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.

## Is Your Personal Financial Information Safe? Practical Lessons in Quicken Password Vulnerabilities

This paper examines password encryption and authentication techniques applied to the file-level protection of personal documents and databases. As a practical example, I have researched protection schemes used by Intuit Corporation's Quicken software. This personal financial software contains information that most people would consider to be extremely sensitive. However, the password protection and encryption schemes that Quicken uses fail to provide the level of security that might be expected. I've found that the pas...

Copyright SANS Institute

Author Retains Full Rights

AD

**Is Your Personal Financial Information Safe?**
**Practical Lessons in Quicken Password Vulnerabilities**

William Geimer
February 27, 2002

SANS Security Essentials GSEC Practical Assignment v. 1.3
GIAC Certification Administrivia v. 2.0

## Summary/Abstract

This paper examines password encryption and authentication techniques applied to the file-level protection of personal documents and databases. As a practical example, I have researched protection schemes used by Intuit Corporation's Quicken software. This personal financial software contains information that most people would consider to be extremely sensitive. However, the password protection and encryption schemes that Quicken uses fail to provide the level of security that might be expected. I've found that the password protection used by Quicken is easily reversed with the purchase of a $30 password cracking application.

Strong password encryption techniques provide a reasonable level of security and are difficult to crack. Implementing this technology to protect application data would not be technically difficult. However, application software makers choose to implement file-level protection that is relatively simple to circumvent. They may be wary of incurring higher support costs and running afoul of U.S. encryption laws. Public discussion of these vulnerabilities can help pressure the software manufacturers to strengthen their file-level protection. More consumer education might encourage software purchasers to consider security features when buying these products.

The responsibility for protection of this information lies with the consumer. The concept of "defense in depth," deploying layers of protection to limit your risk, must be employed to overcome the very weak protection offered by programs such as Quicken.[1] A well-layered defense must include physical security, operating system protection, a firewall, and updated anti-virus protection.

## Is My Financial Information Safe?

I've just completed my SANS Security Essentials course and I must admit that I'm feeling a bit paranoid. With visions of Code Red, Nimda, and SubSeven dancing in my head, I sat down to assess vulnerabilities in my home computing environment. My immediate concern went to potential disclosures of my financial information. In my case, this information was stored in Intuit Corporation's personal financial software, Quicken 2000 Deluxe.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Intuit is the number one maker of personal finance software and Quicken products comprise 73% of the market with over 15 million copies in use.[2] Quicken's features include checkbook balancing, stock and mutual fund portfolio tracking, budget reporting, and online bill payment.

I've configured Quicken to track all of my checking and credit card transactions and to keep a record of my 401(K) account balance. It also is configured to interface with my bank to electronically submit payments for bills, car loans, and mortgages.

I launched my Quicken 2000 software, entered my 10-digit password, and took a look around. The program displayed credit card numbers, checking account numbers, account balances, and almost every financial transaction I have made in the past three years - pretty sensitive stuff.

I had chosen to use the data file password protection option (which is not the default). Every time the application is launched, Quicken prompts for the data file password. Without the correct password, Quicken will not open the data file that contains all of my personal financial information. I assumed that the data would be encrypted and that the password protection would ensure that no personal information could be disclosed. Unfortunately, as it turns out, Quicken passwords are like door locks on cars, only the amateurs have trouble getting in.

## Password Primer

For application software such as Quicken, a password is typically stored on the local machine with the application data. A well-written application will encrypt passwords and protect against unauthorized disclosure, modification, or removal.

Encryption is the process whereby intelligible data is rendered unintelligible. An algorithm is applied against the data that you wish to protect. The original document is known as plaintext. The algorithm takes the input of the plaintext and a secret password (known as a key) and generates an encrypted file. The resulting data file (the ciphertext) is unreadable without some knowledge of the algorithm and password used to convert the plaintext into ciphertext.[3]

There are three basic types of encryption: symmetric, asymmetric, and hash.

Symmetric Encryption: Symmetric encryption uses a single key or password to both encrypt and decrypt the data. Many word processing programs use this method to encrypt documents. The password must be shared with anyone who is authorized to encrypt and decrypt the data.

Anyone who has ever received a protected Microsoft Word document as an attachment to an e-mail reading: "The password is *orange*," understands the problem with symmetric

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

encryption. Somehow the secret key must be discretely distributed. Asymmetric encryption solves this problem.

Asymmetric Encryption: Asymmetric, or public key encryption, uses two separate keys: one for encryption and one for decryption. The algorithms are written such that a shared key, known as the public key, is used to encrypt the plaintext data into ciphertext. A second key, known as the private key, is not shared and is used to decrypt the ciphertext.

The key pairs, the public and private keys, are unique to an individual or organization. When plaintext data is encrypted it can only be decrypted by the holder of the corresponding private key. Therefore, not only is the content of the message data encoded, it is *uniquely* encoded for the sole use of the owner of the private key.

The problem with this method of encoding is that the sender will need to have the recipient's public key prior to encrypting the message. Exchanging public keys can become somewhat burdensome.

Hash Encryption: Hash encryption is an encoding scheme that uses the input data to create a fixed-length numerical result. With this encryption method the secret key *is* the input data - often an entire file of data or a password phrase. The unique resulting number is called the hash. Because it is one-way, hash encryption is often used to verify that two instances of data are identical. If the two inputs are the same, they will generate the same hash.

Hash encryption is used to validate that the contents of a file have not changed. It sometimes used to compare the hash that was generated against the original file with the hash generated by the file after it has been transmitted over an insecure network. The hash algorithm generates a fixed-length number (32-bit, 64-bit, etc.). With a long hash, it is less likely that two input files or passwords would generate the same hash result. Therefore, a longer hash is considered to be more secure.

For example, a 2-bit hash algorithm could be given the input of a long string of characters such as a novel like *War and Peace*. The resulting hash might be the binary number 01. If you changed a word in the novel, the hash might change to be 10 or 11 and by comparing the hash values you would know that the file has changed. One could never reverse the process and generate *War and Peace* by knowing that the hash is 01. A 2-bit hash contains only four possible outcomes (00,01,10,11, or $2^2$ possibilities). The problem therefore with a 2-bit hash is that very many possible inputs would produce the same hash.[i] This is the reason that most hash algorithms are at least 32 bits long, resulting in $2^{32}$, or over 4 billion unique hash possibilities. [4]

Hash encryption is the primary password storage method in use today. A hash algorithm is applied to a password and generates the resulting fixed-length hash. Only by feeding the exact same password into the hash algorithm can you produce the same hash. Because the algorithm is irreversible, knowing the encrypted password, or hash, will not

[i] Hence the expression, "That 2-bit hash is good for nothing…"

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

enable you to quickly generate the original password. For this reason, the hash can be stored openly.

For example, if your password is:

```
password1
```

The resulting ASCII representation of the hash that is generated might look something like this:

```
$oUÔüâðµÑ$avuXV1Kx7jbXXApQpPi8j.
```

The hash that resulted from the algorithm is the starting point for hackers who are intent on getting your password. Password guessing programs, known as password crackers, use the password generation algorithm against all possible keyboard character combinations until the resulting character string is identical to the hash. This is known as a brute-force attack. [5]

A password cracker takes the known hash algorithm and generates a hash for each character combination and compares it to the known hash:

```
Hash(x)  Resulting Hash                    Known Hash
a = %oYÔüâðµÑ$avuXH1Kx7jCXXApQpPTNW ≠ $oUÔüâðµÑ$avuXV1Kx7jbXXApQpPi8j.
b = &^%Ô09*&^$avuXV1Kx7jbXXApQpPi)* ≠ $oUÔüâðµÑ$avuXV1Kx7jbXXApQpPi8j.

Then c, d, e, etc.
Then aa, ab, ac, etc.
Etc… until the resulting hash matches the known password hash.
```

However, a brute force attack is often not necessary. A brute force attack, attempting every character combination possible, can take years provided that a long hash and long password is used. Therefore, password crackers will typically first attempt to guess your password using common words and commonly used passwords prior to resorting to brute force. A good cracking program will allow for the use of industry or language specific dictionaries to customize their guesswork. If the target user speaks English and French and works in the automotive industry, the cracker could broaden his or her approach to include words from dictionaries that the target knows.

In the case of Intuit's Quicken program, as it turns out, the encryption algorithm *is* reversible and therefore highly vulnerable to being compromised.

## Under the Hood

My first quest was to find where Quicken stores the password. I started Quicken, changed only the password, and then exited the program. The only file that appeared to change was the qdata.qdf file. I decided to start there.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

I downloaded a shareware hex editor that allowed me to take a look at the QDF file. A hex editor will display the raw contents of any file. There was no sign of my password or any financial information in clear text. That was comforting. In fact, I later learned that the data is encrypted.

I then created two versions of my Quicken data file. One file, qdata_s.qdf, was protected with the password of "S", the other file, qdata_r.qdf with the password of "R". I thought that a quick ASCII comparison of the two files might show the location of the password within the file and possibly yield some information as to how it is stored.

Using the Microsoft Windows command-line file comparison utility (fc.exe) I compared the two data files. There were hundreds of lines that were different in an ASCII comparison. A binary comparison had similar results (8,128 lines of output). A truncated output of the ASCII comparison follows:

```
C:\> fc /n qdata_s.qdf qdata_r.qdf

Comparing files qdata_s.qdf and qdata_r.qdf
****** qdata_r.qdf
     1:  ¬ ½
     2:   ç+ ÓE÷ócî@GwŠ Íƒ«bª † é  $ìƒ éÌ hU^w›ƒš ‰ u±×!"© ÒÙŠÞ] ´‹üA ¶,¥Ï—C~ `-
Ü¦7z¿ÎŠª; qö    Õ© 9yhzÏ‹ÛE œæ¶   Kç˜ò$ÅG2ø¶vèMô0pþ{;

****** qdata_r.qdf
     1:  ¬ ½
     2:  DãIÀ ò¥Q~!:" :R E ÃŸM…÷ÙŸµÂˆ úÁ×™‡¹K \ógŸk ÒEË9Ñ1Ô3ôEè
Óùx z\^s_â¡ÈEîèN#|²¿ü[UXB 2w$ •Â¬Ï€ƒ®ß¶wÿ± y>ÍÍøÅ©‰*¥
```

(There were hundreds more lines similar to this.)

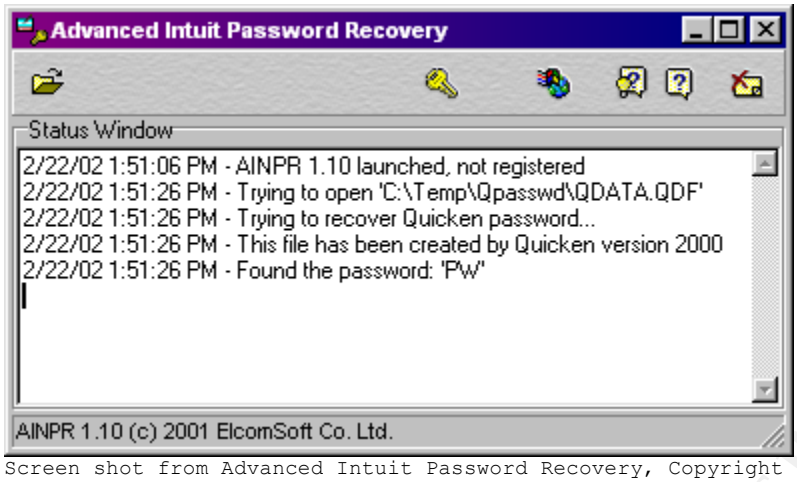
Obviously, the password was not stored in clear text, the password appeared to be stored in the QDF file, and it appeared to be encrypted.

A quick Internet search using the phrase "Quicken Password Vulnerability" returned several references to software designed to "recover" your forgotten Quicken password instantly. There were also many sites that offered a cracking service for a fee. Simply send them a protected file, and they send back the password to you. Passware (www.lostpastword.com) has an entire suite of cracking software for any application ($45 for Quicken Key, $395 for the suite).[6] Elcomsoft (www.elcomsoft.com) has a very similar set of applications ($30 for a personal copy, $60 for business use).[7]

Both applications have a demonstration version with a limitation on the length of the password that it will decode. Because of the demo version limitation, I changed my Quicken password to a 2-character string ("PW") and ran both programs. Running Elcomsoft's Advanced Intuit Password Recovery program, I opened the Quicken data file: qdata.qdf.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

The password was instantly revealed:



Screen shot from Advanced Intuit Password Recovery, Copyright 2001, Elcomsoft Co. Ltd.

The Passware program found the password just as quickly.

Both Passware and Elcomsoft offer software that can crack all versions of Quicken passwords except for the current Quicken, version 2002. Both sites have extensive software utility offerings to crack the file protection of Microsoft Office documents, archive files, personal databases, and e-mail files. Nothing on the file-level protection front seemed to be immune from these cracker applications. This vulnerability also serves to illustrate why it is a bad idea to use the same password on different systems – one compromised password could lead to additional unauthorized accesses.

## Going Around

The current version of Quicken seems to defend itself well against password cracking attempts but it appears to be vulnerable to password bypass attacks. A bypass attack involves using software designed to fool the application into opening the protected document without needing to know what the original password might be. By analyzing the way the application behaves when the correct password is entered into an application, bypass code can be designed to circumvent the normal application authentication process. A low-level analysis of inter-process communication during the authentication process is used to develop code that can fool the application software.

Crak Software (www.crak.com) sells a program to bypass Quicken 2002 password protection.[8] The Crak application is run first, and it runs the Quicken application from within its code.

At this point, I was sure that my Quicken password could be deciphered in under a second. I still didn't know too much about the encryption scheme and how it was broken.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

## How Did They Do That?

Because the two cracking applications worked so quickly and knowing that a brute-force attack would take some time, I was pretty sure that this software was reversing an encoded password. I decided to ask the experts.

During the course of my research on this topic I kept coming back to seven web sites that offered Quicken password-cracking utilities. There were a variety of software cracking sites that could be described as ranging from commercial-and-glossy to dark-and-sketchy. None of these sites offered a decent explanation of how they were able to accomplish the Quicken password recovery. Of course, if they told you how they did it they might not sell as much software.

I sent an e-mail to all seven software manufacturers and asked them how they did it. I was interested in learning how Quicken passwords are stored and if the password cracking utility uses a brute-force attack or simply unscrambles the password. As I expected, most of the software vendors politely declined to discuss their decryption methods. A few offered some helpful hints on where to find some information on how they do their work.

One of the vendors sent a very long rant that was very critical of the American education system, application programmers, and me ("Have you ever looked at a Quicken file, I know the answer is no."). You get the idea. It was very funny, but not informative.

The last two notes I received were the best.[ii] Both explained that Quicken stores the password along with the data in the qdata.qdf file. Quicken uses a simple algorithm to encode the password. In the early versions Quicken did not encrypt the data file but for versions 2000 and 2001 the data file is encrypted using the RC5 algorithm. However, the key for the RC5 algorithm is stored openly in the file, and therefore is easily decrypted. It appears that the data file is first decrypted using the algorithm and the key. Then the password is located in the file and decoded by reversing the simple Quicken password algorithm. To my surprise, one email that I received even included the Quicken password encryption algorithm.

Version 2002 uses a much stronger hash algorithm and therefore is not easy to crack (one vendor thought that it might be SHA1 encryption). As I mentioned, at least one of the cracking software vendors has developed a program to bypass the password verification routines in Quicken 2002 so password discovery becomes a moot exercise.

---

[ii] This information is based on private e-mail communications between the software vendors and me. I have chosen to not use their names and companies and to speak in general terms about the content of their notes. I did not request permission to publish their e-mails. My request for information did include the fact that I was doing research for a paper on Information Security. [Please note that the following information regarding Quicken password format is from non-authoritative sources and I was unable to verify this information independently using publicly available sources.]

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

One point that many of the vendors agreed on was that this is not a brute force attack. The speed of the password discovery tools seemed to suggest that as well. The password is clearly stored in a format that is relatively easy to reverse and it does not appear that any version of Quicken can be considered safe simply by using the password protection of the application.

## Communicating with the Bank

Quicken can be configured to communicate over the Internet with bank and check writing services. This feature is often used to pay bills, reconcile checkbooks, and update account balances. I've shown that the Quicken data file password is vulnerable to password cracking attacks. However, there is a second password needed before online transactions are authorized. The second password, the bank personal identification number (PIN), must be entered before each online session. Therefore, the data file password alone cannot be used to transfer funds and authorize check payments.

I looked at the traffic generated during an online transaction session. I used the free network protocol analyzer Ethereal (www.ethereal.com) to examine the data packets sent between my PC and the bank's servers.[9] Most of the data was encrypted using secure sockets layer (SSL) and there were no obvious plaintext transmissions of account numbers, transactions, or account balances. The SSL encryption ensures that the contents of the PC-to-bank transaction conversation could not easily be disclosed should it be intercepted along the way.

Because the PIN must be entered each time an online transaction is conducted, this suggests that the PIN is not saved on the PC and is not vulnerable to the same type of password attacks as the Quicken data file password. Obviously, it would not be a good idea to use the bank PIN as the Quicken data password.

## Why Application Software is Vulnerable

Software vendors could easily provide more protection for their customer's data. Their reasons for not doing so might include the following:

Encryption Laws: Until 1996, U.S. Government export laws strictly restricted the export of encryption and decryption technologies.[10] Prior to this change, the export of encryption technologies with long keys was prohibited. More recently, restrictions on encryption key length have been relaxed somewhat.

In addition to technology restrictions, the laws vary depending on the destination country. Today, there are stricter export laws to Cuba, Iran, Iraq, Libya, North Korea, Sudan and Syria (known as the Terrorist 7 countries). Naturally, these laws are subject to change at any time.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

An application vendor such as Intuit might prefer to configure their software's protection so that it could be legally sold in the largest number of countries as possible. Of course, they could choose to create separate versions of their product for each category of export (e.g., US, UK, International, and Terrorist 7 versions). That would add to the development, testing, and distribution costs.

Support Costs: The use of passwords inevitably produces forgotten passwords and support calls to the software vendor. The calls and e-mails that are generated by lost password requests are an expense to the manufacturer. Lost passwords also can be frustrating and costly for the software manufacturer's customers. Intuit has a password removal service that they offer for a fee.[11] They ask that you mail the QDF file to them and they will unprotect the file for you.

It is conceivable that if their password encryption technology were strengthened, even Intuit might be unable to recover a lost password in a timely manner. Application vendors might prefer to keep the software security somewhat lax so that their customers never experience a catastrophic loss of data due to a lost password.

Development Costs vs. Benefit: Most software is evaluated based on features and ease of use. Software vendors are focused on adding new features to their products. Strong feature functionality wins awards and drives sales. Magazines and web sites such as PC Week and CNET often compare the features of application software in their reviews. However, you rarely see a category for security and privacy on the feature comparison charts.[12]

If strong security technology will cost more to develop and won't help the bottom line, you can expect that it will not be a priority for the application vendors.

## Conclusion

Information Security professionals teach the importance of choosing unique passwords that are lengthy, difficult to guess, and contain a random mixture of alphanumeric characters. A strong password is more difficult to guess and it will require longer periods of time for a random password generator to crack.[13] These best practices are only worthwhile if the password authentication program stores the password securely. Password protection relies heavily on the effectiveness of the authentication software.

Application programs often provide very weak protection for your sensitive data and should be considered inherently insecure. This is definitely the case with Intuit's Quicken software. The password is stored with the data file and can easily be discovered using any one of several inexpensive password recovery programs.

Software vendors are motivated by software sales and focus their efforts on building features and making their software easy to use. A typical consumer most likely will make their purchasing decisions with little regard to the underlying security. A typical

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

vendor will not likely tighten their application security unless publicity about exploits and vulnerabilities is viewed as a threat to their future sales. Public discussion of these issues can help both to educate the consumer and to motivate the vendor to fix these problems.

The software user must take action to protect this data. Other layers of defense must be employed to protect the confidentiality of your personal financial data. These layers should include a secure operating system authentication, limiting physical access to your systems, a personal firewall application, and the use of current anti-virus software.

## References


[1] Cole, Eric, et al. SANS Security Essentials II: Network Security, SANS Institute Security Essentials Training Material, version 1.4, updated 1 Nov 2001, p.1-3.
[2] Atanasov, Maria. "Checkbook Challengers." Ziff Davis Smart Business. 1 Sep 2001. URL:http://www.smartbusinessmag.com/article/0,3658,s%253D104%2526a%253D10454,00.asp (25 Feb 2002).
[3] Cole, Eric. Hackers Beware. Indianapolis: New Riders, 2001. p. 296.
[4] Ibid. pp. 296-297.
[5] Cliff, A. "Password Crackers - Ensuring the Security of Your Password." 19 Feb 2001. URL: http://online.securityfocus.com/infocus/1192 (27 Feb 2002).
[6] Passware. URL: www.lostpassword.com (27 Feb 2002).
[7] Elcomsoft Co. Ltd. URL: www.elcomsoft.com (27 Feb 2002).
[8] Crak Software. URL: www.crak.com (27 Feb 2002).
[9] Ethereal. URL: www.ethereal.com (4 Mar 2002).
[10] U.S. Department of Commerce, Office of Strategic Trade and Foreign Policy Controls, Commercial Encryption Export Controls. URL: http://www.bxa.doc.gov/Encryption/Default.htm (27 Feb 2002).
[11] Intuit FAQ. "Getting into a password protected Quicken file without the password" http://www.intuit.com/support/quicken/2002/win/1250.html (27 Feb 2002)
[12] CNET Software Head To Head, We Put the Top Two Money Managers to the Test, 8/27/01, http://www.cnet.com/software/0-3227903-8-6954890-1.html
[13] Yan, Jianxin, et al. "The Memorability and Security of Passwords – Some Empirical Results." Cambridge University Computer Laboratory. URL: http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/tr500.pdf (27 Feb 2002).


## Other Resources Used


Electronic Privacy Information Center, "Cryptography and Liberty 2000: An International Survey of Encryption Policy." Washington, DC. http://www2.epic.org/reports/crypto2000/countries.html#Heading133

Seltzer, Larry J. "Password Crackers." PC Week Magazine. 13 Dec 2001. URL: http://www.pcmag.com/article/0,2997,apn=2&s=1481&a=19957&ap=1,00.asp (27 Feb 2002).


© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Srinivas, Raghavan N. “Network Security and Java Technology--A Primer.” URL: http://servlet.java.sun.com/javaone/javaone2000/pdfs/TS-1405.pdf (27 Feb 2002).

Zetter, Kim and Brandt, Andrew. “How Hackers Hack.” ITWorld.com. 2 Apr 2001. URL: http://www.itworld.com/Sec/2199/PCW01040245726/ (27 Feb 2002).

© SANS Institute 2002, Author retains full rights.

Last Updated: May 19th, 2013

# Upcoming SANS Training

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

| | | | |
|---|---|---|---|
| **SANS Malaysia @ MCMC 2013** | **Cyberjaya, MY** | **Jun 03, 2013 - Jun 08, 2013** | **Live Event** |
| **SANS Pen Test Berlin 2013** | **Berlin, DE** | **Jun 03, 2013 - Jun 08, 2013** | **Live Event** |
| **Industial Control Systems Security Training - Houston** | **Houston, TXUS** | **Jun 10, 2013 - Jun 15, 2013** | **Live Event** |
| **Security Impact of IPv6 Summit 2013** | **Washington, DCUS** | **Jun 14, 2013 - Jun 16, 2013** | **Live Event** |
| **SANSFIRE 2013** | **Washington, DCUS** | **Jun 14, 2013 - Jun 22, 2013** | **Live Event** |
| **SANS Canberra 2013** | **Canberra, AU** | **Jul 01, 2013 - Jul 13, 2013** | **Live Event** |
| **SANS London Summer 2013** | **London, GB** | **Jul 09, 2013 - Jul 16, 2013** | **Live Event** |
| **Digital Forensics & Incident Response Summit 2013** | **Austin, TXUS** | **Jul 09, 2013 - Jul 16, 2013** | **Live Event** |
| **SANS Rocky Mountain 2013** | **Denver, COUS** | **Jul 14, 2013 - Jul 20, 2013** | **Live Event** |
| **SANS Mumbai 2013** | **Mumbai, IN** | **Jul 22, 2013 - Jul 27, 2013** | **Live Event** |
| **SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification** | **Washington, DCUS** | **Jul 22, 2013 - Jul 26, 2013** | **Live Event** |
| **SANS San Francisco 2013** | **San Francisco, CAUS** | **Jul 29, 2013 - Aug 03, 2013** | **Live Event** |
| **SANS SEC 560: Network Penetration Testing @ Bangalore 2013** | **Bangalore, IN** | **Aug 05, 2013 - Aug 10, 2013** | **Live Event** |
| **SANS Boston 2013** | **Boston, MAUS** | **Aug 05, 2013 - Aug 10, 2013** | **Live Event** |
| **Critical Security Controls Summit** | **Washington, DCUS** | **Aug 12, 2013 - Aug 18, 2013** | **Live Event** |
| **Industrial Control Systems Security Training - DC** | **Washington, DCUS** | **Aug 12, 2013 - Aug 16, 2013** | **Live Event** |
| **Mobile Device Security Summit 2013** | **OnlineCAUS** | **May 30, 2013 - Jun 06, 2013** | **Live Event** |
| **SANS OnDemand** | **Books & MP3s OnlyUS** | **Anytime** | **Self Paced** |