). In general, application discovery is a simpler task than application mapping. However, sometimes one can create a map in the process of discovering an application. In an application-aware tool, the map can be converted into a pattern

or script that can be used by the tool to dynamically discover the application in the IT inventory data from future scans.[11]

Table 1 defines some terminology used by the BMC Discovery 11 tool that relates to the inventory collection and discovery process.[12]

Table 1: Useful Definitions

Term	Definition	Examples
Software Instance	COTS or GOTS software product that is identified and modeled by a pre-defined pattern	Microsoft SQL Server 2008 R2, Oracle Database Server 11g, Apache Web Server 2.2,
Candidate Software Instance	Process or service that is identified by the discovery tool as being a "candidate" software instance due to defined or observed communication patterns	
Software Component	Software module that is implemented in a Software Instance such as a web server or app server	Website, web app, FTP site, J2EE web module
Database	Named database on a database server Software Instance	Standard names in Microsoft SQL Servers include master, model, msdb, mssqlsystemresource, tempdb, and _Total
Host	General-purpose computer, can be physical or virtual	Server, desktop, laptop
Package	Installed piece of software, such as created through a Microsoft .msi file	Found in Windows Add/Remove Programs, or Linux archive of files that implement a function
Defined Communica-tions	Also called explicit or configured communications that are defined by parsing host configuration files	Symantec Backup Exec Remote Agent as client of a Symantec Backup Exec Server
Observed Communica-tions	Network connections based on commands for processes running during scan. Also called implicit communications	SolarWinds Network Performance Monitor connecting with Microsoft SQL Server
Pattern	Vendor-provided or custom script that searches for the defining characteristics of a software instance.	Parts of a signature that include process names, file names, module names.

The tools will capture information on most types of devices found on the enclave network that have an IP address, including servers, desktops, laptops, network devices (switches, routers), and printers. However, in order to conduct an inventory with an agentless system, there are several preparatory steps necessary. The tool needs to have knowledge of login credentials for each device to be scanned and interrogated. The interrogation methods employed are dependent on the OS of the device; for instance Windows systems are interrogated using different functions and credentials

[11] Introduction to Collaborative Application Modeling [https://docs.bmc.com/docs/display/DISCO113/Introduction+to+Collaborative+Application+Mapping].
[12] Inferred Nodes [https://docs.bmc.com/docs/display/DISCO113/Inferred+nodes]; Directly Discovered Nodes [https://docs.bmc.com/docs/display/DISCO113/DDD+nodes].

than for Linux systems. Additional credentials for SNMP devices, VMWare VCenter[13] instances, cloud applications, and other entities may also be used by the tool to acquire additional useful information.

Recent experience with hosting and deployment of inventory tools in reasonably complex enclaves has shown that a tool can be installed and operational in a day or two, while scans of the organization can be completed within a few days after credentials are obtained. Multiple scans should be organized to run at different times of the day; during working hours, nightly backups, or during special events (e.g., patch Tuesday) in order to maximize the likelihood of capturing applications that are not always active.

We call these results the basic inventory. The inventory obtained is immediately useful for identifying installed software, determining the variety of versions found, and tracking licensed software. There are many commercial tools that can provide basic IT inventory information that are commonly in use in an organization. However, in order to characterize the more complex applications, it is necessary to move into application-aware tools that can perform discovery and mapping.

3. Application Discovery

Three general cases arise when discovering applications: (1) the application is a well-known application (COTS or GOTS), (2) seed data is provided by knowledgeable persons who can supply meaningful keywords related to the elements of an application, and (3) no a priori information about the applications is available or present in the system.

An application-aware tool supports application discovery with several functions and capabilities. These include functions for searching the inventory datastore using keyword search and providing listings of the hardware and software elements that match the keyword search terms. It is possible to drill down into any of these elements to learn more about their details. The most important capabilities of various application-aware inventory tools that distinguish them from other tools that can provide basic inventory data include:

- Candidate software instances are created automatically from observed or defined communications. The tool has internal logic to determine which communications are likely to be interesting for application discovery and ignores many of the communication patterns that are probably not of interest (e.g., operating system calls).
- Database names are determined by interrogating the contents of database servers.
- Website and web application names are determined by interrogating the contents of web servers.

[13] VMWare, Inc. *VMware vCenterServer*. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/vCenter/vmware-vcenter-server-datasheet.pdf

- Details on virtual machines are provided by interrogating the controllers such as the VMware vCenters, or cloud interfaces.
- Visualizations depicting the relationships among software elements and hosting hardware elements are provided. The diagrams also display the observed and defined communications that have been detected in the inventory data.
- A pattern or scripting language can be used to create models of applications or software instances to be incorporated into future scans that will automatically discover applications and any changes to those applications.

Well Known Applications. Application-aware tools, in general, do not fully identify complex applications due to variations in their deployment, but rather have built-in patterns that can identify the core signature elements of certain well-known applications using the names of software instances, candidate software instances, database names, processes, services, packages, or executable modules (e.g., .exe files). The tools have the ability to pattern-match using the built-in or custom patterns to identify these elements in the inventory datastore. Once the presence of one or more signature elements is confirmed, it is highly likely that an instance of the application exists at this site and the application is considered to be "discovered."

For example, a pattern can be defined to discover SharePoint 2013 Server as a part of the well-known COTS application Microsoft SharePoint 2013. Tool vendors continually update their libraries of known software products, so their patterns are kept up to date. It is also possible to build custom patterns to recognize known GOTS, in-house developed software, or lesser-known COTS software instances and applications. Once custom patterns are created for enterprise applications and placed in the library, the applications can be automatically discovered as part of the future inventory scans.

Seed Data Search. Existing knowledge about potential applications can be extremely useful in discovering applications. Typically, seed data is supplied by someone knowledgeable about a particular application's structure or deployment. Types of seed data include whole or parts of software instance names, database names, websites, web applications, host names, package names, process names, service names, files name, and executable names.

The following kinds of information is desired:

1. Application name and acronym;
2. Application Architecture – (client-server, web services);
3. Application Tiers (external website, internal website, web app, database);
4. Environments (e.g., test, prod, dev), names associated with environment;
5. Names of host computers used to run the software;
6. What kind of application environment – (e.g., native executable, J2EE web app, .net, etc.);
7. Names of software products involved (e.g., SharePoint, WebSphere);

8. Names of databases, database servers (e.g., MSSQL, Oracle);
9. Names and types of web servers, web apps, websites;
10. Names of installed packages;
11. Names of main modules or executables;
12. Names of processes and services that should be available or active
13. IP address groupings or subnet groupings (e.g., IP class C address xxx.xx.40.* is Geographic Information Group).

It is not necessary to collect all of this information; however, the more seed data, the better. In some situations, it may be necessary to use other resources and documentation to better understand the architecture of an application. For example, some of the more complex commercial applications are described in documentation found on the web.

Often seed data can be extracted from external data sources such as an enterprise software asset inventory, which supplies generic information such as application name and acronym, or by searching the web for supplemental documentation. This can be used to create a list of keywords for use in a searching the datastore.

Unknown Application Search. In this case, there is little or no existing information on the presence of specific applications, so application discovery proceeds by investigation of the inventory datastore and deductive reasoning by an analyst. The discovery process typically involves an analyst searching the inventory data and piecing together clues that could be used as further keyword data to show the presence of an application. In the worst cases, no evidence can be found in the inventory data and the analyst must use outside sources to try to locate owners or other persons who can provide information on what applications the basic software elements are supporting.

4. A Discovery Process.

A method of discovering applications was developed based on the lessons learned during recent surveys of several complex enclaves. The method assumes that an IT asset inventory collected by an application-aware inventory tool has been performed so that the basic inventory data exists. This implies that a tool has been deployed on the enclaves of interest, that the appropriate credentials have been obtained, and that the tool has successfully scanned the enclave multiple times at different times of the day.

There are two types of tables employed: inventory tables and evidence tables. The Inventory Tables capture the basic inventory data collected by the tool and organized by categories: software instances, databases and other software components, hosts, packages, etc. The Evidence Tables contain well-known or previously discovered applications and search terms for inventory table items that are typically part of the applications. The evidence search terms are used to search the inventory data, and a match is a strong clue to the presence of an application at the site.

The discovery method is implemented in two phases:

- Phase I focuses on steps that do not require interaction with the inventory tool after it has generated and exported the basic inventory data.
- Phase II is best accomplished with the interactive use of the tool's functions and visualization capability.

The goal of the discovery method to identify one or more applications for each major inventory asset (e.g., hosts, database servers, web servers) in the enclave and to mark those assets as hosting an application. Other applications may be found based on the installed packages, processes, and services, but these can be more difficult to identify.

For most enclaves, the recommendation for the basic phase is that the analyst begin by considering the servers. The assumption is that most of the applications of interest will reside on servers. However, in our experience, this is not always the case, since we have observed applications running on workstations.

Construct the Inventory Tables. The inventory tables are constructed using inventory tool data via user queries developed on the inventory datastore. The results of the queries are information about the particular element of a category. In general there is one table of information for each category of element in the inventory (e.g., database names, server names, software instances, candidate software instances, etc.). A special table called the Host Overview Table (HOT) is created based on the basic inventory data. Table 2 presents the column headings of a HOT. The HOT contains one or more rows per server with separate rows for significant software instances and candidate software instances in separate rows of the table. The goal is to account for each entry in the HOT table by assigning it to an application. Initially the Apps column is blank but will be filled out as the process evolves. Here, the SolarWinds network performance monitoring application[14] was inferred from the evidence.

[14] https://www.solarwinds.com/

Table 2: Host Overview Table

Apps Inferred from Evidence	Server Role Code	Server Role	Host Type	Name	OS	Software Instance Type	Software Instance Name	Software Component	Database	Candidate Software Instance Type
SolarWinds	as	Application Server	Windows Server	*-as-008p	Microsoft Windows Server 2008 R2 Enterprise Version 6.1.7601 Build 7601					SolarWinds.Alerting.Service
SolarWinds	as	Application Server	Windows Server	*-as-008p	Microsoft Windows Server 2008 R2 Enterprise Version 6.1.7601 Build 7601	Microsoft IIS Webserver	IIS Webserver 7.5	SolarWinds NetPerfMon, / on SolarWinds NetPerfMon		
SolarWinds	as	Application Server	Windows Server	*-as-008p	Microsoft Windows Server 2008 R2 Enterprise Version 6.1.7601 Build 7601	SolarWinds Network Configuration Manager	SolarWinds NCM 7.3			
SolarWinds	as	Application Server	Windows Server	*-as-008p	Microsoft Windows Server 2008 R2 Enterprise Version 6.1.7601 Build 7601	SolarWinds Network Performance Monitor	SolarWinds NPM 11.5			
SolarWinds	as	Application Server	Windows Server	*-as-009p	Microsoft Windows Server 2008 R2 Standard Version 6.1.7601 Build 7601	Microsoft SQL Server	MSSQL 2008 SP4 *_SW_SQL		*_Solarwinds, SolarWindsOrion	

Build the Evidence Table. The Evidence tables were developed showing various types of evidence that indicate the presence of an application. The evidence can come from two sources: scans of similar enclaves and seed data from local application experts that can be used to infer the presence of certain applications. The evidence for an application is put into categories corresponding to the types of entities (i.e., software instances, software components, software packages, etc.). The keywords in the evidence tables are used to search the inventory tables for matches.

The first form of evidence comes from the application-aware tool and is derived from applications that were found from scans of other sites that have similar applications. If an application was found at the similar site, we can use its constituent parts in the various categories of evidence. For example, SolarWinds was found to often have software instances, candidate software instances, websites, web apps, and databases with the term “SolarWinds” in their names.

The second form of evidence is seed data on the applications that are known or strongly suspected of being present in the enclave. This assumes that persons or stakeholders are available who are somewhat knowledgeable about the applications in their domain. In many cases, local information on the applications hosted at an enclave may be available. This type of explicit information can be easily used in searches of the inventory tables.

Search with the Evidence Table. For each element in the Evidence Table, search the corresponding element type fields of the Inventory Tables (e.g., a particular software instance name in the Evidence Table is compared with all the software instances of the software instance inventory table). Using the inventory tables and the evidence tables, the process of discovering applications proceeds by searching for keyword matches between the tables. If a match is found, the corresponding entry (e.g., a software instance located on a particular host) in the HOT is

marked as covered. This is interpreted as indicating that the application associated with the evidence is present on the system. The HOT is also marked with this application.

In our experience, this semi-automated search methodology described above can readily discover a significant number of applications in a scanned enclave. As more evidence is gathered – meaning that more *patterns* of applications are established – the method will be able to discover even more applications.

Tips for Further Leveraging the Information in the Host Overview Table. As previously noted, the HOT consolidates key information about the servers in a scanned network enclave. This information can be used in various ways to gain insights into the role, or function, of a server and the applications that the server is likely to be hosting.

For example, the following insights were gained based on an examination of the HOT:

- **Server roles**: Server roles can often be determined by the server name, for example if the name contains variations of **application servers**, **web servers**, **database servers** (including Oracle and SQL servers), **file servers**, and **cluster servers**, then this indicates the type of application likely to be hosted. However, this is not always true as servers become multi-purposed over time.
- **Host names**: Similarly, there are servers named after the application they are hosting, e.g., Sharepoint Server, etc. Some servers had names denoting organizations. These names can help the analyst in identifying sources of information on the servers and the applications they host.
- **Proximity of IP addresses**: Sorting the HOT by IP address can help analysts discover groups of hosts related to each other by clusters of IP addresses.
- **Examination of host-to-host communication:** Many Discovery tools can report on host-to-host communication. If there is extensive communication between hosts, then this is another indicator of an application.

Phase II: Advanced Application Discovery. At the conclusion of Phase I, there will likely still be a number of servers with unknown applications (i.e., no applications have been marked for those servers). There are several further methods of investigating which applications are running on the hosts for the cases when an application is suspected but no evidence was found. In the advanced phase, the focus is on direct use of the application-aware tool functions to discover the applications. If Phase II is not employed, then the analyst is relegated to using the basic inventory data in the HOT to target the sources of the information. The analyst must track down the administrator or stakeholder to determine the owner of the server or the application, and then determine the application name and if possible, additional seed data.

For the remaining hosts in the HOT with no solid application matches, the inventory tool will be used to directly search the datastore and to use the visualization capability. In the tool, the analyst can select a host and examine the visualization for connections to other known applications

or candidate software instances. If there are connections (e.g., between a database server with a known application and a web server with no application), then these are possibly two parts of the same application. Various types of connections can be shown for differing relationships between hosts, software instances, and software components, such as “hosted by”, “communicates with”, or “contains.”

Validating the Model. The results of the discovery method should be shown to an application expert who is familiar with the applications to make sure that the identified applications are correct (assuming one is available). Many application names are ambiguous or are names of different products, and it is not difficult to mix up different applications, especially with keyword table searches.

If the application expert finds some errors of omission, then it is necessary to re-search the datastore or to rerun targeted scans when the missing application is known to be active. It is also possible that the application is implemented on a host that was not able to be scanned and the host must be configured to be scanned successfully so the application can be found.

Once an application has been discovered and validated, the evidence for the application can be entered in the Evidence Tables, creating Local Enclave Evidence Tables tailored for the enclave. The keywords could be forwarded to a central collection point to be entered into a global Evidence Table for use in future application discovery efforts.

When is the Discovery Complete? The discovery effort is technically complete when all the hosts and all of the major software assets (e.g., software instances, software components, interesting packages, interesting processes and services) on those hosts have been explained as parts of applications or as not of interest. In practice, this situation will almost never be attainable for many reasons. Some reasons for not completing the list include:

- Assets may be rarely used parts of applications or not of interest.
- Obsolete, leftover assets may be present.
- It may be impossible to locate knowledgeable stakeholders.
- The configurations may have changed significantly from what is noted in the documentation.
- Key parts of the application were not running during scans.
- Hosts containing key parts of the application were not successfully scanned (i.e., the host remains on the unable-to-access list).

Despite these issues, the analyst should make an educated decision on whether all of the applications of interest on a host have been identified, even if unidentified assets remain on that host. When the analyst decides that identification of applications on the host is reasonably complete, the host can be removed from further consideration (e.g., marking the HOT table), and the analyst can concentrate efforts on the remaining hosts.

A continuous process of periodically conducting application discovery should be maintained to improve the coverage, find new applications, and eventually eliminate software that is not being used.

5. Application Mapping

It may be desired to more fully characterize an application beyond just discovering its presence at a site. An application may have several important constituent parts and span several hosts. For purposes of application rationalization, application migration, and cyber defense, decision makers may need information on the architecture of an application. An *application map* is defined as an architectural view of an application instance (i.e., a specific implementation of an application) showing how its parts map onto the underlying hardware and software infrastructure captured by the inventory process. A notional view of a mix of COTS and GOTS applications mapped onto the software resources and the hardware infrastructure is shown in Figure 1.

If application maps are built for the identified applications, then the patterns derived from the maps can be stored and the application-aware tool can automatically identify applications in the inventory, reducing the effort needed to maintain an application inventory or to develop the inventory at other enterprise sites. Further, the application maps can be used to signal changes in the implementations of these applications as alerts for improved configuration management.

In the course of performing the application discovery process, much of the information needed to accomplish application mapping is typically collected. However, discovery stops when sufficient evidence for the presence of the application is determined, while mapping will proceed to find all resources and dependencies of an application.

Application mapping is useful for understanding the dependency relations among the parts of a complex application, as well as any relationships among the applications. This information helps in satisfying mission assurance needs by showing how an application is impacted by outages faults, or cyber attacks. It is also useful in determining application resource usage for migration to the cloud, and for developing programmatic data for planning and budgeting. The maps can be used to identify the main parts of an application and which hardware resources are used to implement the application, and to estimate the computing, power and space requirements of an application if it is to be migrated to another data center or the cloud. Although not illustrated in Figure 1, the application map can also include the communication relationships among an application's parts or among different applications.

The types of software elements in an application map are typically software instances (web servers, database servers, packaged software products), software components (websites, web apps), and databases. The level of detail or granularity of the application maps depends on the needs of the organization and how it wants to manage applications. If fine-grained configuration management is intended or detailed engineering data is required, then the maps should represent dependencies at the lowest practicable level.

Application Pattern. An application map can also be abstracted into a "pattern" that can be used to (1) discover application instances that "match" the pattern and (2) build application maps for the matching application instances. The pattern is sometimes referred to as an application map. However, to avoid confusion, the term *application pattern* is often used to refer to the abstraction.

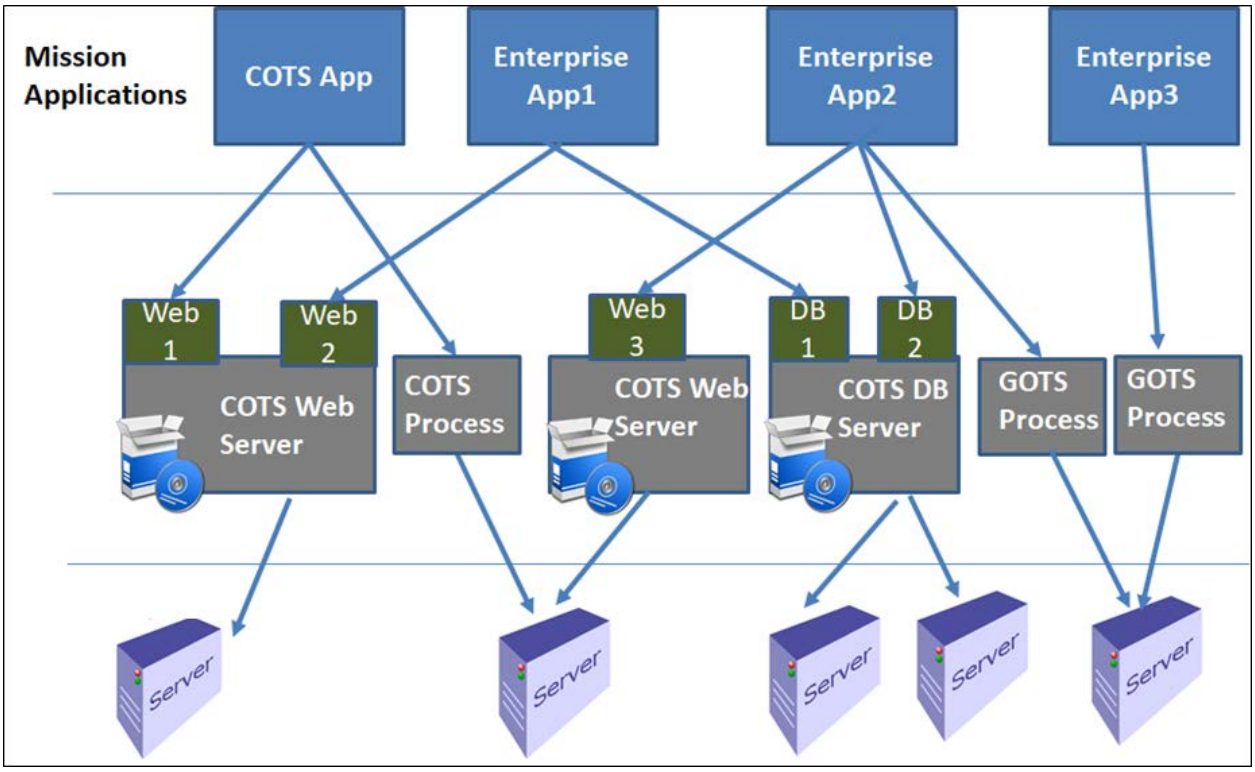

Figure 1: Applications Mapped to Hardware and Software Resources

Mapping Capabilities of Tools. Inventory tools can provide functionality to support application mapping; however, the process typically requires input and direction from an analyst. The analyst can manually identify a collection of software as an application instance by creating a pattern of the application instance in terms of its constituent software instances, candidate software instances, software components, and databases. All of these elements as well as many others are the items collected by application-aware inventory tools during scanning. The application map can also show the relationships between the hosts and other infrastructure elements (e.g., clusters, virtual machines, switches, subnets) on which the main software elements depend.

Visualizations. Visualization capabilities are more useful in mapping than discovery, since the diagram can provide a compact and easily understandable view of the map. An analyst can sometimes quickly determine extraneous elements in the diagram by viewing the map. However, if the application is too complex, then the diagram can become unwieldy and will need to be partitioned.

Many of the inventory tools tool have the ability for the analyst to create maps starting from diagrams of any elements (host, software instance, etc.) and include the relationships between the elements such as communicates-with, hosted-on, or contains. The diagram can be edited to add or remove elements and their relationships will be automatically updated when the diagram is modified. In addition, any additional candidate software instances that the tool determines are related will also be displayed. See Figure 2 for an example of a visualization produced by an analyst working with the tool to create a map of the SolarWinds application. Another useful capability allows the analyst to drill down into the elements in a diagram to display details such as database names, websites, or processes on hosts that may be part of an application. These features have increased analyst productivity in mapping.

With the type of visualization shown in Figure 2, the impact of a component failure—whether from internal processes or a cyber attack—is made apparent, and can be analyzed. The visualization also has the potential to reveal structural vulnerabilities and suggest ways to improve security and robustness. For example, an analyst may obtain a better understanding of all the communication entities and be able to discern anomalous patterns.

There is a close relationship between the discovery process and the application mapping process. If key pieces of evidence are found that support the discovery of an application, then that piece of evidence can be used to start a map. If the tool has the ability to start a map with any IT asset, then the evidence can be placed on a map and the tool will automatically bring related assets into the diagram (e.g., those assets that host this software or communicate with it). In many cases, this is enough information to create substantial portions of the application map.

Candidate software instances also play a large role in application mapping. The tool will automatically include the candidate software instances in the diagram, along with their dependencies. If supported by the tool, the analyst can easily edit the diagram to remove any candidates or other assets not of interest until a map of the application remains. This can be converted into a pattern.

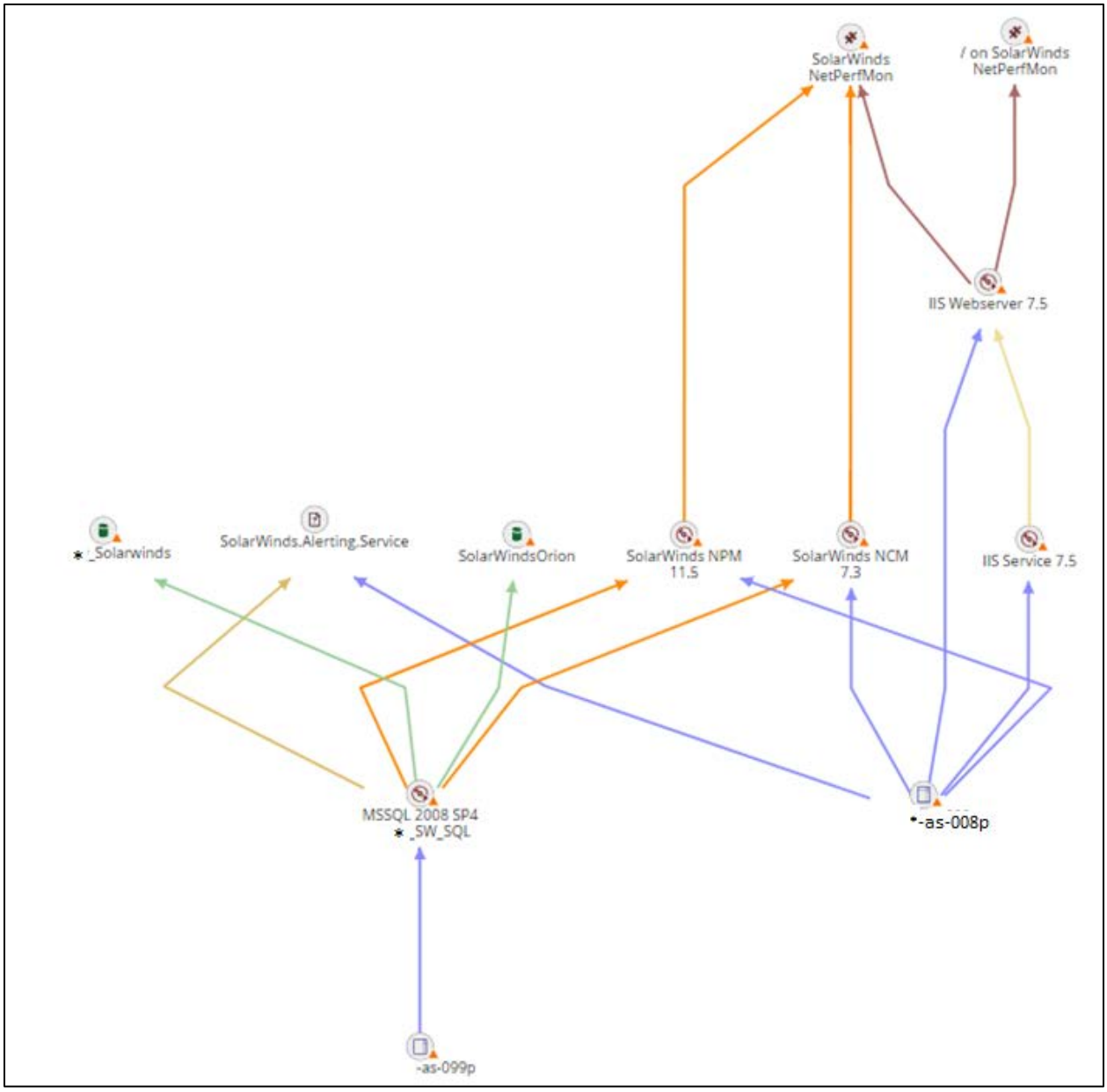

Figure 2: Visualization of SolarWinds in Discovery Tool

6. Application Inventory Results and Cyber Defense

Data was collected using application-aware inventory tools at several large enterprise enclaves. Besides providing valuable inventory data to each enclave, the data was combined into cross-enclave tables and the data was examined for trends. Nearly 100,000 host computing devices were scanned, revealing software instances and packages in the millions. This illustrates the scale of the problem, and the need for automation. Several important observations that directly impact security and operations were documented including:

- A large number of versions of software products were in use. This increases overall system complexity. It makes security patches difficult to implement consistently, and thus potentially exposes more assets to exploitation.
- Many software products were past their end-of-extended-support, or end-of-support, creating possible security issues. In the enterprises analyzed, over half of the products were past support.
- Instances of SQL servers, websites, and web applications were deployed on workstations (desktops and laptops), indicating production software on end user devices, whose security measures are not generally as strong as those of servers.
- Instances of non-enterprise-grade software (e.g., Microsoft SQL Server Express) as components of enterprise applications.
- Non-standardized names were used for servers and network devices. Use of default names for software and database names was common (e.g., Microsoft SQL Server instances were called MSSQLSERVER). This is not helpful for configuration management and application discovery.
- Adverse security practices, such as factory default passwords not disabled on a variety of IT devices, were observed.

Based on the analysis and conclusions, several recommendations are given to achieve an enterprise-wide process for collecting an accurate IT asset inventory that provides benefits to individual IT enclave operations, as well as for higher-level enterprise planning activities.

An enterprise should deploy an integrated, automated, application-aware IT inventory tool, along with a continuing process to collect inventory data and perform application discovery at each enclave. The results show that deploying an automated application-aware IT inventory tool confers significant benefits in terms of accuracy and timeliness of inventory data. A tool with analysis and visualization capabilities that integrates basic inventory data, network data, and detailed information on software components can assist in application discovery and mapping and present a more coherent view of the network and systems. This study has demonstrated that it is feasible for enclave IT administrators to securely deploy and operate an application-aware IT inventory tool with negligible impact on network and system performance.

The application-aware IT inventory tool should be used for enterprise-wide application discovery efforts; however, due to the labor involved in application mapping, the enterprise should initially undertake application mapping for only its highest priority applications or as needed for application migration or rationalization. The enterprise can use an application-aware IT inventory tool to assist in application discovery using the method described earlier, although it is not a fully automatable procedure. The enterprise should implement an ongoing process to identify and capture the applications at each enclave in order to maintain an accurate listing.

This study has shown that a larger commitment of time and resources would be necessary to map all or most of the applications found across all enclaves of a large enterprise. While the benefits are significant, particularly in understanding how software or hardware outages impact applications, there may not be an urgent need to provide this for all applications. Data center consolidation and migration to the cloud are stimulating application mapping. The application mapping could continue at a slightly slower pace (than discovery), and in priority order until all applications are mapped. Any future software development requirements mandate and support application mapping, by, for example, requiring software ID tags.[15]

There should be a standard definition of the applications of interest for tracking and reporting purposes, as well as on a set of data items that could be generated by an application-aware IT inventory tool. The tool should be set up to automatically feed the summary inventory data to enterprise authoritative data sources so that functional users, and other interested activities such as Property Accountability, Software Asset Management, Budgeting, Financial Improvement and Audit Readiness (FIAR), Data Center Consolidation, and governance boards have access to this information. Automatic and periodic reporting will reduce the burden on the enclave IT administrator staff to collect this information and will increase the accuracy, timeliness, and completeness of the information in the authoritative data sources.

An enterprise should reduce the complexity of its IT asset inventory to reduce the attack surface and to reduce operational costs. It should undertake a number of steps to improve the security and operational posture of its enclaves. Since the complexity of the systems impacts both the attack surface and the operational costs, actions such as reducing the number of product versions, enforcing standard naming conventions, and improving procedures for managing UNIX/Linux systems will reduce costs and streamline operations. Standardizing on an application-aware IT inventory tool and eliminating the other redundant tools will also contribute to cost savings and operational benefits.

Much of the ability to quickly respond to these issues depends on the integration of the various types of information on networks, systems, and software into a coherent datastore. The operator/analyst needs to be able to flexibly query the inventory datastore and to visualize the relationships among the types of data. In comparison with more manual methods that use files and spreadsheets to integrate and process the data from multiple tools, an integrated system offers many advantages. It enables all stakeholders to manage from the same common operating picture. This visibility helps to correct errors, further strengthening the data.

The inventory data are vital for understanding and defending the networks and systems. The above recommendations describe a future-state vision that incorporates the use of an application-aware IT inventory tool across the enterprise, mitigating several important gaps observed in the current IT operational environment. The operations can be improved in several ways. By exposing

[15] ISO/IEC 19770-2:2015. Information Technology—Software Asset Management, Part 2: Software Identification Tag. Geneva, Switzerland: International Organization for Standardization.

the complexity of the system in terms of the many versions of software that are deployed, actions can be taken to reduce the complexity.

Application maps help isolate the effects of network or system failures, determine their impact on missions, and enable corrective actions. This applies both to internal failures, and ones arising from cyber attacks. Configuration management can be improved and enforced by noting any changes that occur between scans and making sure that these were authorized.

More effective scanning and mapping of applications can lead to improved IT operations in general and better cyber defense in particular. As experience is gained, a virtuous cycle can be established as shown in Figure 3, in which improved operations also lead in turn to better scanning and mapping.

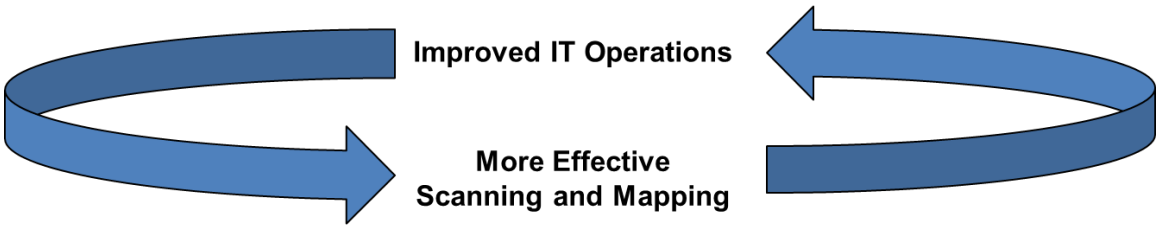

Figure 3: Virtuous Cycle enabled by Local Control of Inventory Tool

Disclaimer

The views, opinions, and findings are those of the authors, and should not be construed as representing the official position of the Institute for Defense Analyses, the United States Government or any of its agencies.