Interested in learning more about security?

SANS Institute InfoSec Reading Room

An Approach to Application Security

Applications themselves are often crafted with little oversight of security professionals and without standards of development which creates an opportunity for disaster. This document discusses an approach to assessing application security that will work within most organizations. It first discusses some classes of threats that should be considered when designing security for applications. It then shows how to develop a simple Security Development Life Cycle to complement an organization's Systems Development Life Cycl...

An Approach to Application Security

By: Ian Rathie
SANS Security Essentials
GSEC Practical Assignment
Version 1.2f

1 Introduction

One of the basic flaws in how risk is assessed and security solutions implemented is that the various components are viewed within stovepipes rather than holistically together. For instance, there are usually separate approaches and teams assessing the network, operating system, web server, database, middleware and application. Given that the applications themselves are often crafted with little oversight of security professionals and without standards of development this has created an opportunity for disaster.

This paper details an approach to application security that when implemented not only brings these disparate views of risk together where they belong (within the application) but also prescribes how to involve the security professionals in the development process so that the resultant applications behaves predictably and with no surprises.

Development groups are often overlooked when Security teams work within an organization. Developers are usually hired for their development or coding expertise and may have zero or minimal security knowledge. As security often tends to focus on firewalls and servers, server and network administrators often get most of the attention. Without Security involvement, applications can be developed that create major security exposures, despite heavy investment in firewalls and other technologies. Such security flaws, if discovered late in the application development life cycle, can result in applications having to be redeveloped before being deployed, or can force reliance on expensive, inflexible security solutions that can be added after the fact, often using hardware or third party applications. Security organizations should have a liaison working closely with development teams to ensure that security expertise is available as applications are being developed. In addition, security education should be mandatory for application developers, to give them the tools they need to avoid developing insecure applications.

Early websites used static pages. Gaining access to the code running on the website had the potential to alter content displayed on the site, but it was not possible to compromise the security of the platform through the code making up the site. Solid network, platform and physical security were adequate defenses against a serious compromise. All of these areas of security are still important layers in defense of a website, but more is required today.

Today's e-commerce websites are comprised of many applications interacting with one another, with back-end databases, and with other entities. Applications are objects that have privileges that can be compromised. Firewalls protect websites by only allowing certain ports to be accessible to the outside world. Typically, only port 80, HTTP, and port 443, secure HTTP, are open to the outside world. Operating system security protects against compromise from within, an intruder getting onto the internal network somehow, legally or illegally. Both of these approaches make it difficult to compromise machines, but what if a hacker accesses resources by entering through the very ports that must be open for the website to operate? By attacking the applications running on a website, hackers can potentially do just that.

One way to ensure that developed applications are secure is to work with developers throughout the product development life cycle, ensuring that flexible, scaleable security is built into

applications from the start. The best approach for working with development teams is to develop a security life cycle that matches the Systems Development Life Cycle (SDLC) that is in use. Security can be planned right from the concept stage of a project, allowing for application growth and easier replacement of security components as technology develops. Decisions regarding security can be made before the application architecture is completed and before code is written.

This document discusses an approach to assessing application security that will work within most organizations. It first discusses some classes of threats that should be considered when designing security for applications. It then shows how to develop a simple Security Development Life Cycle to complement an organization's Systems Development Life Cycle. One approach for assessing risk in applications or systems is then discussed, with an example. Finally, some conclusions are reached about how to approach security in applications.

2 Classes of Threats

This section of the document discusses different types of threats that can be used to compromise an application, taking advantage of a security vulnerability. It is by no means a complete list of threats, but is included to give the reader an idea of the types of attacks to think about when designing security into web applications.

2.1 Privilege Elevation

Privilege elevation is a class of attacks where a hacker is able to increase his/her system privileges to a higher level than they should be. If successful, this type of attack can result in a hacker gaining privileges as high as root on a Unix system. An example of such an attack can be found at http://www.ciac.org/ciac/bulletins/m-026.shtml. In this example, authorized users of versions of OpenSSH earlier than 3.0.2 could gain the ability to run arbitrary code with superuser privileges. Once a hacker is able to run code with this level of privilege, the entire system is effectively compromised.

2.2 Unauthorized Data Access

One of the more popular types of attacks is gaining unauthorized access to data within an application. Data can be accessed on servers or on a network. The data can then be used for further attacks, such as using illicitly gained personal information to steal identities. Session hijacking is an example of this class of attack. HTTP is a stateless protocol, so in order to maintain the concept of logged-in session with a web application; session ids are used to tie user actions together at the web server level. Session hijacking involves guessing session ids of other users' web sessions. If session ids are not random enough, hackers can predict session ids of logged-in users and take over their sessions, thereby gaining access to any data accessible within the security context of the logged in users.

2.3 Denial of Service

Denial of service (DOS) attacks are currently getting a lot of attention, but the attacks that appear in the press are often network-based attacks. Applications can also be attacked in ways that render the application, and sometimes the entire machine, unusable. Poorly designed applications can sometimes be knocked offline simply by attempting logins to the userid that the application runs under enough times to lock out the userid. For an example of an application DOS attack, see http://www.securiteam.com/exploits/5XP0D1F5FM.html. In this case, a backdoor in the Kazaa and Morpheus web-based file sharing applications can be exploited to consume all available bandwidth, effectively knocking out the entire machine the application is running on. Also of interest is the fact that this attack bypasses personal firewalls setup to prevent this type of thing from happening.

2.4 Data Manipulation

Data Manipulation attacks involve changing data used by a website in order to gain some advantage or to embarrass the website's owners. Hackers will often gain access to HTML pages and change them to be satirical or offensive. One well-known example of a data manipulation attack is called hidden field manipulation. Data is often stored in hidden fields in a web page. This data can be viewed and changed using the 'View Source' option on the browser. If the values of the hidden fields are not verified by the application when a form page is submitted, results can be different than what is expected. For example, if prices on a shopping site are stored in hidden fields, what will happen if a hacker changes the price of an item from $100 to $1 before submitting the form and the server does not verify the price field? The hacker could receive the item for a fraction of the actual price, and the error might not be discovered until it is too late.

2.5 Identity Spoofing

Identity spoofing is a technique where a hacker uses the credentials of a legitimate user to gain access to an application or system. This can be a result of: a) users being careless with their ids and passwords, b) ids and passwords being transmitted over a network or stored on a server without encryption, or c) users setting easy to guess passwords. Once a hacker has possession of a user's credentials, he/she can login to the application with all of the privileges normally assigned to that user. This threat can be reduced or eliminated by requiring the use of strong passwords and forcing frequent password changes, by not storing or transmitting clear-text passwords, or by the use of hardware tokens. The approach to be taken depends on the value of the data protected by the id and password.

2.6 Cross-Site Scripting

Cross-site scripting is a relatively new approach that is being used to attack websites. It involves disguising a script as a URL variable within a URL from a legitimate site, and tricking a user into clicking on that URL, perhaps by sending it in an official looking e-mail or using some other approach. If the site does not check URL variables, the script will then execute in the user's browser. While this does not attack the website directly, it can be used to run arbitrary code in a

user's browser, collect userids and passwords, or anything else that a script can be used to do. One of the most well known cross-site scripting attacks was used against Microsoft's Hotmail service and other e-mail services. See a description at http://www.whitehatsec.com/labs/advisories/WH-Security_Advisory-08152001.html.

3 Security and the Systems Development Life Cycle

Most large organizations that develop software applications follow a Systems Development Life Cycle or SDLC. The SDLC describes the product development methodology that takes a system from concept to reality. An SDLC is multi-phased, with different phases representing different moments in a product's life. In order to ensure that applications are developed with adequate security, a Security Development Life Cycle, or SecDLC, should be in place that complements the SDLC being used by the development teams. The SecDLC does not have to match the SDLC exactly, as long as the required security tasks can be mapped to the phases of the SDLC.

3.1 A Simple Security Development Life Cycle

A general Security Development Life Cycle has two phases: Risk Analysis and Test. Each phase has its own set of tasks that contributes towards designing and building a more secure product.

3.1.1 Risk Analysis

In the Risk Analysis phase, the security and development teams work together to ensure that security is a major consideration in the design of the system. At the end of this phase, the full costs of implementing security measures for the designed application should be known. There are three major security activities in this phase.

1. Business Requirements
   As business requirements are developed, the security team needs to ensure that the organization's security standards are reflected in the business requirements documentation. At this stage, the security requirements will be at a high level. For example: ensuring that customer information to be transmitted across the Internet is encrypted. It is good practice for organizations to require that one of the areas that must sign off on business requirements is Information Security.

2. Risk Assessment
   Once business requirements are developed and signed off, the development team will be working on producing technical specifications for the new application. As the design begins to take shape, the security team should be performing risk analysis of the system design, ensuring that designs that would violate security standards are rejected, or modified. The risk assessment process documents risks and threats to the application, and determines countermeasures that must be taken in order to mitigate each threat. As the Risk Assessment is, arguably, the most important security document relating to an application, the next section of this document looks at risk assessments in more detail.

3. Technical Specifications
   Finally, countermeasures determined by the risk assessment need to be included in the technical specifications for the application. Again, it is good practice to have Information Security as one of the required sign-offs for this document.

3.1.2 Test

The Test phase is the next phase of the Security Development Life Cycle. In this phase, the security team works with the developers and the test team to ensure that countermeasures are correctly implemented and that code is developed following security best practices. There are three security activities in this phase.

1. Unit Testing
   As system modules are created, developers test them in isolation from the rest of the system using test driver programs and other tools. The security team should be working with the developers at this stage, defining tests for each module to test its security behavior. For example, all buffers for a module should be checked to determine if they are susceptible to overflows. If resources permit, it is good practice to have independent code reviews at this stage also. However, for many large organizations it is impractical to review all code, but the security team may want to require that certain code with major security implications (e.g. Code that performs authentication) undergo review.

2. Integration / Quality Assurance Testing
   This is the stage where the system modules are being assembled and tested as an integrated application. The security team should ensure that security testing is embedded in the overall test plan for the product at this stage. Specific tests should be in place to check authentication, authorizations and entitlements, and to test all countermeasures that were put in place as a result of the risk analysis.

3. Deployment
   At this stage, all integration, QA and user acceptance testing has been completed and the application is deployed in production. The security team should monitor the deployment to ensure that all countermeasures that are external to the application code (e.g. Firewalls, intrusion detection, etc.) are correctly installed so that they operate as anticipated. If the application is a web application, the last step before the site is entered into DNS servers should be to conduct a full penetration, or ethical hacking, test. This final level of security testing involves professional 'hackers' attempting to penetrate the system. The testing should be done from the perspective of an outsider with no approved system access and a malicious insider who has access to the system. It is often a good idea to have this testing done by a third party who had no involvement in the development or design of the system. Once the penetration testing is complete, there will be a very good picture of any remaining vulnerabilities that may need to be corrected prior to opening the system to the public. The reason for doing penetration testing at this stage, and not during integration testing, is to ensure that the system is tested in its production state, and not in a QA or development environment, which may not completely mirror production. It is okay to do the testing in a QA environment, and many companies do, but it should be done with the awareness that, if

the QA environment does not exactly mirror production there may be security vulnerabilities that are missed.

The Security Development Life Cycle that has been presented here is very general in nature. It can be adapted to fit any Systems Development Life Cycle. The approach is to ensure that most of the time Security personnel spend working on application development projects is consultative in nature, and takes place at points in the project where it is most effective. Concentrating on designing an application for security, and then testing to ensure that design goals were met, is significantly less expensive than retrofitting security to a product that is already built.

4 Creating the Risk Assessment

There are many different ways to perform risk assessments on applications and no one variation will work in every single environment. It is really a matter of finding an approach that works in most cases and only deviating from it when it is clearly not appropriate for the project being reviewed. Using a consistent approach whenever possible makes it easier for developers to know what to expect and allows Security personnel to follow a 'cookie cutter' approach to assessing risk. This section explains, at a high level, an approach to creating a risk assessment. There are many other approaches that could be used, but this method is fairly simple to use and works well.

The approach described involves using attack trees to quantify threats to an application, and then using Residual Risk Tables to show how threats are mitigated to ensure that the application is secure.

4.1 Attack Trees [1]

Attack trees are a formal method of quantifying threats against a computer system. They are explained in detail in the articles by Schneier and by Salter, Saydjari, Schneier and Wallner, cited in the References section of this document. They are flexible, easy to use, and modular.

An attack tree uses a tree structure with the attack goal as the root node. Ways of attacking the goal are represented as leaves on the tree. If some methods of attack turn out to be impossible to exploit, their leaves can be pruned from the tree. What remains is a tree containing only exploitable attacks.

The more complex a system is, the deeper the tree becomes. In many cases, attack methods involve breaching sub-systems that, themselves, are goals and, correspondingly, sub-trees. Every sub-tree in an attack tree represents either an AND or an OR junction, depending on whether the nodes below it must all be true to recognize the goal (AND) or any of the nodes below it must be true to recognize the goal (OR).

In Figure 1, a very simple attack tree is shown for breaking into a locked shed. It should be noted that this is an incomplete attack tree, as there are many other methods that could be used to break into a shed, but it can serve as an example for the purposes of this paper. The root of our tree is the goal of accessing the shed. There are four methods that could be used: stealing a key,

picking the lock, breaking a window, or using a copy of the key. Picking the lock and breaking a window are the simplest methods. If either of them can be done, the shed has been breached. Stealing a key is more difficult. To steal a key, an intruder would have to break into the house and obtain the key from its storage location, OR take the key off of the owner's key ring. Note that both of these methods could actually become rather large sub-trees themselves. If either of these methods is possible, the shed can be broken into. To obtain a copy of the key, a potential intruder would have to befriend the shed's owner AND borrow the key AND make a copy. If any of these tasks is impossible, the shed cannot be broken into.

To make attack trees more useful, we can use various methods of weighting the leaves. We can put costs of each task on each of the outer leaves and propagate the costs up the tree to determine costs of various attacks. The value of an AND node is the sum of the value of all of its sub-trees, while the value of OR node is the value of its lowest value sub-tree. We can also use any useful Boolean values on leaves, such as Possible and Impossible, or Special Equipment required or No Special Equipment required.

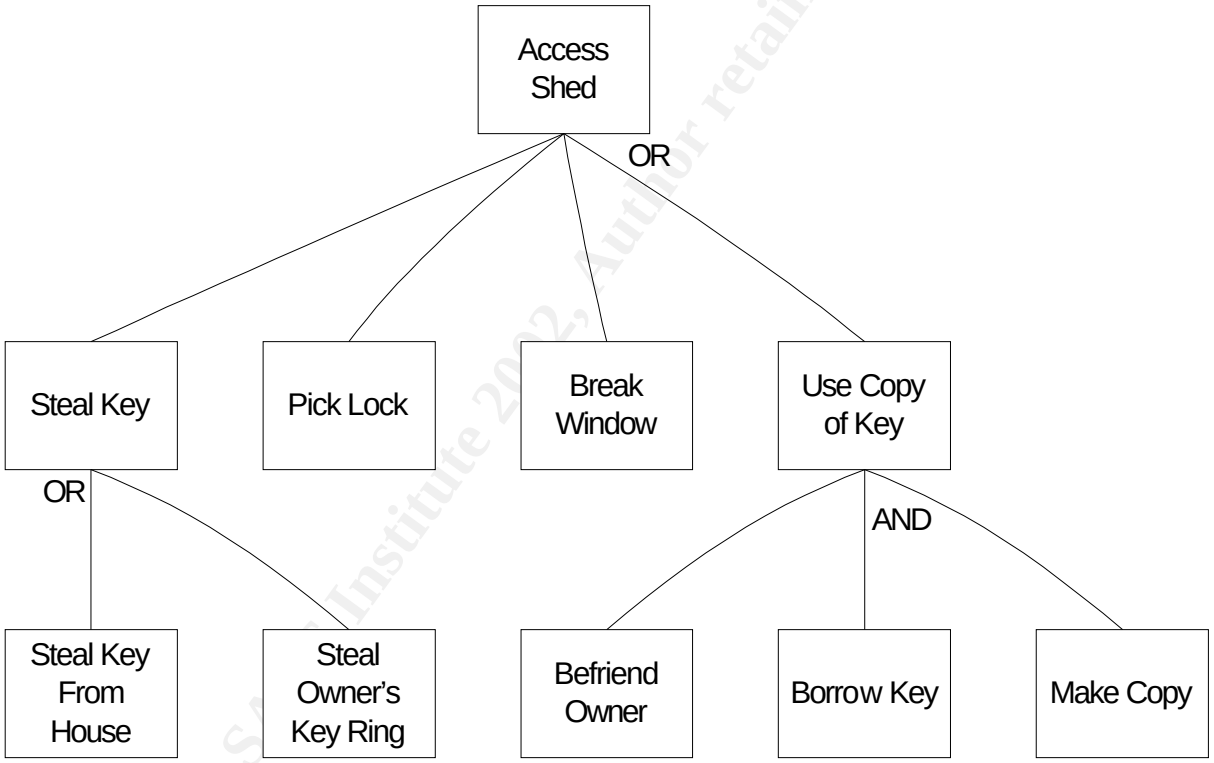

Figure 1: Attack tree for breaking into a shed

It is also possible to create a library of attack trees that can be used as sub-trees for larger attack trees. For example, the sub-tree for stealing a key would also work in an attack tree for stealing a car.

Attack trees are a very useful method of understanding the nature of threats against an application, the first step in assessing risk.

4.2 Residual Risk Tables

Once attack trees have been completed for an application, the next step is to determine how threats can be mitigated to bring the risk to an acceptable level. Enter residual risk tables.

Residual risk tables look at risks, threats, mitigating controls and residual risk levels once the mitigating control is in place. These columns are placed in a table and the risks, threats and controls are entered, leading to a residual risk level, which is also entered. A residual risk table for the shed example is shown in Figure 2.

The table shows how risks/threats can be mitigated to reduce the possibility of our shed being broken into. For the risk of the key being stolen, there are two threats to worry about: that the house is broken into and the key stolen from inside the house, or that the key is stolen off of the owner's key ring. The possibility of the house being broken into is mitigated by the alarm system on the house, leaving a low residual risk. The possibility of the key being taken from the key ring is mitigated by not carrying the shed key on the key ring, but since the key ring contains the house keys, it still could be stolen to obtain access to the house and get the shed key, leaving a medium residual risk. (Note: that the alarm system on the house might reduce this risk to low, but was not considered for this example.) For the risk of opening the lock without the key, there is no mitigating control for picking the lock. The residual risk is still high. All risks are tabulated in this fashion to get a picture of the residual risk of the shed being breached. The risk of using a key copy only requires one of the threats (it's an AND node) to be mitigated to bring the residual risk down.

Risk	Threat	Control	Residual Risk Level
Stolen Key	Break Into House	Alarm System	Low
	Steal Key Ring	Shed Key not on ring	Medium
Open the Lock Without Key	Pick Lock	None	High
Enter Through Window	Break Window	Install iron bars on window	Low
Use Key Copy	Befriend Owner	None	Low
	Borrow Key	Policy of not lending key	
	Make Copy	None	

Figure 2: Residual Risk Table for breaking into the shed.

All of the threats are mitigated using a control that is either a technical (alarm system, iron bars) or policy based (Not lending the key).

Residual risk tables create a picture of the risk to an application or system after attack trees are used to quantify threats. The example used, breaking into a locked shed, is simple for the purpose of illustrating this technique. In an actual application or system, many tables might have to be used in order to show risk mitigation in all sub-trees of a large attack tree. There are also

other columns that might be added to a residual risk table for a real-life application. Examples might be: cost of the control, OSI layer in which the threat occurs, or a reference number that refers to a labeled point in an application or architecture diagram at which the threat occurs. The tables are tools that can be customized to suit the particular needs of an organization. The tables are also modular in nature, like the attack trees, and can be re-used when other applications are developed that have similar requirements.

When used together, attack trees and residual risk tables create an effective picture of risks and risk mitigation for an application. The resulting document can be used to show controls during an audit, or to analyze required changes to security controls when an application's risk profile changes.

4 Conclusions

An effective application security program must start at the point where applications are designed. Attempting to implement security once an application has already been developed can be difficult and costly. In order to effectively embed security into the Systems Development Life Cycle an effective Security Development Life Cycle should be implemented across an organization.

This paper has shown one approach to creating a Security Development Life Cycle and an approach to assessing risks in systems and application. The SecDLC is simple to implement, but is an effective method of getting security in place at the beginning of the development cycle where it is usually simpler and less expensive to implement. The risk assessment method is simple, flexible and re-usable. It will work for almost any application or system and creates an effective snapshot of risks and risk mitigation that is an essential part of any set of application documentation.

5 Notes

[1] Schneier, p.1.

6 References

Conry-Murray, Andrew. "Best Practices: Keeping IT Alive Through the Dot-Com Bust. Ten Types of Hacks." August 3, 2001. URL: http://www.networkmagazine.com/article/NMG20010718S0001. (January 11, 2002).

Howard, Michael. Designing Secure Web-Based Applications for Microsoft Windows 2000. Redmond: Microsoft Press, 2000. 12 – 39.

Howard, Michael. "Secure Systems Begin With Knowing Your Threats, Part 1." October 2000. URL: http://security.devx.com/upload/free/Features/zones/security/articles/2000/09sept00/mh0900[2]-1.asp. (January 11, 2002).

Howard, Michael. "Secure Systems Begin With Knowing Your Threats, Part 2." November 2000. URL: http://security.devx.com/upload/free/Features/zones/security/articles/2000/10oct00/mh1000[1]-1.asp. (January 11, 2002).

Peteanu, Razvan. "Best Practices for Secure Development." V4.03. October 2001. URL: http://members.rogers.com/razvan.peteanu. (January 11, 2002).

Salter, Chris; Saydjari, O. Sami; Schneier, Bruce; Wallner, Jim. "Toward A Secure System Engineering Methodology." URL: http://www.counterpane.com/secure-methodology.html. (January 11, 2002).

Schneier, Bruce. "Attack Trees: Modeling Security Threats." December 1999. URL: http://www.counterpane.com/attacktrees-ddj-ft.html. (January 11, 2002).

Last Updated: June 28th, 2017

Upcoming SANS Training

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

SANS London July 2017	London, GB	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, JP	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CAUS	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS ICS & Energy-Houston 2017	Houston, TXUS	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, SG	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Munich Summer 2017	Munich, DE	Jul 10, 2017 - Jul 15, 2017	Live Event
SANSFIRE 2017	Washington, DCUS	Jul 22, 2017 - Jul 29, 2017	Live Event
Security Awareness Summit & Training 2017	Nashville, TNUS	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TXUS	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Hyderabad 2017	Hyderabad, IN	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Prague 2017	Prague, CZ	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MAUS	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS New York City 2017	New York City, NYUS	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UTUS	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VAUS	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Chicago 2017	Chicago, ILUS	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Adelaide 2017	Adelaide, AU	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SEC564:Red Team Ops	OnlineCAUS	Jun 29, 2017 - Jun 30, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced