Interested in learning more about security?

SANS Institute InfoSec Reading Room

Security Scenarios in Analysis and Design

This article addresses the issue of designing security into systems rather than trying to add it to systems after development. It is found by surveying teaching materials that security is only given brief acknowledgement as a concern in software development and that security is not well integrated into development life cycles used in schools. It is proposed that initial security requirements be addressed at the end of the requirements analysis phase and that update and refinement of security requirements continue throu...

Security Scenarios in Analysis and Design

Dwight A. Haworth
GSEC Practical Assignment 1.4b
September 16, 2002

Abstract

This article addresses the issue of designing security into systems rather than trying to add it to systems after development. It is found by surveying teaching materials that security is only given brief acknowledgement as a concern in software development and that security is not well integrated into development life cycles used in schools.

It is proposed that initial security requirements be addressed at the end of the requirements analysis phase and that update and refinement of security requirements continue through the design phase. This would be achieved by making the security administrator a major stakeholder in each and every system being developed. This would be implemented through a library of security scenarios that would be applied to each use case where appropriate. The management of the scenario library is discussed and the resource requirements are addressed.

Background

For the past thirty years, it has been understood that security concerns need to be addressed early in the system life cycle. Schell, Downey and Popek clearly speak to this issue in 1973 saying “Most contemporary shared computer systems are not secure because security was not a mandatory requirement of the initial hardware and software design” (I-1). Nearly thirty years later, Pipkin observes “It is nearly impossible to effectively add security to a system after it is designed” (71).

Ghosh devotes several pages to a discussion of failure to consider security requirements early in development (189-202). His comments carry a clear implication that such failures continue today. Such an implication leads to the question "What is being taught to students about security and the design of software today?"

Current Education

College book representatives of the major publishers were contacted to ascertain which books were their top selling textbooks in systems analysis and design. The goal was to form a list of the best selling textbooks and then review each textbook to determine how security considerations are addressed in each, if at all.

The returns from the textbook representatives produced a list of six textbooks:

- Systems Analysis and Design by Dennis and Wixom (2000);
- Modern Systems Analysis and Design by Hoffer, George, and Valacich (1999);
- Systems Analysis and Design by Kendall and Kendall (1999);
- Systems Analysis and Design in a Changing World by Satzinger, Jackson, and Burd (2000);
- Systems Analysis and Design by Shelly, Cashman, and Rosenblatt (2001); and
- Systems Analysis and Design Methods by Whitten, Bentley, and Dittman (2001).

Each of the textbooks in the list was searched for security, authorization, and authentication. Brief summaries of the coverage of each are given in the following paragraphs.

Hoffer, George and Valacich discuss security needs at a general level (578-80). They mention user views and give an example of authorization rules for databases; further, they have brief presentations on encryption and authentication. Significantly, they mention these items as important in designing user interfaces.

Kendall and Kendall discuss security needs at a general level, as well (847-49). They do not address security requirements and the relationship of those requirements to the design phase, preferring to give the impression that a firewall, encryption, passwords, and system logs are all that is needed to secure a system.

Dennis and Wixom address security at the architectural design level, but not at the software design level (263-67). They outline threat and risk analysis and remedies, among those are firewalls, call-back modem systems, strong passwords and encryption. They address neither the Internet nor security at the Web server.

Shelly, Cashman, and Rosenblatt discuss security needs at the system and architecture level (927-29). They mention intrusion detection and devote a page to several screens from a commercial third party product. They do not discuss security requirements and the relationship of those requirements to the software design phase, only requirements as related to system architecture.

Satzinger, Jackson, and Burd devote seven pages (out of more than 600) to security issues, including an introductory description of the problem of managing user access, and they identify these considerations as design phase issues (400-07). However, their discussion ends there, with no detail about how to integrate security issues into the overall system analysis and design process.

Whitten, Bentley, and Dittman speak briefly about authorization and authentication, using a display of a certificate to give the student an idea of a method for user authentication at the browser level (587-88). However, considering security requirements in the design of software is not mentioned.

As these textbooks represent current teaching, it is easy to generalize that the concern with functional requirements overshadows all else and that little is being done to emphasize the need for security requirements to be addressed. Indeed, when the space devoted to security is usually only two or three pages in a book of more than 500 pages, one may conclude that the subject is hardly being mentioned.

A survey of software engineering books produces a little more. Pressman discusses a security metric briefly but concedes later that security considerations are beyond the scope of his book (97, 774). Pfleeger lists several security requirements of a general nature (143). She also devotes two lines to security testing but does not specifically address security in the design of software (402). Sommerville gives a process for identifying security requirements in a chapter on critical systems (387-88). He separately discusses denial of service considerations (367). As a result his discussion is fragmented and becomes an add-on to his overall analysis and design process. It is no surprise to find students being instructed at the design phase of a term project, “Security Requirements. You are not required to do a lot on this aspect of the system, since it may be a major job in its own right” (Brabazon).

Moving out of the mainstream of systems analysis and design literature, Schneider and Winters refer to login and password failure scenarios (95). Further, they describe in name an attempting-a-prohibited-function scenario in the context of a login subsystem (95, 118). They also mention the need to find exception scenarios, but they do not specifically cite security deviations as being in the domain of exception scenarios (40). Wilkinson states that the analysis phase is devoted to uncovering requirements from the application domain (82). In discussing the design stage, she says "...certain real-world constraints must be considered and incorporated.... The impacts of these constraints, in such areas as environment, language, supporting software components, and performance requirements, are discussed next" (106). She cites security as one of the constraints and lists login authentication and the need for encrypted storage as two examples of security concerns (110). Wilkinson specifies that security considerations enter at the design stage, but leaves the discussion of security at that (110). Although security constraints are last in the list of constraints, they are exactly the kinds of constraints, alluded to in the quote above, that should condition the selection of platform, language, and supporting software components.

In the end, Wilkinson clearly leads one to the concept of addressing security in the documented scenarios that are input to the design phase. Exactly how this is

to be accomplished is left unstated. The purpose of this paper is to describe a method that will address security at the design stage. First, a conceptual model of security scenarios is presented. Second, the application of security scenarios to the development process is outlined. An example of security scenarios and their application is presented. Finally, administrative implications of the model and its application are addressed.

Conceptual Model

The primary reference for establishing security requirements must be the organizational security policy. Security requirements must take account of the data being manipulated by the system under design and the organizational policy relative to that data. This evaluation will determine security requirements for the system under design, and this evaluation cannot be undertaken until the data that will used by the system under design has been identified. It follows that security requirements cannot be addressed until late in the analysis phase when the data that will be manipulated by the system under design is known.

Not so obvious but of equal consideration is the other data on the platform on which the system will run. Even though the system under design may not deal with particularly sensitive information, the system must be implemented in such a fashion that other data on the server cannot be compromised through clever manipulation of the system under design. Therefore, the security requirements for the system under design must be as stringent as those required for the most sensitive data on the deployment platform. Anything less may place the sensitive data in jeopardy.

If the system under design deals with particularly sensitive data, the decision of platform is also important. Care must be taken to insure that the platform on which the system is placed is secure enough to protect the data of the system under design. The security of platform is determined by the security of those applications already resident on that platform. There will be little gain if a very secure system is placed on a platform on which reside other applications with little or no security. The decision of platform must be made first in the design phase to deal with the two-way nature of this decision.

To determine the level of security required, the data that the system will manipulate and the other data that will reside on the platform must be evaluated. First, this evaluation must identify the security policy (or policies) that applies to the data to be manipulated by the system. Requirements for encrypted storage, encrypted transmission, access restrictions, user authentication, user authorization, automatic backup, and activity logging must be integrated into the scenarios that will be input to the design phase. These are not exception scenarios; rather these are the mainline transaction scenarios that describe how the system will perform normally, hereinafter referred to as Catagory I scenarios.

Second, this evaluation must identify the level of protection that must be maintained with respect to the other data on the platform. Requirements for intrusion detection, access control, and preventing exploitation of known vulnerabilities in the platform, language, and supporting software components emerge at this point. These will be embodied in exception scenarios.

Two categories of exception security scenarios are possible. What happens if an attempt is made to violate aspects of security policy relative to the data manipulated by the system under design, and what happens if an attempt is made to exploit a vulnerability in the platform, language, database management system (DBMS), or graphical user interface (GUI)?

The first exception category deals with security policy relative to the system under design. These scenarios deal with the users of the system under design and their permissions to view and update data. These scenarios also deal with ensuring that attempts at violations are known and that the system under design responds appropriately. Examples of this type of scenario are "What happens if a user attempts to update a field for which he/she has only view permissions?" and "What happens if an unauthorized person attempts to use the system?" These scenarios will be referred to as Category II scenarios.

The second category of exception security scenarios is concerned with attempts to gain access outside the system under design. Examples of this type of scenario are "What happens if the user embeds a system command in the input string?" and "What happens if the user attempts a buffer overflow?" These scenarios, hereinafter referred to as Category III scenarios, are typical of attempts to compromise the entire platform through vulnerabilities in the platform, language, or supporting software. These attempts have little or nothing to do with the logic of the application domain; they are governed by the platform, language, and supporting software. The system under design would serve only as a conduit for the intrusion, but nevertheless it must be secured against such attempts.

Thus, a group of scenarios will be constructed to implement security policy in the areas of access, authentication, and authorization (Categories I and II) and to protect against attempts to exploit the identified vulnerabilities of the platform, language, and GUI to be employed (Category III).

Scenario Application

For the purpose of the present discussion, software development is regarded as proceeding from an analysis phase to a design phase and then to coding and testing. To insure that security is "designed in" rather than "added on", security issues must begin to be addressed at the end of the analysis phase. At this point, requirements and requirements-oriented scenarios make it possible to determine the data that will be handled by the system under design.

Category I security scenarios must be added to the documentation package as soon as sufficient information is available to judge which security policies are applicable to the system under consideration. Category I security scenarios are developed by the analysis team in collaboration with the security administrator. The security administrator must be regarded as a major stakeholder in the system under development and must be represented on the analysis and design team. The combination of requirements-oriented scenarios and Category I security scenarios expresses the normal behavior of the proposed system and becomes part of the input to the design phase.

Decisions about platform, language, and supporting software are made, if required, early in the design phase (Wilkinson 106). The combination of application requirements and security requirements and choices of platform, language, and supporting software determine the Category II security scenarios that will be applied. The choices about platform, languages, and supporting software as well as knowledge about the other data and systems on the platform become part of the input to development of Category III scenarios. Finally, the entire documentation package (application domain scenarios, Category I, II, and III security scenarios, and choices of platform, language, and supporting software) becomes input to the rest of the design phase.

It must be emphasized that at this phase of the development process not all of the potential vulnerabilities are known. Design decisions may introduce additional vulnerabilities; for example, a decision to employ user input to generate a file retrieval or a database search may expose the system under development to a new exploit. The vulnerability and the need for one or more scenarios to test for the vulnerability may not be obvious until a design decision is made. For that reason, the security administrator must remain a concerned participant in the design process throughout.

An Example

Consider the situation in which Internet access to a Human Resources Management System (HRMS) is being implemented. In this situation, there is an existing HRMS with its associated databases. The functional requirements for the Internet interface have been defined. It is now appropriate to add security scenarios to the documentation package that will be input to the design stage. These will be treated as Wilkinson recommends, "Scenarios should be very specific..." (56).

Category I

These scenarios describe normal operations of the system. The examples below are meant to representative of security scenarios. Although the first example is clearly focused on security issues, the other two examples are simply augmented

scenarios of normal application requirements. The development of such examples, that integrate security considerations into normal operational requirements, is the reason to include a security administrator on the design team.

"What happens when John Smith (a valid user) attempts to access the system?" This scenario will elicit the normal access procedures, including check of userid and password, and methods to display the opening menu.

"What happens when John Smith (an employee and valid user) attempts to view his personal data which is encrypted?" This scenario will elicit a check of the user's access permissions and normal retrieval and display procedures, including decryption processes.

"What happens when Mary French (a valid user) enters a new address (an encrypted field) into her record?" This scenario will elicit the normal interaction of the system components, including a check of the user's permissions to update the address, any edits on the address, and the encryption needed for the field or fields to be updated.

"What happens when James Wright (an authorized programmer) adds a new report program to the user menu for the HR system?" This scenario is typical of a group of scenarios that must be developed to embody the configuration management rules for the system.

These examples typify Category I scenarios. Their numbers must be expanded to account for all possible normal operations of the proposed system. Many of these scenarios can by created by adding security characteristics and qualifications to the scenarios that define the functional requirements of the system. When the configuration management rules are included, these scenarios may be more numerous than the functional scenarios.

Category II

Category II scenarios deal with attempts to violate policies that govern the normal operations of the system under development. These exceptions may be derived from the Category I scenarios, but they will be more numerous because they must deal with all possible violations of permissions to create, access, update, and delete data, and also to execute programs. Because of this dependence on the capabilities of the file management system or the database management system, these scenarios cannot be developed in full until decisions about the platform and, if one, database management system have been made.

"What happens if John Smith (an authorized user) attempts to view the birth date of Mary French for which he does not have view permission?" This and other similar scenarios must be constructed to account for all of the data that will have restricted access.

"What happens if Robert Jones (an authorized user) attempts to update his pay rate for which he does not have update permission?" This and other similar scenarios must be developed to account for all of the data that will have restricted update permissions.

Other access exceptions are represented by the following scenarios. "What happens if David Adams (an authorized user) attempts to add an employee record when he does not have permission to add records to the database?" "What happens if David Adams (an authorized user) attempts to delete an employee record when he does not have permission to delete records from the database?" "What happens if John Smith (an authorized user) attempts to execute a program that he does not have permission to execute?"

"What happens if Mike Early (an authorized user who does not have permission to delete program files) attempts to delete a program from the system?" This scenario represents the class of scenarios that will identify the response of the system to violations of configuration management rules.

Depending on the system under consideration, there may be many combinations of field, record, and file access permissions. To manage such complexity, a matrix of field, record, and file permissions and permission combinations may be constructed. This matrix can be used to identify the different classifications of system users, and scenarios can then be constructed for each classification of system user. Such a matrix will also allow analysts to assess the completeness of the defined access controls and may be useful in developing security controls for some commercial packages.

Category III

Given that an existing HRMS is being extended to provide Internet access, decisions about the DBMS and the database server are given. Decisions about the Web server and language for CGI processes have to be made. For the purposes of this paper, it will be assumed that the server has a Unix operating system, that the Web server is Apache, and that Standard C is being used for CGI processes. These scenarios will address specific vulnerabilities of the platform.

"What happens if the user attempts to embed ';who' in an input field?" This scenario is an example of many that will be needed to test the vulnerability of cgi scripts to metacharacters described by CERT Advisory CA-1997-25 Sanitizing User-Supplied Data in CGI Scripts. Although the particular command included above is not pernicious, its successful execution would indicate that the script is vulnerable to exploitation.

"What happens if the user enters a string of 65536 non-whitespace characters in

a character field?" This scenario must be applied to every character field that the user is allowed to enter. If the software handles a 64K string correctly, it will not be vulnerable to the buffer overflow described by Farrow.

"What happens if the user attempts to embed '<SCRIPT>' in an input field?" This scenario represents a number of scenarios that must be applied to every input that is used to dynamically form output, and the filter must eliminate all of the potentially exploitable characters identified by CERT CC Understanding Malicious Content Mitigation for Web Developers.

These Category I, II, and III scenarios must be applied to all applications in the system under development. They constitute the security requirements for the system, and they will become the basis for security testing during development.

Administrative Implications

Security requirements derive both from the data and the processing environment. After a level of security has been established for a given platform, based on the requirements of the most sensitive data in that environment, it is unlikely that data requiring a higher level of protection would be placed in that environment. To do so would require that all applications running in that environment be reviewed to determine whether they can support the higher security. Such an action would amount to more of the after-the-fact security engineering that Pipkin, Ghosh, and other authors have deplored. Such considerations suggest that the scenarios themselves may be relatively static and may be reapplied to new applications.

A computer environment may be compromised by any application that runs in that computer environment. This means that all applications that run in a given environment must enforce the same policies, and these policies are dictated by the highest requirements of the data that is stored and processed in that environment. It follows that the scenarios that apply to one application must be applied to all other applications that run in the same environment to assure a consistent policy and level of security is attained.

The foregoing observations lead to the concept of a scenario library so that each scenario is cataloged according to its platform, language, DBMS, GUI, or other vulnerable component. When a system is under development or update, one may select from the library the applicable scenarios to be applied in design and testing. Assuming that the chosen platform reflects the protection that must be afforded the data used by the system, the selection of scenarios would depend on the platform, language(s), DBMS, user interface, and other components being used in the application.

Further, because security scenarios are dictated by policy, they do not have to be developed specifically for each application; they may be drawn from a library of security scenarios that have been developed to support each security policy as

that policy is promulgated. Such a library will assure consistent implementation of policy and reduce the likelihood that a specific vulnerability is overlooked.

Beyond the scenarios themselves, the library could also include names of subroutines or functions or class methods that implement specific checks or filters implied by the Category III scenarios. Developing reusable modules that implement the requirements of scenarios will insure quality and consistency and will make the implementation of security requirements less burdensome on the development team. Moreover, the employment of tested reusable modules will help prevent programmers' oversights that sometimes introduce new vulnerabilities.

A scenario library that categorizes scenarios based on platform, language, DBMS, GUI, and other components will also aid in keeping security updated when new vulnerabilities are announced. One of the duties of the security administrator would be to keep the scenario library current. Knowledge of the scenarios and the reusable modules that implement the protections implied by Category III scenarios will support an efficient review of existing systems when new vulnerabilities are reported.

If the scenario library were implemented with a cross reference to the systems where each scenario was applied, quick remediation could also be achieved. The problem of finding all of the programs affected by a change in a given scenario would be reduced to a search of the library for the cross-references. Jennifer Myers acknowledges being aware of one set of metacharacters and discovering that the newline belongs in that same set (1). Consider the security administrator who discovers himself in a similar position. With a library of scenarios with cross-references to the systems where the relevant scenarios have been applied, remediation is greatly simplified by knowing which programs are involved. Remediation would be even easier if reusable modules were employed in development and cross-referenced to relevant scenarios.

The scenario library system will require resources, including a librarian who catalogs and adds new scenarios when they are developed, monitors security advisories to determine if a cataloged scenario requires update or a new scenario is needed, and maintains the cross-reference list of systems in which scenarios are employed. Knowing the scenario library well, the librarian might also serve on development teams in place of the security administrator.

Conclusion

Security considerations have not been included in the analysis and design process. To insure security is given adequate consideration in analysis and design, it is proposed that the security administrator or a knowledgeable representative be included in all analysis and design teams.

To support the added workload placed on the software development team, it is proposed that a system of security scenarios be used to express security requirements. Category I scenarios define security requirements in normal operations. Category II scenarios define responses to attempts to violate of the security requirements of normal operations. Category III scenarios define responses of the system under development to attempts to exploit vulnerabilities in the system itself, the platform, DBMS, or other software components. It is further recommended that a library of scenarios be assembled and cross-referenced to facilitate reuse and maintenance of the scenarios. It is believed that such measures will facilitate the inclusion of security considerations early in the software development life cycle.

Bibliography

Brabazon, Kevin. Phase III of the Term Project. 18 August 2002 <http://pages.stern.nyu.edu/~kbrabazo/PhaseIII.htm>.

CERT Advisory CA-1997-25 Sanitizing User-Supplied Data in CGI Scripts. 13 February 1998. Carnegie Mellon Software Engineering Institute CERT Coordination Center. 8 September 2002 <http://www.cert.org/advisories/CA-1997-25.html>.

CERT CC Understanding Malicious Content Mitigation for Web Developers. 2 February 2000. Carnegie Mellon Software Engineering Institute CERT Coordination Center. 14 September 2002 <http://www.cert.org/tech_tips/malicious_code_mitigation.html>.

Dennis, Alan and Barbara Wixom. Systems Analysis and Design. New York, NY: John Wiley & Sons, 2000.

Farrow, Rik. Network Defense. 9 September 2002 <<http://www.spirit.com/Network/net0999.txt>>.

Ghosh, Anup. E-Commerce Security. New York, NY: John Wiley & Sons, 1998.

Hoffer, Jeffrey, Joey George and Joseph Valacich. Modern Systems Analysis and Design, 2nd ed. Reading, MA: Addison-Wesley Longman, 1999.

Kendall, Kenneth and Julie Kendall. Systems Analysis and Design, 4th ed. Upper Saddle River, NJ: Prentice Hall, 1999.

Myers, Jennifer. CGI Security Holes. 6 February 1996. Newsgroup: comp. security.unix. 18 September 2002 <http://bau2.uibk.ac.at/matic/cgi2.htm>.

Pfleeger, Shari. Software Engineering Theory and Practice, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2001.

Pipkin, Donald. Information Security: Protecting the Global Enterprise. Upper Saddle River, NJ: Prentice Hall PTR, 2000.

Pressman, Roger. Software Engineering: a practitioner's approach, 5th ed. New York, NY: McGraw Hill, 2001.

Satzinger, John, Robert Jackson and Stephen Burd. Systems Analysis and Design in a Changing World. Boston, MA: Course Technology, 2000.

Schell, Roger R., Peter J. Downey and Gerald J. Popek. Preliminary Notes on

the Design of Secure Military Computer Systems, MCI-73-1, The MITRE Corporation, Bedford, MA 01730 (Jan. 1973). <http://seclab.cs.ucdavis.edu/projects/history/CD/index.html#sche73>.

Schneider, Geri and Jason P. Winters. Applying Use Cases: A Practical Guide. Reading, MA: Addison-Wesley, 1998.

Shelly, Gary, Thomas Cashman and Harry Rosenblatt. Systems Analysis and Design, 4th ed. Boston, MA: Course Technology, 2001.

Sommerville, Ian. Software Engineering, 6 ed. Reading, MA: Addison-Wesley, 2001.

Whitten, Jeffrey, Lonnie Bentley and Kevin Dittman. Systems Analysis and Design Methods, 5th ed. Boston, MA: McGraw-Hill Irwin, 2001.

Wilkinson, Nancy. Using CRC Cards. New York, NY: SIGS Books, 1995.

Last Updated: August 23rd, 2017

Upcoming SANS Training

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

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 Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 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
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS San Francisco Fall 2017	OnlineCAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced