Security Issues with DNS

Florent Carli

Florent Carli
SANS GSEC Practical Assignment
Version 1.4b
Option 1 - Research on Topics in Information Security

Security Issues with DNS

Abstract:

DNS continues to be a nice target for hackers. The ubiquity of BIND as DNS server software around the world, and the possibilities a hacker can expect should he succeed in taking over a server or simply use DNS implementation to reorientate traffic, are some of the things which make DNS a source of security issues.

This document first reviews some basics about how DNS works, then goes into explaining the different ways a hacker can attack the DNS protocol implementation to use it to his own advantage. We will focus on the relationship between all the terms we hear, which are usually misemployed. We will then review the different possible server attacks and finish by explaining some of the ways that should be used to protect against these issues.

Introduction:

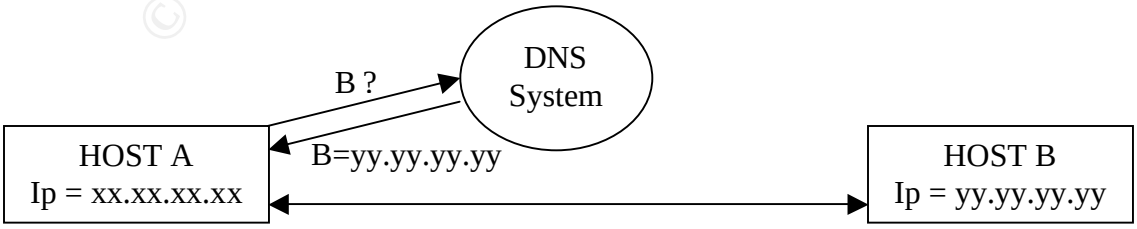
Humans can't think like computers. They just can't remember dozens of IP addresses. They need easy-to-remember names to locate their mail server or their favorite web pages.
To make our lives on the Internet easy, DNS was therefore invented. And with it came a new place for hackers of all sorts to have fun.
Moreover, the purpose of DNS makes it a very sensitive area; for this is the place the client connection is orientated. The possibilities a black-hat can have by succeeding in hacking DNS are tremendous (a user can be directed to a host controlled by a hacker, whatever service he might be using: http, ftp, telnet …). Anything is possible!
There are two ways DNS can be hacked: by using protocol attacks (attacks based on how DNS is actually working) or by using server attacks (attacks based on the bugs or flaws of the programs or machines running DNS services). We will see both kinds, as well as how to protect against them and why we should worry. But first, let's try to understand the basics of DNS.

A. How does DNS work?

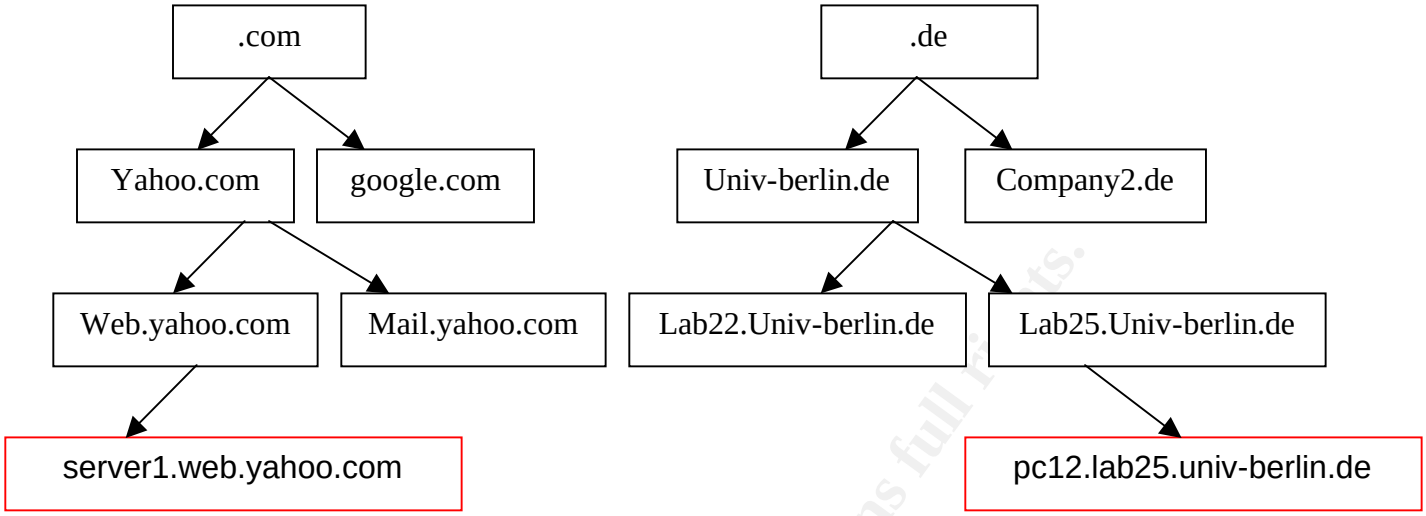
DNS stands for Domain Name Service. All in all, what it does is translate a host's name into its IP address.

Internet is an IP network. Every host is affected an IP address that must be known to any other host willing to communicate. But it would be impossible for a human being to remember all the IP addresses it will use on the Internet. It would be possible to create the mappings between IP addresses and names locally to each computer. But the update of those tables would be very complex and slow given the number of computers on the net and how fast a modification in their address or name can occur.

DNS provides a way to know the IP address of any host on the Internet. It is no different than any other directory service.



DNS is also a hierarchical service. Indeed a name has a structure: it can be server1.web.yahoo.com or pc12.lab25.univ-berlin.de



A host is a leaf on this tree, and any other node is called a “domain”. In our example, server1 is one of the hosts in the “web.yahoo.com” domain.
We need servers to be responsible to find any name-ip mapping. These servers are called “authoritative” for a domain when they can map the IP addresses with the host names of all the hosts in the domain.

When a host asks for a mapping, the search will start at the top of the tree, going down. In our example we might have a DNS server for the Univ-berlin.de domain, which may know the IP address of pc12.lab25.univ-berlin.de or the authoritative DNS server for this host might be lower, at the Lab22.Univ-berlin.de level. In any case the DNS server at the “.de” level orientates the search toward univ-berlin.de DNS, etc.

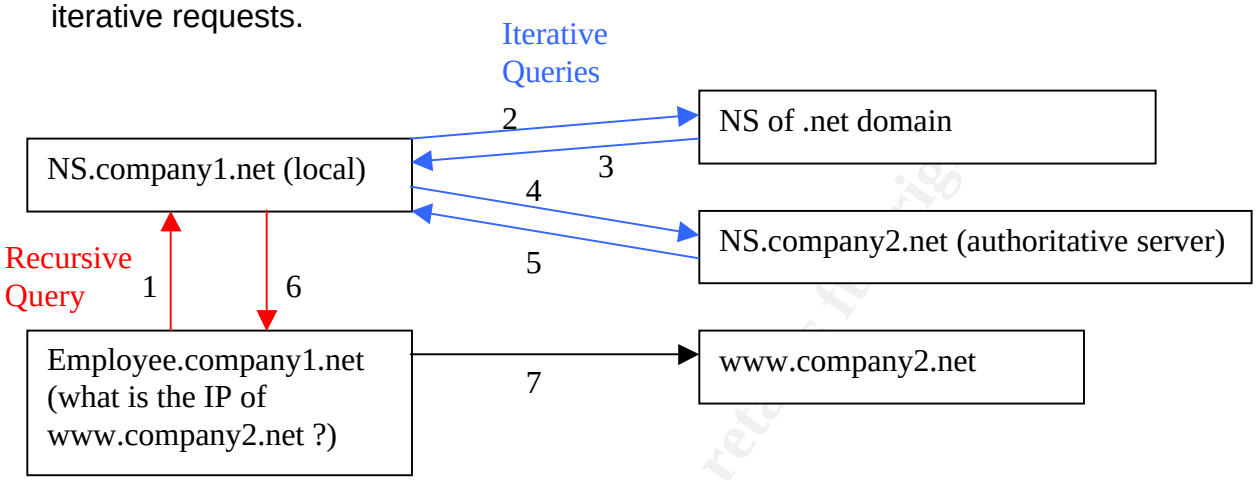
Reference 1 provides more detailed information of the basics of DNS.

Iterative and Recursive Queries

Before going on with the attacks, we need to understand the difference between a recursive and an iterative DNS query.
When a host queries a DNS server, it can choose to use a recursive query: in that case the client wants the answer, or an error message that what it’s looking for could not be found anywhere. We see that the queried host must do whatever it takes to find the answer: query other servers until it gets the information, or until the name query fails.
When a host queries a DNS server in the iterative way, it basically asks the server for an answer IF the server knows it (if it has the answer in its cache). If it

does not, then the client will receive a "referral" which is the name of a server that may have the answer (a authoritative server at a lower level in the hierarchical structure we talked about).

Recursive queries are usually made by client hosts so that they don't have to take care of the whole search process, whereas local DNS servers usually make iterative requests.



Reference 2 provides an applet that demonstrates the particularities of these two different kinds of requests.

B. DNS: Protocol Attacks

As we saw earlier, DNS protocol attacks are based on flaws in the DNS protocol implementation – the actual way DNS works on the Internet.

When talking about DNS protocol attacks, we often hear 3 terms:

- DNS spoofing
- DNS ID hacking
- DNS cache poisoning

We will now explain what each of them means and how they relate to each other depending on the situation.

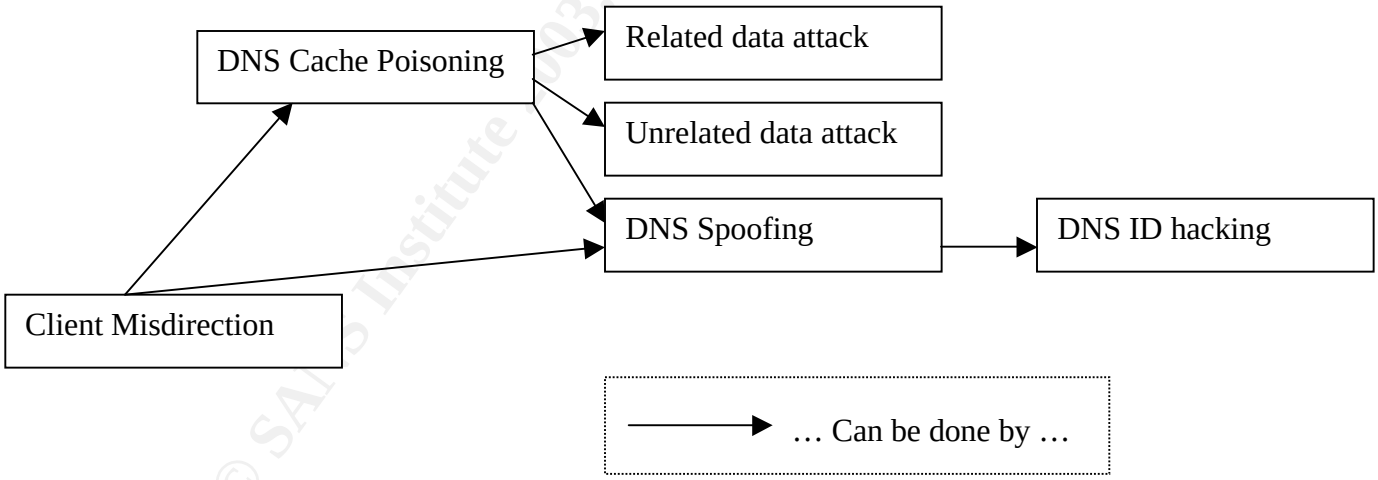
DNS cache poisoning relates to an attack consisting of making a DNS server cache false information: usually, a wrong record that will map a name to a "wrong" IP address. We will see that there are different ways for a hacker to do that, and that they are often related to DNS spoofing. With DNS cache poisoning, the hacker will try to make a DNS answer something he wants for a specific request. For instance, try to make the ns.defense.gov DNS to answer with the IP of the hacker's computer to any query about the IP of telnetaccess.defense.gov.

Cache poisoning can be done by related data or unrelated data attacks (we will see what these are later), as well as by using DNS spoofing.

DNS spoofing is a term referring to the action of answering a DNS request that was intended for another server (a "real" DNS server). This can be in a server-server exchange (a DNS server asks another for a mapping) or in a client-server dialog (when a client asks a DNS server for a mapping). There is no functional difference. The hacker "spoofs" the DNS server's answer by answering with the DNS server's IP address in the packets' source-address field.

But this is not enough to spoof a DNS reply. DNS uses ID number to identify queries and answer, so the hacker needs to find the ID the client is waiting for. For that, he will use DNS ID hacking. With DNS spoofing, the hacker will try to impersonate the DNS reply so that the requesting client is misdirected, but without touching the DNS cache of the impersonated DNS.
Note: if the requesting client is another DNS server, then its cache is going to be poisoned, which would therefore be a way to do cache-poisoning.

Given the fact that "Client Misdirection" (making a client host go where he was not willing to in the first place) is in general the basic purpose of DNS hacking, then we finally come with this :



We have already defined the concepts (Cache poisoning, DNS spoofing and Client Misdirection). Now we will explain how the real attacks (Related and Unrelated data attacks, and DNS ID hacking) work. Reference 4 and 5 will provide more information on ID hacking and CachePoisoning attacks.

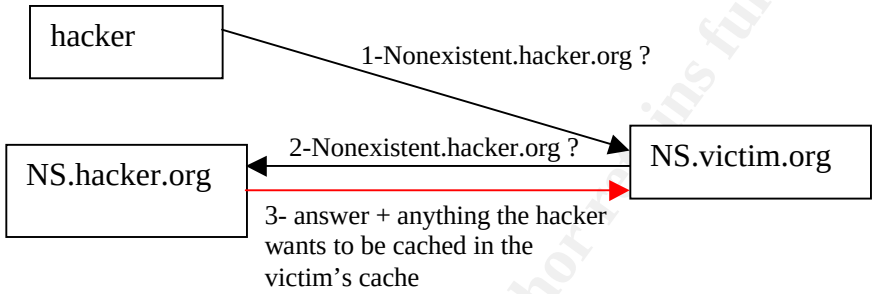
Unrelated Data Attack:

This was the first attack, the simplest and the most widely used:

1 – The hacker asks the victim DNS for a nonexistent name mapping in a domain for which he controls the DNS. The hacker uses a "**recursive**" query so that the remote DNS server will make further inquiries by itself.

2 – The remote DNS, which is not aware of such mapping, will go and ask the DNS server responsible for the required domain. Remember this server is under the control of the hacker.

3 – The hacker will answer, and add in the answer anything he wants to be cached in the victim DNS' cache. That way, he will have poisoned the cache of the remote DNS server.



This problem has been fixed in BIND, by forbidding anything that is not related to the original request to be cached.

Related Data Attack

The attack is exactly the same as an unrelated data attack, but this time the hacker has to make the "extra" information related to the original query. He solves this problem by adding MX, CNAME or NS records, which point to unrelated data. Those three records we can find in a DNS database are not a real "mapping" between an IP address and a hostname. They point to some other useful information (MX: mail server for a domain, CNAME: Canonical name for an alias, NS: DNS servers for a domain).
Therefore, the information in these records is "related" to the original request, but they can point to totally different information the hacker wants to be cached.

This problem has also been fixed in BIND, by rejecting all the "out of zone" information, that is to say all the information that is not fully and directly related to the zone the DNS is responsible for.

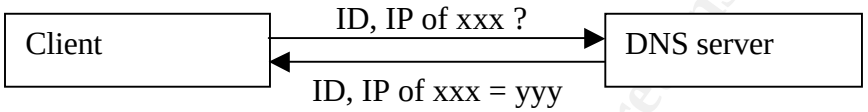
Both these attacks are quite old and won't work anymore on BIND. Indeed, a lot of patches have been released since they were discovered.

However, we still hear about DNS spoofing, via the DNS ID hacking technique.

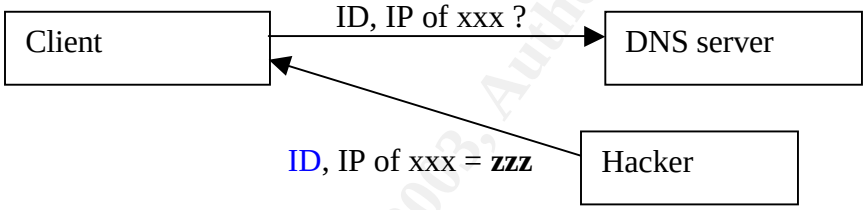
DNS ID hacking

As I explained before, the DNS ID hacking is a necessary technique for a hacker to succeed in impersonating a DNS server (this is the basic of DNS spoofing). Indeed, to be able to forge a fake answer, the hacker must first use the DNS server's IP address as a source for his own IP packet, and then use the correct ID number without which the client won't take the reply into account.

The client will send a query to the DNS server using a specific ID number. The server will reply using the same ID number. This is the number the hacker has to find.



Normal dialog



On a LAN, getting the ID is pretty easy: all the hacker has to do is sniff the network for the initial query and answer quicker than the DNS (which is very easy on a LAN). The late reply from the real DNS server will be discarded.

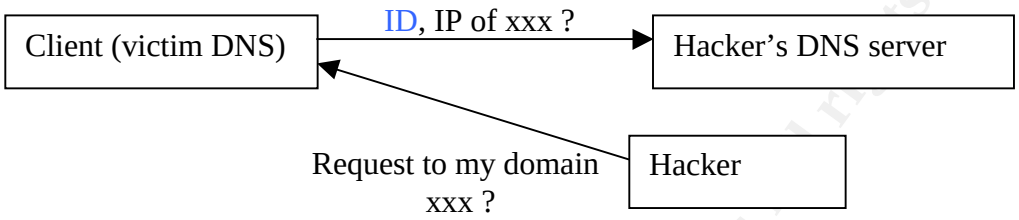
When the hacker is not on the same LAN as the victim client, he has four options to try to guess the ID:

1 – Test all the possible values of the ID flag (or as many random values as you can before the NS replies): this is quite an obsolete method and quite useless since its only advantage is that it will let you know what the ID is.

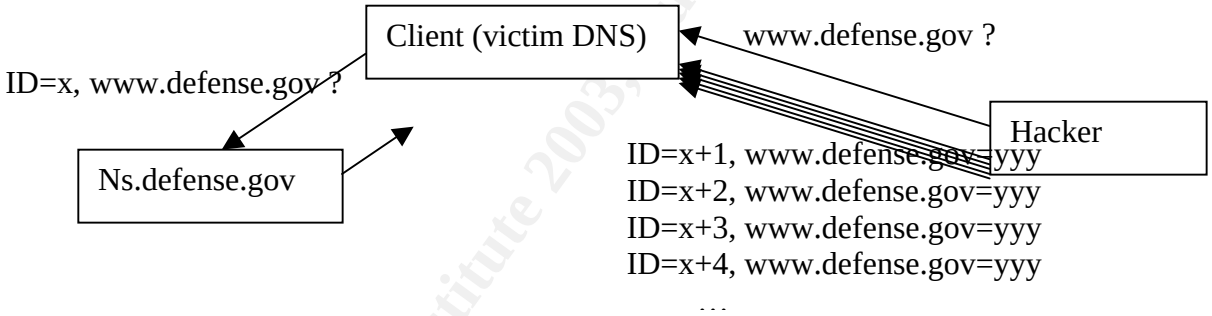
2 – Flood the DNS server to buy some more time for trying different ID numbers. The hacker can even hope it will crash the server.

3 – Send a few hundred replies at the same time to increase his chances to find the good ID. The hacker can do that several times one after the other with different ranges until the server replies.

4 – Use a vulnerability in the server, knowing that some of them just increase the ID number from one request to another. This works in a server-server dialog (The "client" in our last figure is a DNS server, and the hacker is trying to poison its cache). In that case, the hacker can first make a request to the "client" using a host name in a zone controlled by the hacker, and sniff the ID used by the victim DNS server.



That way the hacker knows the range of IDs currently used by the victim server. All he has to do now is to make a request to the host name he wants to poison the cache of our victim with, and fake the answer using an increased value of the stolen ID (using several replies will increase the chances in case the victim is busy)



If the hacker does not control a DNS, he can still try a range of random ID until it is too late (the real DNS server replies). If the hacker receives an answer from the victim saying the address is "yyy" then, it worked. Otherwise, that means it's too late, and that another range can be tried with another query.

Note: There is a vulnerability in Windows95 which makes very easy for a hacker to guess the ID: Windows uses 1 by default and then increments if other queries are directly following.

This type of flaw in DNS server software is where protocol attacks meet server attacks.

C. DNS: Server Attacks

It is quite difficult to detail any of those attacks. They are evolving way too fast. As soon as a bug is discovered, a patch is released and the contrary is also true.

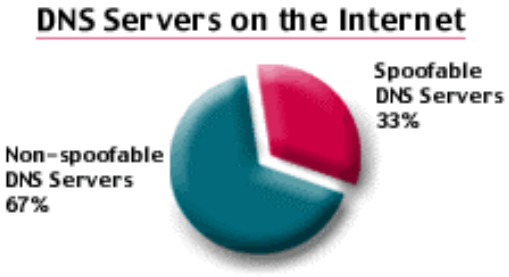
Two kinds of attacks can aim at the server instead of the DNS protocol itself (ref. 6).

- Attacks taking advantage of bugs in DNS Software implementation (buffer overflows in BIND for instance) or in any other running service on the DNS server machine (ref. 7).
- Attack by Denial of Service (using flooding for instance, of the DNS service using in-band attacks, or of the machine in general using ICMP smurfing for example)

The advice here is to update as often as possible and to be aware of any newly discovered bug. Many sites (like www.securityfocus.com or www.cert.com) will catalog all the existing bugs, exploits and fixes; some are even specialized on BIND.

D. Relevance and How to protect?

Most of those issues have been addressed with patches, new releases of BIND etc. But according to menandmice.com, the threat is still real:



A lot of other alarming surveys are available on this website (reference 3).

We hear a lot about DNSSEC, which is a proposition for secure DNS transactions (RFC 2535: http://www.ietf.org/rfc/rfc2535.txt). This appears to be a very good solution, which solves most of the protocol problems we discussed in this paper. However, due to backward compatibility reasons, this will not be implemented for quite some time, and current network administrators should think about other ways of securing their network against those threats.

There are a lot of things to do in order to secure a DNS server or group of servers. Reference 8 is very explicit on this subject and provides the ways to implement those recommendations on BIND.
We can quote some of those measures, especially the ones that match exactly the problems we have seen in this paper:

- Forbid recursive queries to prevent spoofing,
- Update BIND as often as possible to limit bug problems,
- To avoid having a single point of failure, do not put all DNS servers on the same subnet, or even behind the same router or the same leased line,
- Restrict the possible queries and the possible hosts who are allowed to query to the minimum.

We didn't talk about zone transfer, since this is a method used to gather information more than to really "hack". Issuing a specific request (AXFR), a hacker can retrieve the whole content of the DNS server database. This is usually used to synchronize the DNS secondary servers with the primary server. Doing this, the hacker can gather much information on the victim's network (number of hosts, names, addresses) as well as identify potential particular targets (servers like mail, DNS, etc.)
Of course, it is a good idea to turn this feature off, or at least restrict it to the specific hosts that need it.

A good idea of DNS architecture is what we wall "Split-Service" or "Split DNS architecture". The principle is as follows: we "split" the DNS system in two parts, one will be responsible for advertising the name-to-address mappings we are authoritative for, and the other is there to resolve the requests coming from the internal or, more generally, the "trusted" hosts. That way, if the external DNS is hacked, at least it won't affect the service provided to the internal hosts.

E. CONCLUSION:

DNS is not new, and DNS hacking has also been around for some time. The techniques to crack DNS keep getting better and DNS can't fight back using evolved techniques since it is still bound to maintain backwards compatibility.

That is why until IPv6 and all the associated security features it promises are there, we have to keep patching our versions of BIND.

F. References:

1 - Stevens, Glenn. “The Domain Name Service”. June 21, 1995. URL: http://eeunix.ee.usm.maine.edu/guides/dns/dns.html

2 - Kulapala, Beshan. “Recursive/Iterative Queries in DNS”. URL: http://wps.aw.com/wps/media/objects/221/227091/applets/dns/dns.html

3 – “www.menandmice.com”. “DNS Surveys”. 2002. URL: http://www.menandmice.com/6000/6000_domain_health.html

4 – “Gamma”. “DNS ID Prediction And Exploitation”. August 1998. URL: http://c0vertl.tripod.com/text/dnsattack.txt

5 – Erdfelt, Johannes. “Everything you ever wanted to know about DNS Spoofing”. July 25, 1997. URL: http://www.the-project.org/admins/0897-1097/0381.html

6 - UK Security Online Ltd. “Hacker threat - DNS Server attacks”. 2002. URL: http://www.uksecurityonline.com/threat/dns.php

7 – Carnegie Mellon University. “CERT Advisory CA-2001-02 Multiple Vulnerabilities in BIND”. Aug 2001. URL: http://www.cert.org/advisories/CA-2001-02.html

8 – Liu, Cricket. “Securing an Internet Name Server”. Feb 2001. URL: http://www.linuxsecurity.com/resource_files/server_security/securing_an_internet_name_server.pdf

Last Updated: December 12th, 2019

Upcoming SANS Training

Click here to view a list of all SANS Courses

SANS Miami 2020	Miami, FLUS	Jan 13, 2020 - Jan 18, 2020	Live Event
SANS Threat Hunting & IR Europe Summit & Training 2020	London, GB	Jan 13, 2020 - Jan 19, 2020	Live Event
Cyber Threat Intelligence Summit & Training 2020	Arlington, VAUS	Jan 20, 2020 - Jan 27, 2020	Live Event
SANS Tokyo January 2020	Tokyo, JP	Jan 20, 2020 - Jan 25, 2020	Live Event
SANS Anaheim 2020	Anaheim, CAUS	Jan 20, 2020 - Jan 25, 2020	Live Event
SANS Amsterdam January 2020	Amsterdam, NL	Jan 20, 2020 - Jan 25, 2020	Live Event
MGT521 Beta Two 2020	San Diego, CAUS	Jan 22, 2020 - Jan 23, 2020	Live Event
SANS Vienna January 2020	Vienna, AT	Jan 27, 2020 - Feb 01, 2020	Live Event
SANS San Francisco East Bay 2020	Emeryville, CAUS	Jan 27, 2020 - Feb 01, 2020	Live Event
SANS Las Vegas 2020	Las Vegas, NVUS	Jan 27, 2020 - Feb 01, 2020	Live Event
SANS Security East 2020	New Orleans, LAUS	Feb 01, 2020 - Feb 08, 2020	Live Event
SANS Northern VA - Fairfax 2020	Fairfax, VAUS	Feb 10, 2020 - Feb 15, 2020	Live Event
SANS New York City Winter 2020	New York City, NYUS	Feb 10, 2020 - Feb 15, 2020	Live Event
SANS London February 2020	London, GB	Feb 10, 2020 - Feb 15, 2020	Live Event
SANS Dubai February 2020	Dubai, AE	Feb 15, 2020 - Feb 20, 2020	Live Event
SANS Scottsdale 2020	Scottsdale, AZUS	Feb 17, 2020 - Feb 22, 2020	Live Event
SANS San Diego 2020	San Diego, CAUS	Feb 17, 2020 - Feb 22, 2020	Live Event
SANS Brussels February 2020	Brussels, BE	Feb 17, 2020 - Feb 22, 2020	Live Event
Open-Source Intelligence Summit & Training 2020	Alexandria, VAUS	Feb 18, 2020 - Feb 24, 2020	Live Event
SANS Training at RSA Conference 2020	San Francisco, CAUS	Feb 23, 2020 - Feb 24, 2020	Live Event
SANS Secure India 2020	Bangalore, IN	Feb 24, 2020 - Feb 29, 2020	Live Event
SANS Manchester February 2020	Manchester, GB	Feb 24, 2020 - Feb 29, 2020	Live Event
SANS Jacksonville 2020	Jacksonville, FLUS	Feb 24, 2020 - Feb 29, 2020	Live Event
SANS Zurich February 2020	Zurich, CH	Feb 24, 2020 - Feb 29, 2020	Live Event
SANS Secure Japan 2020	Tokyo, JP	Mar 02, 2020 - Mar 14, 2020	Live Event
ICS Security Summit & Training 2020	Orlando, FLUS	Mar 02, 2020 - Mar 09, 2020	Live Event
SANS Munich March 2020	Munich, DE	Mar 02, 2020 - Mar 07, 2020	Live Event
SANS Northern VA - Reston Spring 2020	Reston, VAUS	Mar 02, 2020 - Mar 07, 2020	Live Event
Blue Team Summit & Training 2020	Louisville, KYUS	Mar 02, 2020 - Mar 09, 2020	Live Event
SANS Jeddah March 2020	Jeddah, SA	Mar 07, 2020 - Mar 12, 2020	Live Event
SANS St. Louis 2020	St. Louis, MOUS	Mar 08, 2020 - Mar 13, 2020	Live Event
SANS Paris March 2020	Paris, FR	Mar 09, 2020 - Mar 14, 2020	Live Event
SANS Austin Winter 2020	OnlineTXUS	Jan 06, 2020 - Jan 11, 2020	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced