Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

# Extranet Access Management (EAM)

This document will give a description of the problems EAM software solves for E-business security. The Scope of this document will be: Overview of EAM Architecture, EAM Security, EAM a standard security model, and How does EAM integrate with JAVA.

Copyright SANS Institute
Author Retains Full Rights

AD

**Extranet Access Management (EAM)**
Nev Sealey
January 28, 2002

## Executive Summary

As businesses develop Internet applications they are increasing their exposure to external security vulnerabilities from the Internet. In many companies the security of web applications is the responsibility of each application development team. Today, every new web-based application potentially brings a new and different way to manage application access. Because each application team must develop and implement its own security processes, the quality of security varies with each application development team. Often there is not a common way to monitor application development security for consistency or an effective method for administering security for heterogeneous operating systems that run the applications.

Web applications are a potential access point to corporate information assets by anyone who has access to the Internet. Security controls that provide strong user authentication (public/private key technology) access and detection controls are critical for controlling access from the Internet to the company's computer environment and business data. Typically custom security implementations are costly to develop and maintain. This is due to several factors security routines/procedures have to be created for every application scenario defined (and not clearly defined). Technical Documentation is not properly maintained therefore knowledge transfer between programmers is incomplete or lost therefore E-business applications security quality varies with the developer who writes the code.

Can one application centrally administer various user stores? Can the same application provide industry standard controls and administrative tools to manage web application security access? What about developing alerts and reports on current and potential exposures, management audits. Can a company accomplish this without having a staff specialized in the niche discipline of application security development in disciplines like JAAS, JCE and JSSE (1) The answer is yes; Extranet Access Management (EAM) (2) software can provide these things. EAM is not a "silver bullet" for E-business security challenges, but I would suggest that one look at EAM software as a foundation piece of building defenses against malicious attacks.

## Scope

This document will give a further description of the problems EAM software solves for E-business security. The Scope of this document will be:

- Overview of EAM Architecture
- EAM Security
- EAM a standard security model
- How does EAM integrate with JAVA

**Overview of EAM Architecture**

EAM applications allows an organization to assign authentication schemes, define and Manage authorizations to specific resources, and create rules and policies to implement these Authorizations. From an architectural perspective EAM product typically consists of two main components, a Policy Server and the EAM Agents. The Policy Server provides policy management, authentication, authorization, and accounting Services to Web-based applications. The EAM Agents include the following:

- Web Agents— A Web Agent is integrated with a standard Web server and enables EAM applications to Manage access to the Web server and its content.
- Affiliate Agents— An Affiliate Agent runs on the Web server of an affiliate site (business partner) and provides single sign-on and personalized content for users of the affiliate site.
- Application Server Agents— An Application Server Agent is integrated with the HTTP server of an application server (or a built-in Web server) to secure Java 2 Platform, Enterprise Edition (J2EE) components.

EAM products usually provide a set of developer Application Programming Interfaces (APIs), which allow an organization to extend the EAM product capabilities to meet specific needs of an organization. Other features typically include Delegated Administration Services (DAS) which provide an integrated set of delegated user administration tools. DAS allows an organization to extend administration of users and their profiles and entitlements to partner and affiliate site so those users can be managed independently. DAS allows administrators to manage organizations, users, and user groups and offers a number of customization options.

The key services offered by EAM products are as follows:

- Multiple authentication's scheme, from traditional memorized passwords to Public-key certificates.
- Single sign-on, a mechanism that allows each user to authenticate only once to access applications across multiple Web servers and (sometimes) sites.
- Registration, sometimes including dynamic registration, provision of public-key certificates, and biometrics enrollment.
- Rights management and enforcement services embracing both "traditional" authorizations based on Access Control Lists (ACLs), and other entitlements, based on business rules.
- Administrative interfaces that support both centralized and local user management.
- Audit and monitoring services, providing event logging and reporting to show relevant security events and to provide information about the usage habits of different users.
- Ability to use Lightweight Directory Access Protocol (LDAP) and/or Relational Database Management Systems (RDBMS) to store user attributes and access information
- Web agents which would allow integration with standard Web server that would enable the EAM product to manage access to the Web server and its content
- An Application server agent which is integrated with the HTTP server of an application server (or a built-in Web server) to secure J2EE components
- Developer Application Program Interface (API) to extend EAM software capabilities if necessary
- APIs support C,C++, and Java-language bindings
- Auditing reports, predefined reports and Crystal reports functionality to monitor activity
- Directory mapping functionality (specifies that users will be authenticated using one directory, but authorized to use another directory)
- Extensible Markup Language (XML) schema to enable integration of access rules of different EAM products

**EAM Security**

The focus is on the four "A"s: authentication, authorization, administration, and auditing. Authentication and authorization services benefit the organization by decreasing the risks of unauthorized access to Web pages, applications, and so on, and its auditing service provides full accountability. EAM provides a single point of administration for multiple Web-enabled applications and a way of delegating administration to different organizational units (or even external organizational units) so that an organization can spread the administrative overheads.

Authentication

When looking at an EAM product authentication method combinations ("certificate and password") will be useful when an organization requires stronger security for a specific set of resources. Support for alternative methods ("certificate or password") will be useful when an organization wants to deploy public-key certificates gradually: a certificate can be used for authentication if one is installed; otherwise, it will fall back to regular password authentication. Forms-based authentication can exploit user attributes (say, social security number [SSN] or mother's maiden name) stored in the user directory with no application coding changes required. Most EAM products support authentication via public-key certificates from VeriSign, Microsoft, Netscape, Entrust, RSA Security's Keon (including Xcert Sentry), and Baltimore.

Authorization

EAM products supports a variety of ways to restrict access for specific users to only certain resources that improve the organization's security and reduce programming overheads. These methods are as follows:

- Full URL access control (for pages, applications, etc.)

- Fine-grained access control (sub-page-level control)

- Content access control (providing only selected portions of a database to a specific user)

Full URL access control lets an administrator group similarly name resources by using names that include wild cards and/or regular expressions, reducing administrative overhead. Sub-page-level access control gives an organization an easy way to create Web pages that are personalized to the specific needs and authorizations of each user. This greatly simplifies and standardizes programming. Content access control allows an organization to restrict access for each user or user group to a specific subset of the protected data. So for example a response will return an SQL search modifier string to the application, based on the authorizations or attributes of each user. The application includes that search modifier in the SQL query statement, automatically returning only the data that this user is entitled to see. This significantly reduces administrative and programming overheads.

Administration

An EAM product allows central as well as delegated administration of all resources, realms, actions, responses, rules, and policies. A super administrator can delegate user administration responsibilities to organizational administrators within, say, internal departments, external partners, Application Service Provider (ASP) subscribers, and e-commerce trading partners. For more fine-grained administration, users can be granted privileges to modify their own attributes. Delegation can be done to as many levels of administrators as desired. DAS allows an organization to gain all the benefits normally associated with delegated administration:

- Each administrator best knows the needs of users within the organizational unit, improving security
- The workload of central administrators is reduced, lowering costs.
- Central administrators do not become an administration bottleneck, improving service levels.

DAS also provides a Workflow API, so that other customer-defined events can be triggered by specific user management events, such as a new user being created, further reducing administrative overheads.

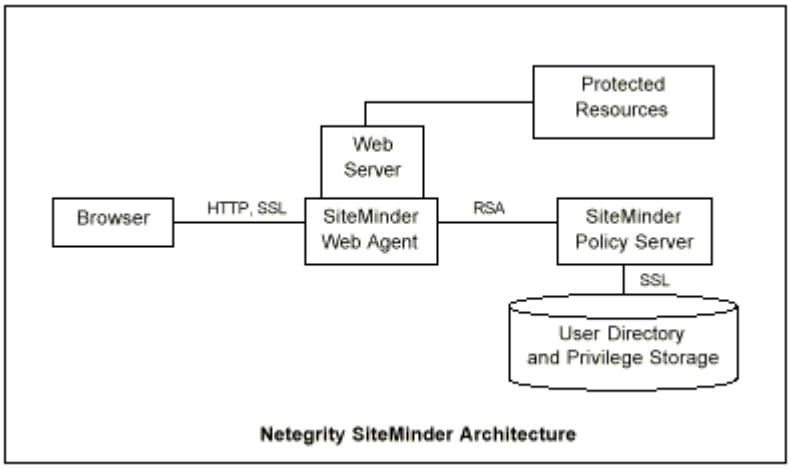
Auditing

A Policy Server generates a log file, which can be either a flat file or an Open Database Connectivity (ODBC) database. The log file contains comprehensive auditing information about the events that occurred within the system— user authentication, access to protected resources, and administrative events. An organization can use a number of pre-defined reports, or create its own using Crystal Reports, to monitor activity and so maintain full accountability of all users and administrators. This is significant because this means central auditing reporting occurs no matter what platform, information store, or programming language the E-business application(s) are using.

**EAM a standard security model**

J2EE builds upon the Java 2 Platform, Standard Edition, and adds full support for servlets, JavaServer

Pages TM (JSP), Enterprise JavaBeans TM (EJB) components, and XML technology for business-to-business transactions. J2EE components are deployed to an application server, which hosts them in containers for access by Java Clients. One of the many benefits of J2EE components is that they can be ported to any Vendor's application server provided it complies with Sun's J2EE API and test specifications. We will come back to Java components and EAM later.



Netegrity SiteMinder Architecture

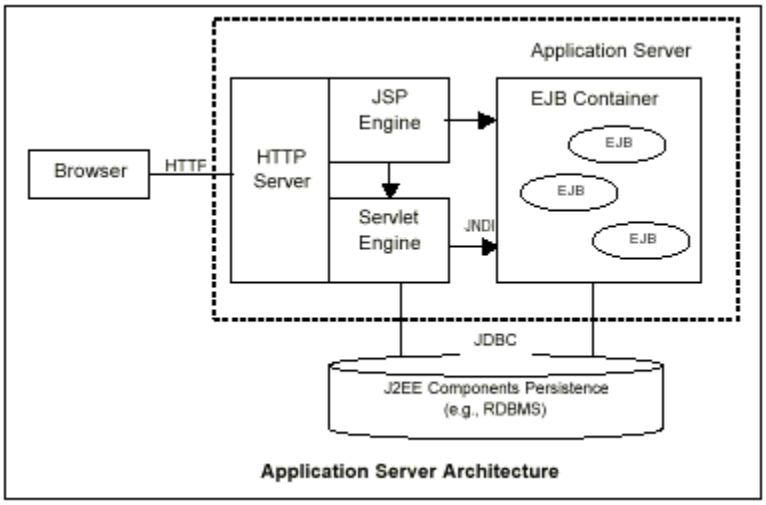
EAM software can protect entry to a resource (application or file) by intercepting end-user requests sent to the Web server, which in turn makes calls to EAM Policy Server. This is significant when you consider that most malicious attacks on Web servers are Buffer overflow based attacks.

An EAM product called Siteminder illustrates this protection above (3). In order to secure more fine-grained objects such as servlets, JSPs, or EJB components, EAM products use Application Server Agents. I will use SiteMinder (4) and JAVA to describe in greater detail how EAM can be the foundation piece in building E-business security at the application level.

**How EAM integrates with JAVA**

Java is currently considered by many to be the best way to develop applications for "the web" the appeal is its portability to different hardware platforms (client and server). A brief description of Java components is necessary to understand how EAM software secures Java applications. I will use Netegrity's Siteminder as an example of how an EAM application integrates with Java components.

Java Background




The figure below (5) shows application server architecture for HTTP clients.

Servlets

Servlets are Java programs that have primarily been used with Web servers. With the emergence of J2EE application servers, servlets typically run on application servers instead of the Web servers. Java servlets are becoming increasingly popular as an alternative to CGI programs. The biggest difference between the two is that a Java applet is persistent. This means that once it is started, it stays in memory and can fulfill multiple requests. In contrast, typically a CGI program disappears once it has fulfilled a request. The persistence of Java applets makes them faster because there's no wasted time in resetting up and tearing down the process. This persistence of Java applets also increase’s the risk of sensitive and proprietary information getting into the hands of competitors.

JavaServer Pages

Java Server Pages (JSP) are a server-side technology; Java server pages are an extension to the Java servlet technology that was developed by Sun as an alternative to Microsoft's Active Server Pages (ASPs). JSPs have dynamic scripting capability that works in tandem with HTML code, separating the page logic from the static elements -- the actual design and display of the page. Embedded in the HTML page, the Java source code and its extensions help make the HTML more functional, being used in dynamic database queries, for example. JSPs are not restricted to any specific platform or server.

Enterprise JavaBeans

Enterprise JavaBeans (EJB) is a Java API developed by Sun Microsystems that defines a component architecture for multi-tier client/server systems. EJB systems allow developers to focus on the actual business architecture of the model, rather than worry about endless amounts of programming and coding needed to connect all the working parts. This task is left to EJB server vendors. Developers just design (or purchase) the needed EJB components and arrange them on the server. Because EJB systems are written in Java, they are platform independent. Being object oriented, they can be implemented into existed systems with little or no recompiling and configuring.

Java Naming Directory Interface

The Java Naming and Directory InterfaceTM (JNDI) is a standard extension to the Java platform, providing Java technology-enabled applications with a unified interface to multiple naming and directory services in the enterprise. As part of the Java Enterprise API set, JNDI enables seamless connectivity to heterogeneous enterprise naming and directory services. Developers can now build powerful and portable directory-enabled applications using this industry standard.

Java Database Connectivity

Short for Java Database Connectivity (JDBC), a Java API that enables Java programs to execute SQL statements. This allows Java programs to interact with any SQL-compliant database. Since nearly all relational database management systems (DBMSs) support SQL, and because Java itself runs on most platforms, JDBC makes it possible to write a single database application that can run on different platforms and interact with different DBMSs. JDBC is similar to ODBC, but is designed specifically for Java programs, whereas ODBC is language-independent. JDBC was developed by JavaSoft, a subsidiary of Sun Microsystems.

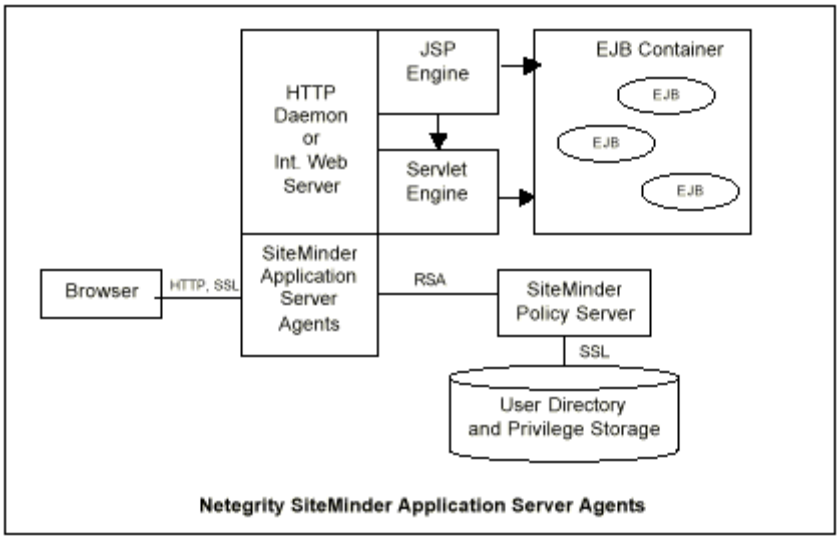
Java 2 Platform Enterprise Edition

Short for Java 2 Platform Enterprise Edition (J2EE). J2EE is a platform-independent, Java -centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multitiered, Web-based applications.

Anatomy of a Secure J2EE Application

In an ideal component architecture, the independent bean provider (or business-component vendor) creates the home interface, the remote interface, and the bean classes for a business component. The In-house application developer creates EJB clients from the EJB Object. In this way, the application developer can use existing, industry-standard business components, in-house business components, or newly developed components, and assemble them into what will become a distributed J2EE-based application.

The next step is for the application server administrator to deploy packaged business components that match the business need. So for example an online banking application may specify very strict security requirements on some components. Advanced application servers software (i.e. BEA WebLogic, IBM WebSphere) available today include security features, they fail to fully support enterprise-wide security. Such as single sign-on to an entire Web site (or portal) concurrent use of heterogeneous user directories (LDAP directories, Microsoft ADS or RDMS), Public Key Infrastructures (PKI) applications, auditing and reporting, etc. which are important security concerns.




EAM Application Server Agents (EASA) are a set of servlets that communicate with the EAM Policy Server via the EAM Agent API. EASAs are to J2EE applications what EAM Web Agents are to general-purpose Web-server resources. Instead of interfacing with a Web Server as Web Agents do. EASAs interface with an application server's HTTP daemon (an HTTP server process, or a full-fledged Web server integrated with the application server). Figure above illustrates how SiteMinder (6) secures application-server resources.

SiteMinder Application Server Agents are designed to protect fine-grained resources hosted in an application server, such as servlets, JavaServer Pages, and EJB components, by superseding the native application server's security functionality.

This implementation allows the invocation of all three types of agents simultaneously (i.e., for each class you can define what type of agent you want to support). In order to allow you to "SiteMinder-enable" J2EE components hosted by your application server. As with other EAM applications SiteMinder provides a set of APIs (servlet class framework extending your application server's classes, classes replacing the EJB initial context factory provided by the application server.

Java servlets (or Java Server Pages compiled to servlets) run in an application server process. Basically, the end-user sends a request from a Web browser to the application server's HTTP deamon (or integrated Web server). An application server proxy (plug-in) communicates with the application server's engine which uses the application server's Java Virtual Machine (JVM) to interpret compiled servlets. The results of the servlet process are sent back to the end-user via the application server's HTTP daemon. With SiteMinder, a call to a servlet is intercepted by the SiteMinder Java Servlet Agent, which provides an SiteMinder-enabled version of the servlet class extending the application server's class. (The application server's servlet engine has to be configured to use that class). Responses for the Java Servlet Agent are the same as for the existing Web Agent. This relationship between the EAM product, in this case the SiteMinder –enabled servlet and the application server servlet allows organization to build a standard security infrastructure. This is significant because developers are not limited by the application server security servlets.

An application server invoker is a servlet that calls other servlets by class name. Servlets are usually mapped to a particular virtual directory under the Web server. So for example pointing your browser to http://www.sans.com/servlet/com/sans/MainServlet (this is not an actual servlet call) makes a call to the invoker servlet, which then calls the service() method of the servlets it invokes. EAM or SiteMinder server-specific invokers are used to supersede the default application-server invokers so that SiteMinder functionality can be provided in the application server.

This model allows servlets to handle HTTP requests for many types of content, not just servlets. The model also allows for invocation based on a URL as well as MIME type (the format of the data returned by the servlet is defined by a MIME content type). The EAM Java Servlet Agent provides an application server-specific invoker that extends the classes provided by the application server thus allowing its resources to be protected by SiteMinder.

With existing Web Agents, EAM products can protect an entry to a resource (e.g., an application, a file, etc.) by intercepting end-user requests sent to the Web server, which in turn makes calls to the EAM Policy Server, the repository for user authorization and authentication information. Granularity of such security is at the resource level.

**Final thoughts**

I attempted to outline the benefits of EAM software. The question will come up (usually after an application is developed) do we build or buy security package? Try to point out that security is more complicated than it appears. There are some questions to consider if they wish to build an enterprise security solution for E-business applications.

- If your user base grows to the tens or thousands or in today's business climate decreases by a large number. Would a simple database table be sufficient?
- How does one secure the storage mechanism that holds sensitive information
- How do you secure the transmission of user data as it is passed around from the storage mechanism to the engine that will do the authorization/access evaluation on the web application server? What cipher do we use? Do we care?
- How do you maintain a session state without continual re-authentication if the user makes multiple requests to resources that span more than one data store or web application?
- Are proxy user ids acceptable? If not how would we get information from heterogeneous data stores without logging in multiple times?
- How often do we review the security controls that seemed appropriate yesterday but may not protect us today, who is responsible for keeping current in this niche disciplines (i.e. JAAS, JCE and JSSE)?
- Is forensic analysis important?
- How do we administer activity? Should we audit adminstration activity?
- Will we set up an administrative tool that allows allows you to delegate administration? How will we develop policy?
- Who is ultimately responsible for E-Business security breeches?

If one goes through the exercise I described above and still feels they rather build there own application keep in mind the Return of investment (ROI) (7) analysis is necessary part of the decision making process, some things that should be considered:

- Up front expenses (design, development, testing, deployment)
- Recurring Expenses (Component application server updates, J2EE, CORBA, and so on)
- Cost to integrate additional applications
- Ongoing product support (8)

If a software solution is available from an external "packaged application vendor"; it should be strongly considered. If that software solution is in an area of expertise different from the core expertise of your company one should pursue the software solution.

**Sources**

1. "Directions in Java Security." 23 Jan 2002.
URL: http://Java.sun.com/security (1/25/02).

2. Allen, Ant. "Extranet Access Management Perspective." 7 June 2001.
URL: http://www.gartner.com Article Number DPRO-90488 (1/20/02).

3. "Securing Enterprise Java Components with Siteminder." June 2000.
URL: http://members.netegrity.com/access/files/securing_enterprise_java_components_with_sm.pdf (1/23/02).

4. Allen, Ant. "Netegrity Siteminder Extranet Access Management Product." 29 August 2001.
URL: http://www.gartner.com Article Number DPRO-100176 (1/20/02).

5. "Securing Enterprise Java Components with Siteminder." June 2000.
URL: http://members.netegrity.com/access/files/securing_enterprise_java_components_with_sm.pdf (1/23/02).

6. "Securing Enterprise Java Components with Siteminder." June 2000.
URL: http://members.netegrity.com/access/files/securing_enterprise_java_components_with_sm.pdf (1/23/02).

7. Guptill Bruce, Terhune Alyse. "Managing ROI for Extranet Projects." 23 March 1999.

URL: http://www.gartner.com Article Number G-07-8053 (1/21/02).

8. "Chronology of Security Related Bugs and Issues." 23 Jan 2002.
URL: http://java.sun.com/sfaq/chronology.html (1/23/2002).

Last Updated: September 5th, 2015

# Upcoming SANS Training

**Click Here for a full list of all Upcoming SANS Events by Location**

| | | | |
|---|---|---|---|
| SANS Crystal City 2015 | Crystal City, VAUS | Sep 08, 2015 - Sep 13, 2015 | Live Event |
| SANS Network Security 2015 | Las Vegas, NVUS | Sep 12, 2015 - Sep 21, 2015 | Live Event |
| SANS Seoul 2015 | Seoul, KR | Sep 14, 2015 - Sep 19, 2015 | Live Event |
| SANS Tallinn 2015 | Tallinn, EE | Sep 21, 2015 - Sep 26, 2015 | Live Event |
| SANS Baltimore 2015 | Baltimore, MDUS | Sep 21, 2015 - Sep 26, 2015 | Live Event |
| Data Breach Investigation Summit & Training | Dallas, TXUS | Sep 21, 2015 - Sep 26, 2015 | Live Event |
| SANS Perth 2015 | Perth, AU | Sep 21, 2015 - Sep 26, 2015 | Live Event |
| SANS ICS Amsterdam 2015 | Amsterdam, NL | Sep 22, 2015 - Sep 28, 2015 | Live Event |
| SANS Bangalore 2015 | Bangalore, IN | Sep 28, 2015 - Oct 17, 2015 | Live Event |
| SANS Seattle 2015 | Seattle, WAUS | Oct 05, 2015 - Oct 10, 2015 | Live Event |
| SANS DFIR Prague 2015 | Prague, CZ | Oct 05, 2015 - Oct 17, 2015 | Live Event |
| SOS: SANS October Singapore 2015 | Singapore, SG | Oct 12, 2015 - Oct 24, 2015 | Live Event |
| SANS Tysons Corner 2015 | Tysons Corner, VAUS | Oct 12, 2015 - Oct 17, 2015 | Live Event |
| SANS Gulf Region 2015 | Dubai, AE | Oct 17, 2015 - Oct 29, 2015 | Live Event |
| SANS Tokyo Autumn 2015 | Tokyo, JP | Oct 19, 2015 - Oct 31, 2015 | Live Event |
| SANS Cyber Defense San Diego 2015 | San Diego, CAUS | Oct 19, 2015 - Oct 24, 2015 | Live Event |
| SANS Sydney 2015 | Sydney, AU | Nov 09, 2015 - Nov 21, 2015 | Live Event |
| SANS South Florida 2015 | Fort Lauderdale, FLUS | Nov 09, 2015 - Nov 14, 2015 | Live Event |
| SANS London 2015 | London, GB | Nov 14, 2015 - Nov 23, 2015 | Live Event |
| Pen Test Hackfest Summit & Training | Alexandria, VAUS | Nov 16, 2015 - Nov 23, 2015 | Live Event |
| SANS Hyderabad 2015 | Hyderabad, IN | Nov 24, 2015 - Dec 04, 2015 | Live Event |
| SANS Cape Town 2015 | Cape Town, ZA | Nov 30, 2015 - Dec 05, 2015 | Live Event |
| SANS San Francisco 2015 | San Francisco, CAUS | Nov 30, 2015 - Dec 05, 2015 | Live Event |
| Security Leadership Summit & Training | Dallas, TXUS | Dec 03, 2015 - Dec 10, 2015 | Live Event |
| SANS Milan 2015 | OnlineIT | Sep 07, 2015 - Sep 12, 2015 | Live Event |
| SANS OnDemand | Books & MP3s OnlyUS | Anytime | Self Paced |