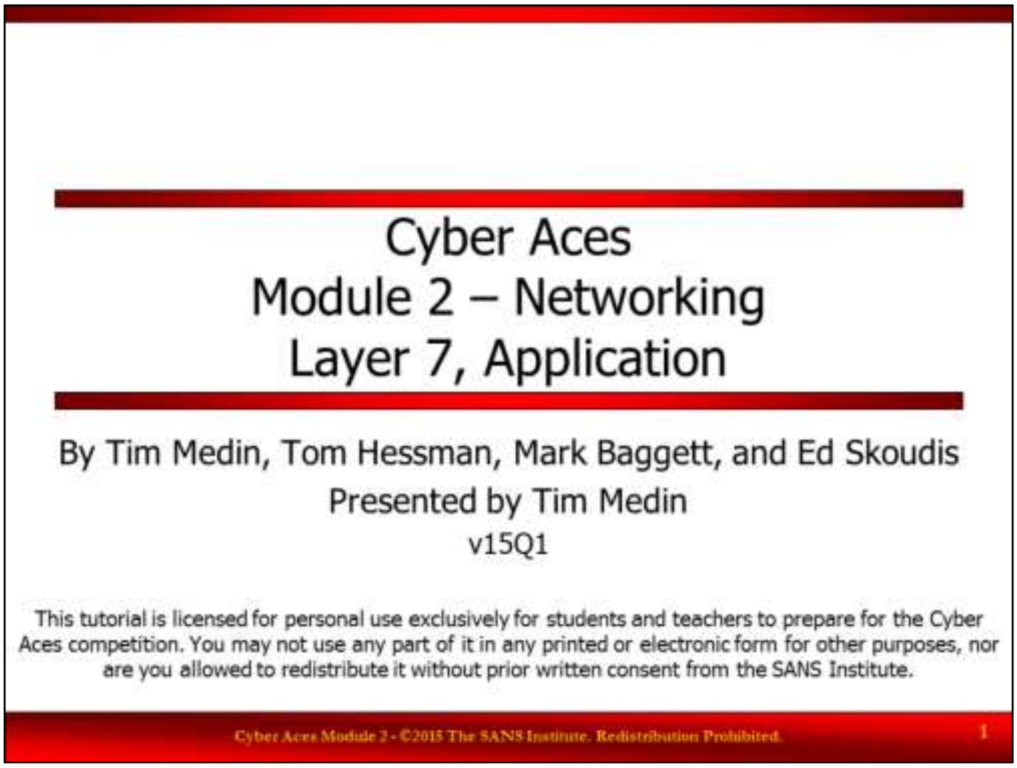


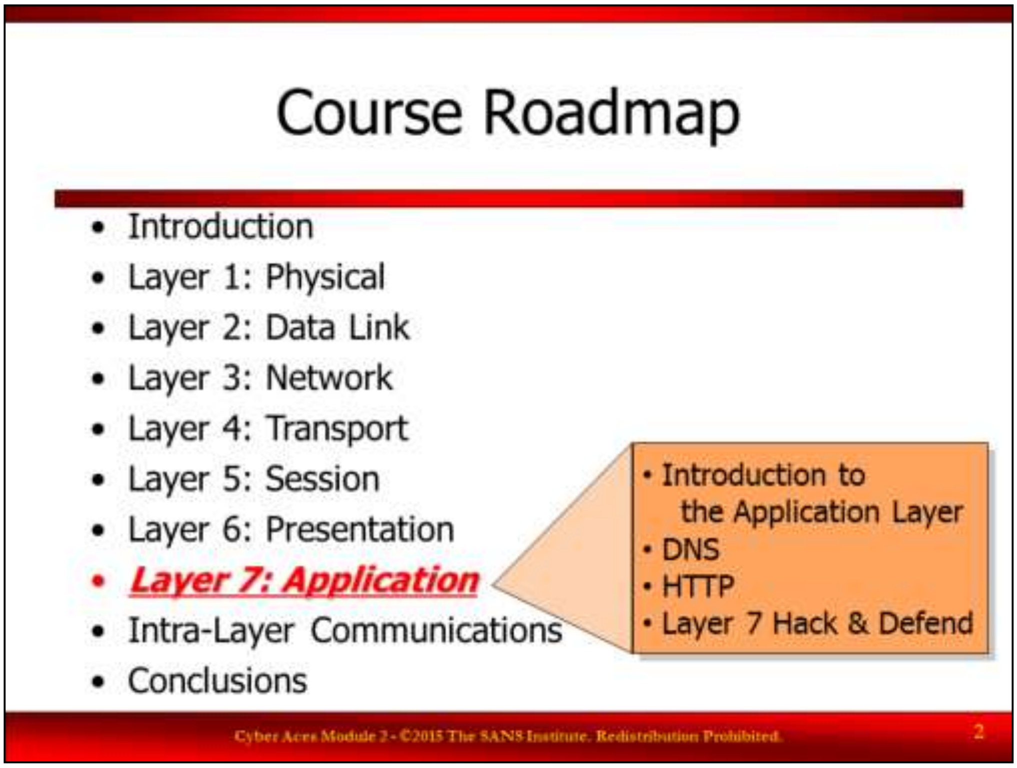
Cyber Aces
Module 2 – Networking
Layer 7, Application

By Tim Medin, Tom Hessman, Mark Baggett, and Ed Skoudis
Presented by Tim Medin

v15Q1

1

Welcome to Cyber Aces, Module 2! A firm understanding of network fundamentals is essential to being able to secure a network or attack one. This section provides a broad overview of networking, covering the fundamental concepts needed to understand computer attacks and defenses from a network perspective.



Course Roadmap

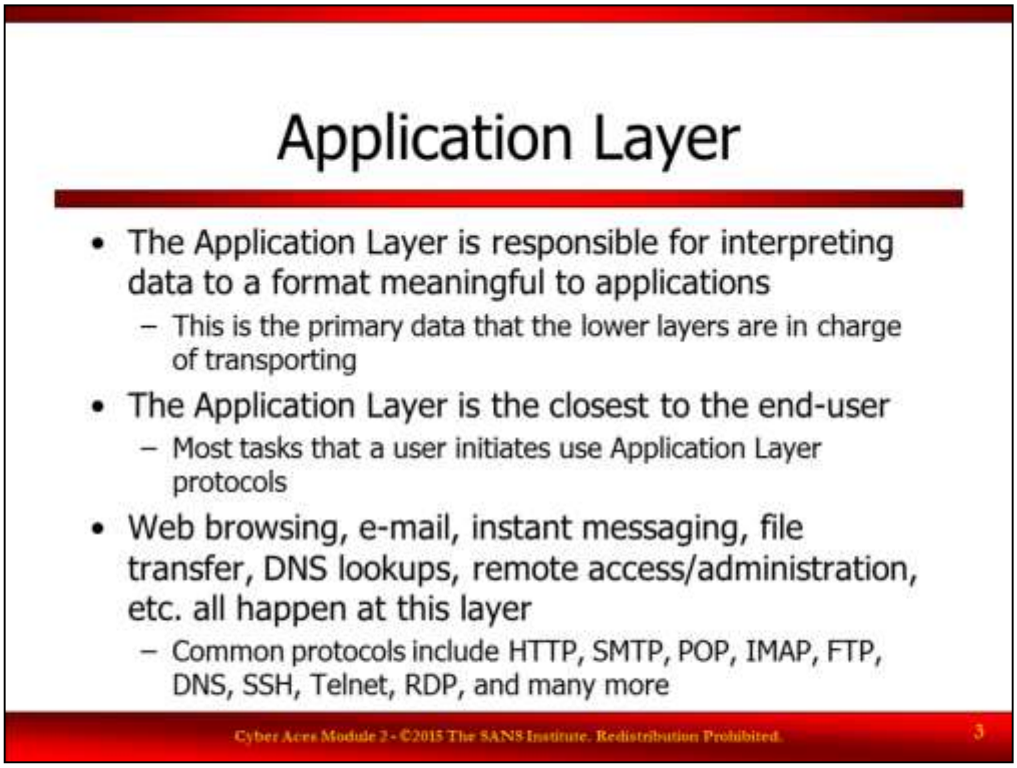
- Introduction
- Layer 1: Physical
- Layer 2: Data Link
- Layer 3: Network
- Layer 4: Transport
- Layer 5: Session
- Layer 6: Presentation
- ***Layer 7: Application***
- Intra-Layer Communications
- Conclusions

- Introduction to the Application Layer
- DNS
- HTTP
- Layer 7 Hack & Defend

Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 2

Course Roadmap

In this section, you'll learn about the Application Layer. We'll cover two important protocols operating at this layer, DNS and HTTP, which are both essential to the World Wide Web.

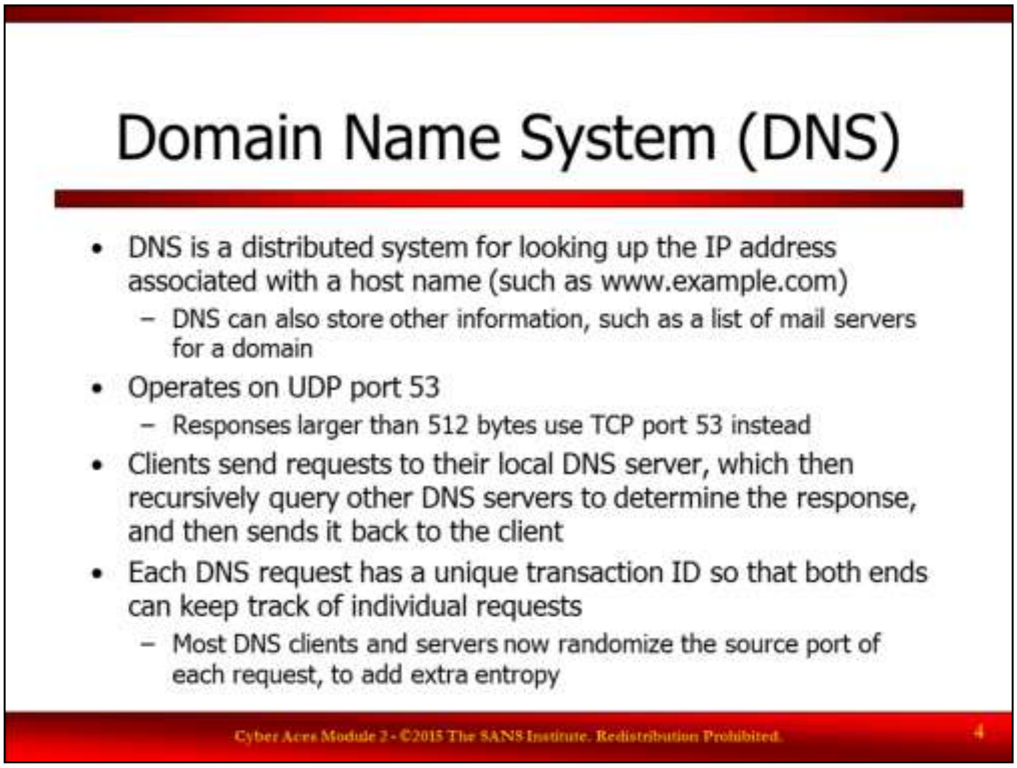

Application Layer

- The Application Layer is responsible for interpreting data to a format meaningful to applications
  - This is the primary data that the lower layers are in charge of transporting
- The Application Layer is the closest to the end-user
  - Most tasks that a user initiates use Application Layer protocols
- Web browsing, e-mail, instant messaging, file transfer, DNS lookups, remote access/administration, etc. all happen at this layer
  - Common protocols include HTTP, SMTP, POP, IMAP, FTP, DNS, SSH, Telnet, RDP, and many more

Application Layer

The Application Layer is responsible for interpreting data to a format that is meaningful to applications. Being the highest layer, this layer contains the data that the lower layers are in charge of transporting. The Application Layer is the closest to the end-user, and most tasks that a user initiates use Application Layer protocols. The Application Layer is the home of familiar protocols such as HTTP (Web), SSH (Secure Shell), SMTP (sending mail), POP3 and IMAP (receiving email), DNS (Domain Name Services) and many others.

This is where important functions such as retrieving web pages and resolving host names are implemented. Let's look at two important Application Layer protocols called HTTP and DNS. First, let's look at DNS.



Domain Name System (DNS)

- DNS is a distributed system for looking up the IP address associated with a host name (such as www.example.com)
  - DNS can also store other information, such as a list of mail servers for a domain
- Operates on UDP port 53
  - Responses larger than 512 bytes use TCP port 53 instead
- Clients send requests to their local DNS server, which then recursively query other DNS servers to determine the response, and then sends it back to the client
- Each DNS request has a unique transaction ID so that both ends can keep track of individual requests
  - Most DNS clients and servers now randomize the source port of each request, to add extra entropy

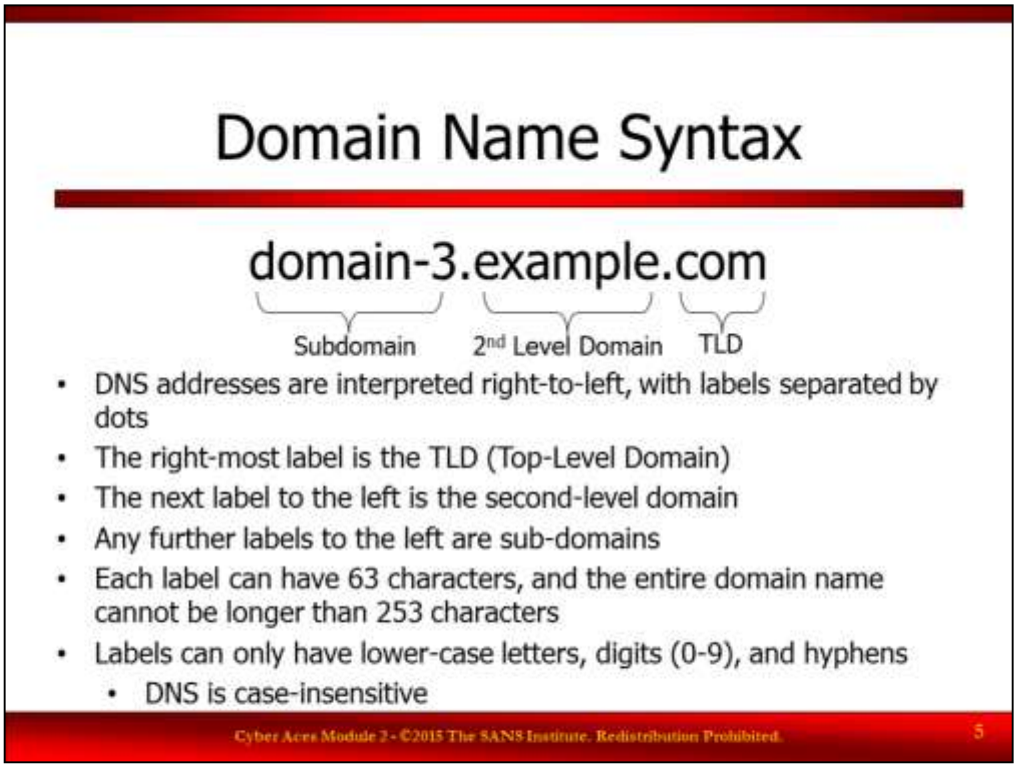
Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 4

Domain Name System (DNS)

Human beings remember names easier than we remember long strings of numbers. For example, most people can remember www.google.com but struggle to remember 74.125.159.147. DNS (Domain Name Services) maps human friendly domain names into computer friendly IP addresses. When we type a human friendly DNS name into a web browser, command prompt, or some other tool, our computer first asks a DNS server to translate that into an IP address. DNS is decentralized, meaning that there are multiple servers that can be queried for the same information (there is no single point of failure). It uses UDP port 53 to communicate, except for responses that are larger than 512 bytes (such as zone transfers), which use TCP port 53. Clients send requests to their local DNS server, which then recursively query other DNS servers to determine the response, and then the local server sends it back to the client.

Each DNS request has a unique transaction ID, allowing both ends to keep track of unique requests (since UDP is a stateless protocol). Since the transaction ID is only 2^{16} bits (0-65535), there is not a lot of entropy to help prevent against attacks. Most DNS clients and servers now randomize the source port of each DNS request and keep track of that too, in order to add more entropy and mitigate against transaction ID guessing (or brute forcing).

Reference Dan Kaminsky DNS Vulnerability: http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html



Domain Name Syntax

$\underbrace{\text{domain-3}}_{\text{Subdomain}}.\underbrace{\text{example}}_{\text{2nd Level Domain}}.\underbrace{\text{com}}_{\text{TLD}}$

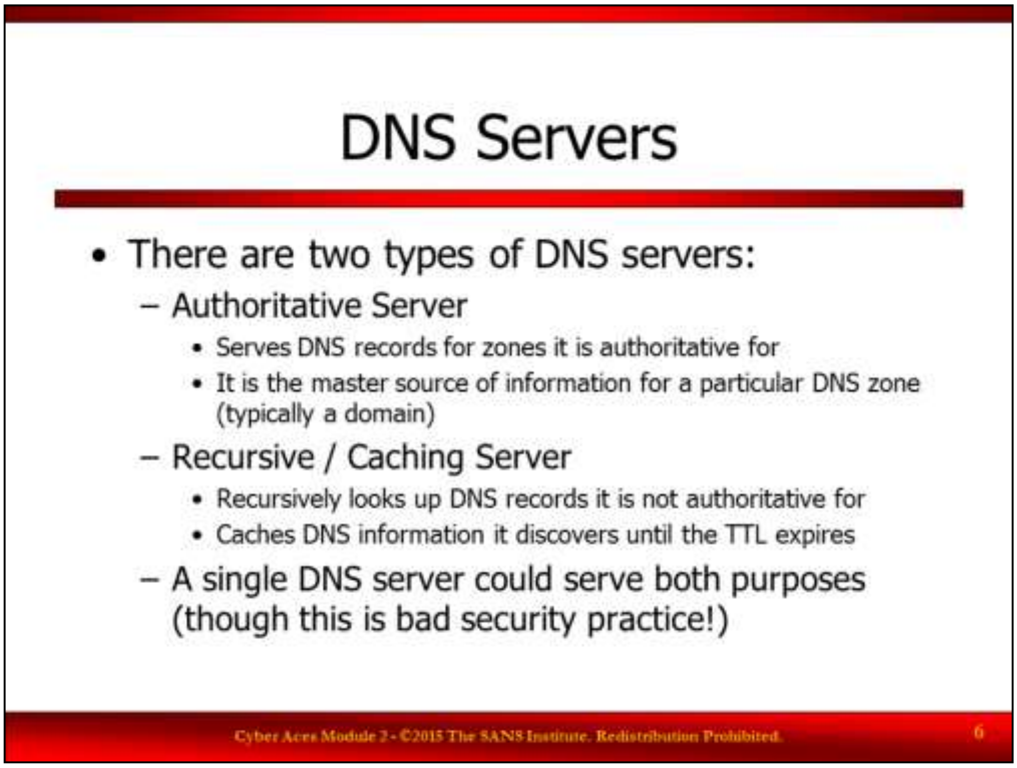
- DNS addresses are interpreted right-to-left, with labels separated by dots
- The right-most label is the TLD (Top-Level Domain)
- The next label to the left is the second-level domain
- Any further labels to the left are sub-domains
- Each label can have 63 characters, and the entire domain name cannot be longer than 253 characters
- Labels can only have lower-case letters, digits (0-9), and hyphens
  - DNS is case-insensitive

5

Domain Name Syntax

DNS has a certain syntax for domain names that must be follow. DNS addresses are interpreted right-to-left, with each "label" (component) of the address separated by dots. DNS is hierarchical, meaning the right-most labels are authoritative over the left-most ones, whereas the left-most labels are the most specific identifier. The right-most label is called the Top-Level Domain (TLD). ICANN defines a number of generic top-level domains (gTLD's), such as .com, .org, and .edu, as well as country code TLD's (ccTLD's), such as .us and .uk. The next label to the left is the second-level domain, which normally defines a specific entity on the Internet (such as "sans.org" for SANS). Note that in some TLD's, second-level domains are used as gTLD's, such as ".co.uk" (the UK equivlant of ".com"), meaning the third-level domain serves as the second-level domain. Any further labels to the left (such as the third-level domain) are sub-domains of the label to their right.

Each label can have a maximum of 63 characters, and the entire domain name cannot be longer than 253 characters. Labels can only contain lower-case letters, digits (0-9), and hyphens (though a label cannot start or end with a hyphen). DNS is case-insensitive, meaning that the domains "example.com" and "ExAmPlE.cOm" are the same.



DNS Servers

- There are two types of DNS servers:
  - Authoritative Server
    - Serves DNS records for zones it is authoritative for
    - It is the master source of information for a particular DNS zone (typically a domain)
  - Recursive / Caching Server
    - Recursively looks up DNS records it is not authoritative for
    - Caches DNS information it discovers until the TTL expires
  - A single DNS server could serve both purposes (though this is bad security practice!)

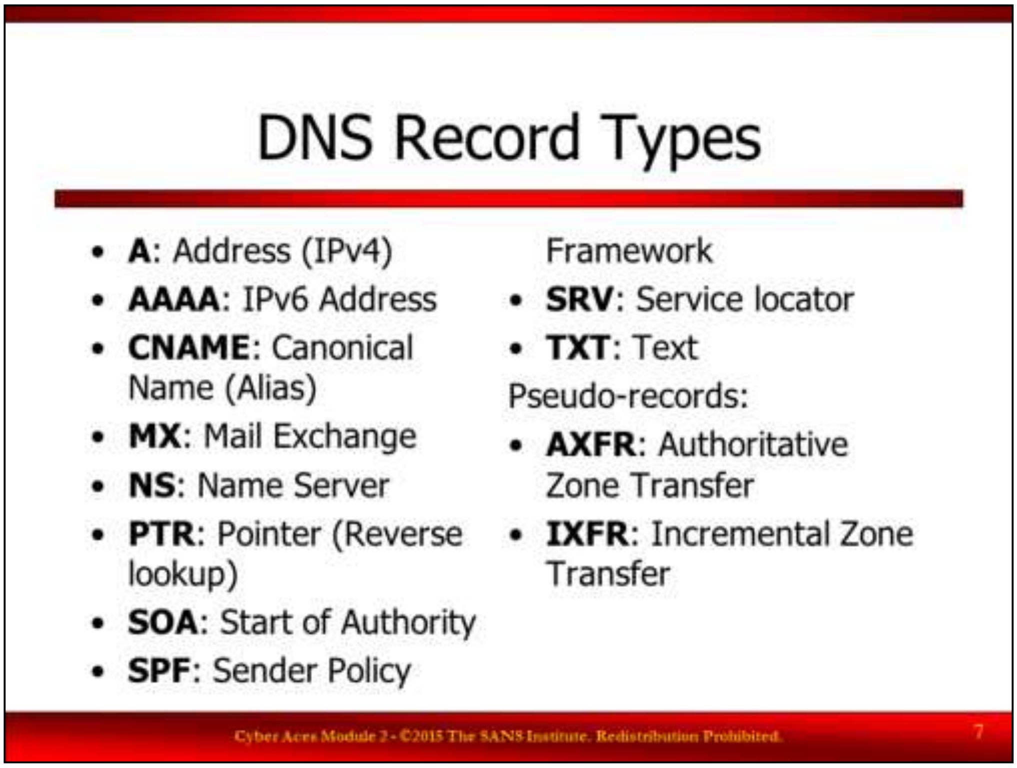
DNS Servers

There are two types of DNS servers: Authoritative and Recursive/Caching.

An authoritative server serves DNS records for zones it is the authoritative server for. (A *DNS zone* is a portion of DNS namespace that has had its administrative responsibility delegated to a specific entity.) An authoritative server serves as the master source of records for a particular zone, which is typically a particular second-level domain, though it could be more specific (or even broader). For example, a particular server could be authoritative for all of example.com (including all sub-domains), but particular sub-domains could also have their own authoritative servers. In fact, each TLD (managed by ICANN-accredited domain *registrars*) also has authoritative servers that contain records for the second-level domain servers, and there are 13 root name servers that contain records pointing to all of the TLD name servers. All domains generally have at least two authoritative name servers for redundancy.

A recursive server recursively looks up DNS records that it is NOT authoritative for. They are generally used by clients to perform DNS lookups, rather than every client in the world needing to manually query every authoritative server. A recursive server caches the DNS responses that it receives in order to speed up subsequent requests and reduce the amount of network traffic being generated. All DNS records specify a "TTL" (Time To Live) indicating how long the record should be kept in DNS caches before being purged. DNS changes can take a long time to propagate across the entire Internet, because the TTL has to expire on the record in every cache the record is stored in.

While a single DNS server could serve as both an authoritative server and a recursive server, this is bad security practice.



DNS Record Types

- **A**: Address (IPv4)
- **AAAA**: IPv6 Address
- **CNAME**: Canonical Name (Alias)
- **MX**: Mail Exchange
- **NS**: Name Server
- **PTR**: Pointer (Reverse lookup)
- **SOA**: Start of Authority
- **SPF**: Sender Policy Framework
- **SRV**: Service locator
- **TXT**: Text

Pseudo-records:

- **AXFR**: Authoritative Zone Transfer
- **IXFR**: Incremental Zone Transfer

Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 7

While most people think of DNS as simply being a phone book mapping names to IP addresses, DNS supports many types of resource records. Here are some of the most important ones to know:

A: Stores an IPv4 address that is associated with a given name. For example, asking for the A record for "example.com" returns "192.0.43.10".

AAAA: Stores an IPv6 address that is associated with a given name. For example, asking for the AAAA record for "example.com" returns "2001:500:88:200::10".

CNAME: Stores an alias, pointing one name to another name. The client should then look up the address of the new name. For example, if asking for "name.example.com" returns a CNAME record pointing to "realname.example.com", then these records share the same address record.

MX: Stores the names of mail servers for this domain, allowing e-mail to be routed to the correct place. There can be multiple MX records for a given domain, and they can specify a priority in which mail servers should be connected to.

NS: Stores the names of the authoritative name servers for the domain.

PTR: Stores the name associated with a given IP address. This is the reverse of an A (or AAAA) record, since it maps IP addresses to names.

SOA: Stores authoritative information about the domain, such as the primary name server, administrative contact, and how often records should be refreshed.

SPF: Stores Sender Policy Framework records. Sender Policy Framework is a means of defining mail servers that are authorized to send mail on behalf of the domain, which is used to help detect spam e-mail (which is generally sent from compromised systems or open mail relays).

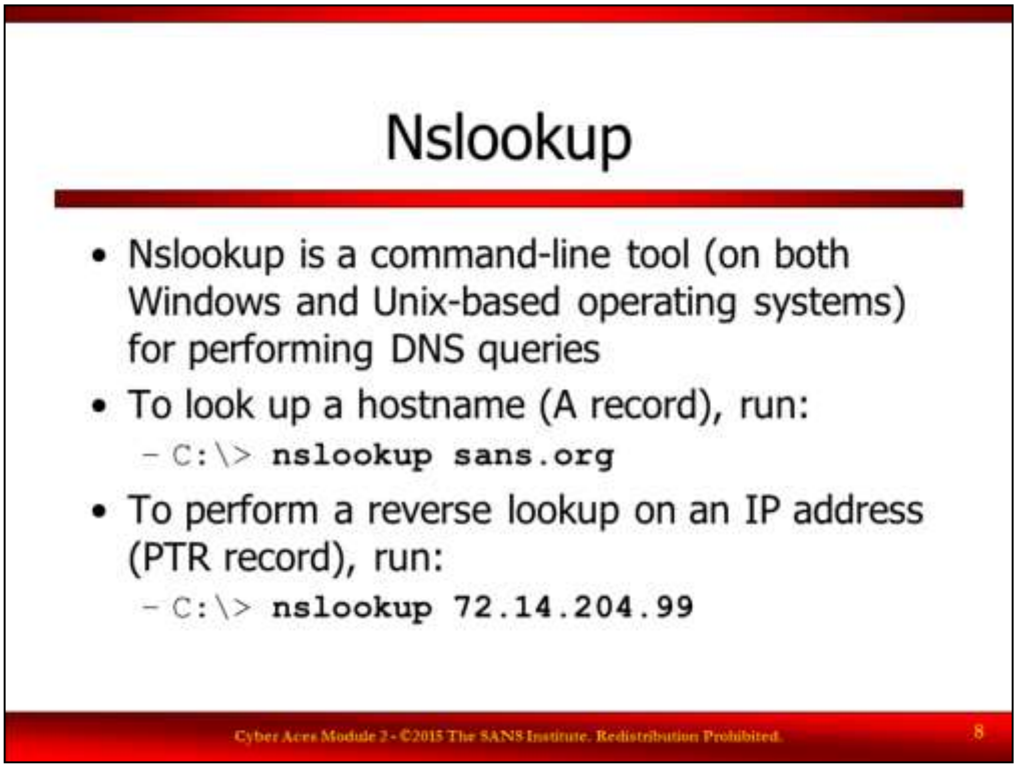
SRV: Used to store information about where services are located. This was defined so that service-specific records (such as MX) would not have to be defined in the future, though it is not widely used.

TXT: Stores arbitrary text. It commonly stores machine-readable information about a domain, such as SPF records (before the SPF record type was created).

There are also some pseudo-record types, that are more like commands:

AXFR: Performs an Authoritative Zone Transfer, which downloads a complete copy of all records in a DNS zone. This is intended to be used for backup (or "slave") servers to remain synchronized with the primary (or "master") server. Zone transfers should generally be blocked from public access, as the the DNS zone may contain sensitive information (such as a list of all servers on a domain).

IXFR: Performs an Incremental Zone Transfer, which downloads only the incremental changes to a DNS zone.



Nslookup

- Nslookup is a command-line tool (on both Windows and Unix-based operating systems) for performing DNS queries
- To look up a hostname (A record), run:
  - `C:\> nslookup sans.org`
- To perform a reverse lookup on an IP address (PTR record), run:
  - `C:\> nslookup 72.14.204.99`

Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 8

Nslookup

Nslookup is a command-line tool (on both Windows and Unix-based operating systems) for performing DNS queries. It is a great tool to use for diagnosing DNS problems, or for getting more information about a domain or IP address.

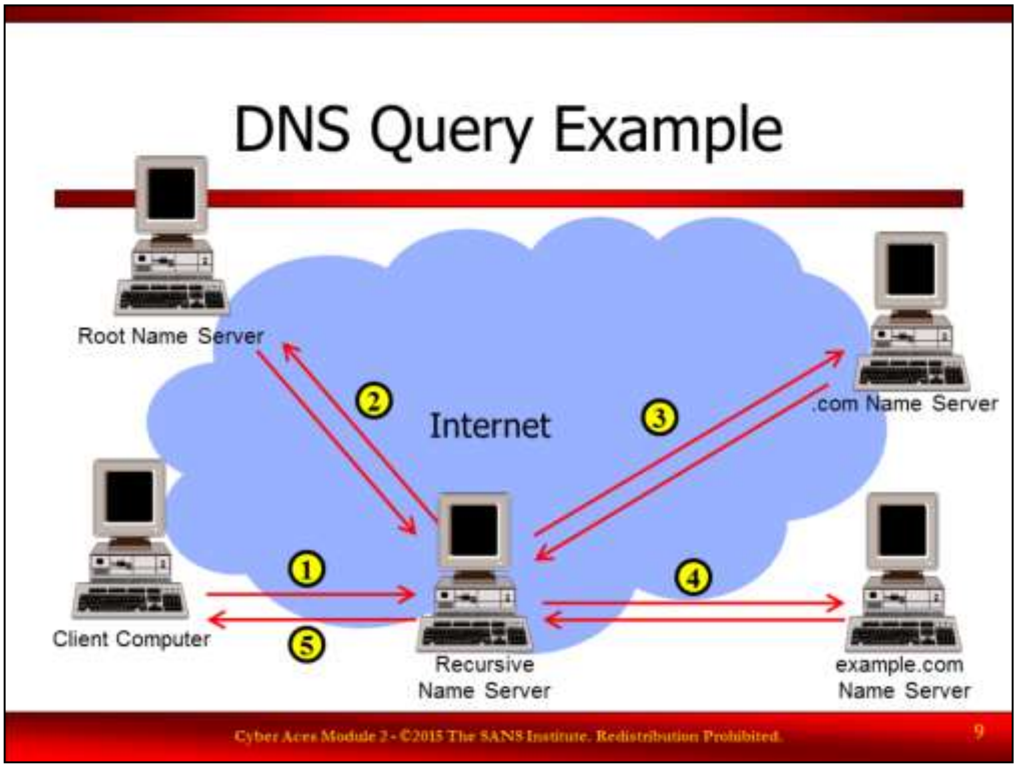
To look up a hostname (A record), run:

```
C:\> nslookup sans.org
```

To perform a reverse lookup on an IP address (PTR record), run:

```
C:\> nslookup 72.14.204.99
```

Nslookup can also look up other types of DNS records, and can query any arbitrary name server (it queries your default name server by default).

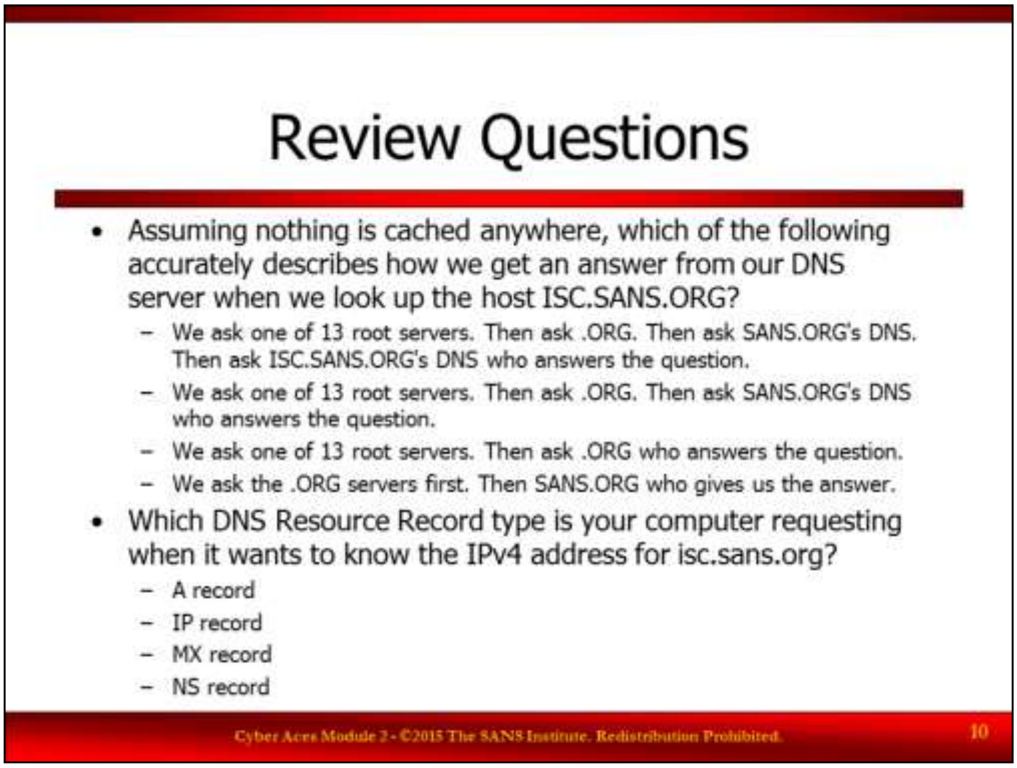


DNS Query Example

DNS Query Example

Here is an example of a complete DNS transaction, in which none of the servers have anything cached. In Step 1, the client computer sends a DNS request to its local recursive name server asking for the A record for "example.com". In Step 2, the recursive name server asks one of the root name servers for the address of the ".com" name server, and it receives a response. In Step 3, the recursive name server asks the ".com" name server for the address of the "example.com" name server, and it receives a response. In Step 4, the recursive name server asks the "example.com" name server for the address of "example.com", and it receives a response. Finally, in Step 5, the recursive name server sends a response to the client computer containing the address of "example.com".

Of course, chances are that the recursive server would have had the address of the ".com" name server cached, and may have had the address for "example.com" and/or the address of its name server cached, greatly simplifying this process. Also, in reality, there are multiple authoritative servers for ".com", and for "example.com".



Review Questions

- Assuming nothing is cached anywhere, which of the following accurately describes how we get an answer from our DNS server when we look up the host ISC.SANS.ORG?
  - We ask one of 13 root servers. Then ask .ORG. Then ask SANS.ORG's DNS. Then ask ISC.SANS.ORG's DNS who answers the question.
  - We ask one of 13 root servers. Then ask .ORG. Then ask SANS.ORG's DNS who answers the question.
  - We ask one of 13 root servers. Then ask .ORG who answers the question.
  - We ask the .ORG servers first. Then SANS.ORG who gives us the answer.
- Which DNS Resource Record type is your computer requesting when it wants to know the IPv4 address for isc.sans.org?
  - A record
  - IP record
  - MX record
  - NS record

10

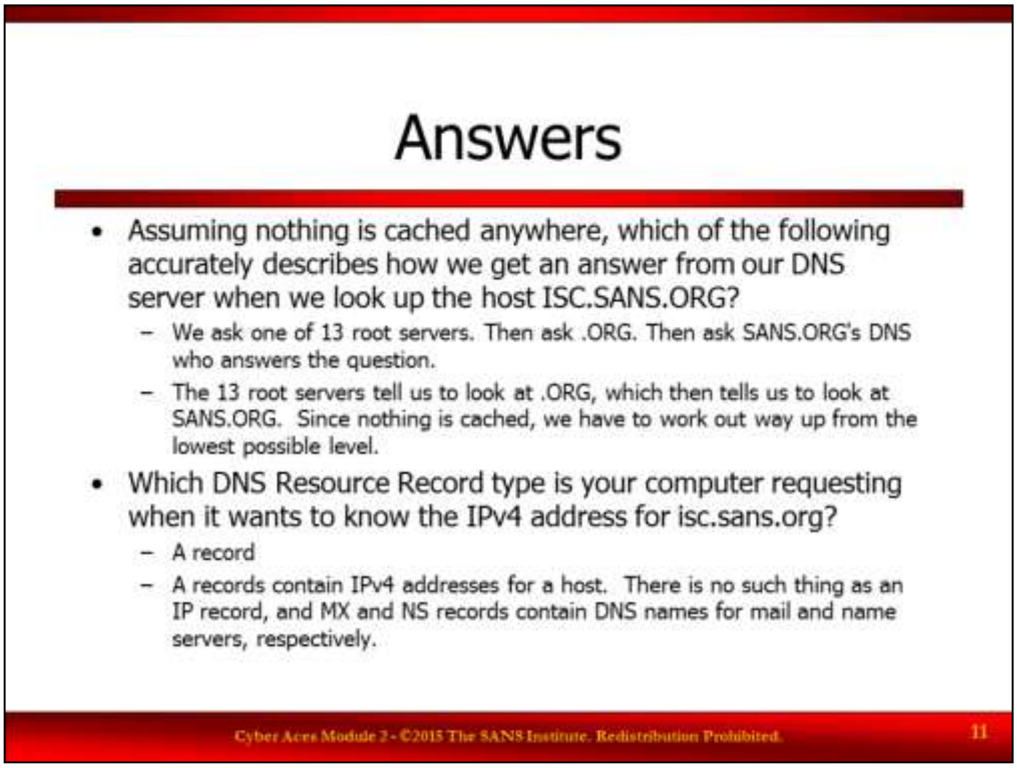
Review Questions

Assuming nothing is cached anywhere, which of the following accurately describes how we get an answer from our DNS server when we look up the host ISC.SANS.ORG?

- •We ask one of 13 root servers. Then ask .ORG. Then ask SANS.ORG's DNS. Then ask ISC.SANS.ORG's DNS who answers the question.
- •We ask one of 13 root servers. Then ask .ORG. Then ask SANS.ORG's DNS who answers the question.
- •We ask one of 13 root servers. Then ask .ORG who answers the question.
- •We ask the .ORG servers first. Then SANS.ORG who gives us the answer.

Which DNS Resource Record type is your computer requesting when it wants to know the IPv4 address for isc.sans.org?

- •A record
- •IP record
- •MX record
- •NS record



Answers

- Assuming nothing is cached anywhere, which of the following accurately describes how we get an answer from our DNS server when we look up the host ISC.SANS.ORG?
  - We ask one of 13 root servers. Then ask .ORG. Then ask SANS.ORG's DNS who answers the question.
  - The 13 root servers tell us to look at .ORG, which then tells us to look at SANS.ORG. Since nothing is cached, we have to work out way up from the lowest possible level.
- Which DNS Resource Record type is your computer requesting when it wants to know the IPv4 address for isc.sans.org?
  - A record
  - A records contain IPv4 addresses for a host. There is no such thing as an IP record, and MX and NS records contain DNS names for mail and name servers, respectively.

11

Answers

Assuming nothing is cached anywhere, which of the following accurately describes how we get an answer from our DNS server when we look up the host ISC.SANS.ORG?

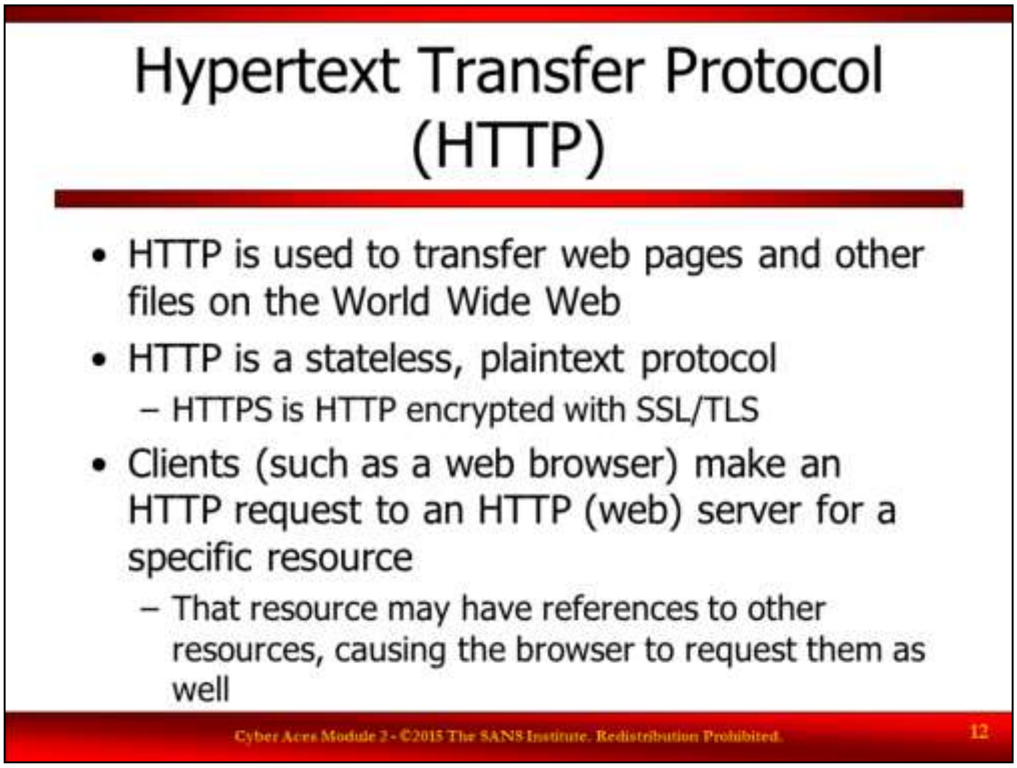
•We ask one of 13 root servers. Then ask .ORG. Then ask SANS.ORG's DNS who answers the question.

•The 13 root servers tell us to look at .ORG, which then tells us to look at SANS.ORG. Since nothing is cached, we have to work out way up from the lowest possible level.

Which DNS Resource Record type is your computer requesting when it wants to know the IPv4 address for isc.sans.org?

•A record

•A records contain IPv4 addresses for a host. There is no such thing as an IP record, and MX and NS records contain DNS names for mail and name servers, respectively.



Hypertext Transfer Protocol (HTTP)

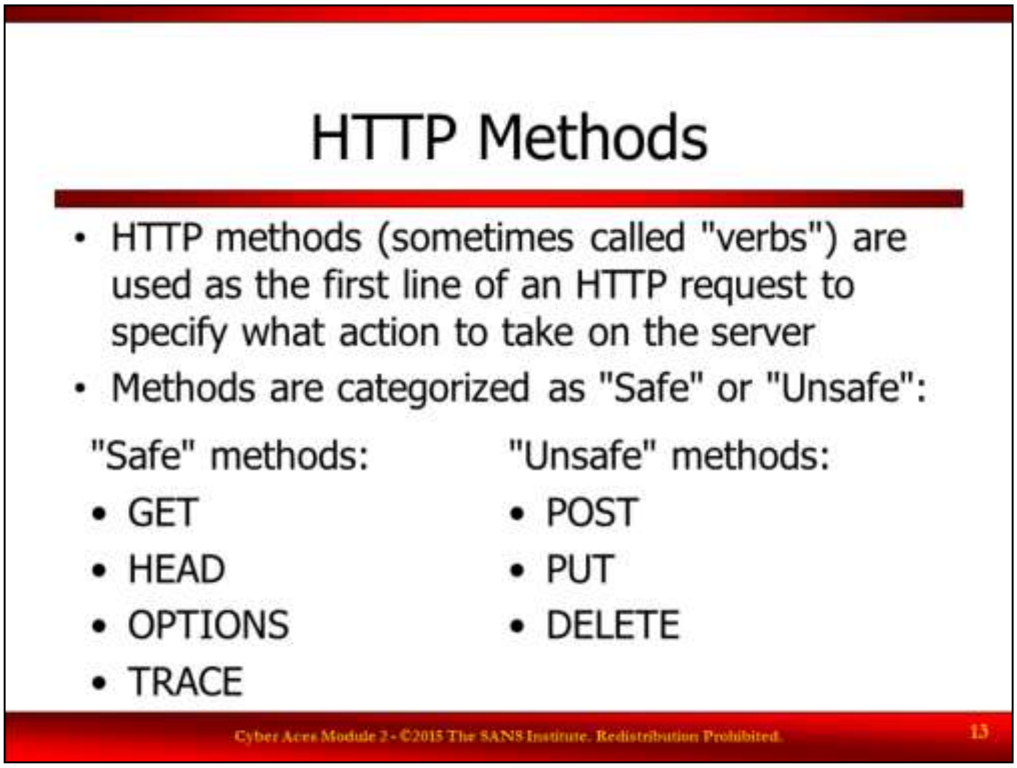
- HTTP is used to transfer web pages and other files on the World Wide Web
- HTTP is a stateless, plaintext protocol
  - HTTPS is HTTP encrypted with SSL/TLS
- Clients (such as a web browser) make an HTTP request to an HTTP (web) server for a specific resource
  - That resource may have references to other resources, causing the browser to request them as well

Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 12

Hypertext Transfer Protocol (HTTP)

HTTP, the Hypertext Transfer Protocol, is used to transfer web pages and other files on the World Wide Web. It was originally designed to connect from a web browser to a web server and retrieve an HTML file, though its use and functionality has been greatly extended. HTTP is a stateless protocol, meaning that the client and server have no built-in way to maintain a session; each request stands on its own. HTTP is also a plaintext, almost human-readable protocol, making it unsuitable for sensitive data. HTTPS is the HTTP protocol encrypted using SSL/TLS encryption, which is used for secure transactions such as online banking.

Client software, such as a web browser, uses HTTP to make an HTTP request to an HTTP (web) server, asking for a specific resource (such as a particular file). The server then responds (with an HTTP response) with that resource. That resource (such as an HTML file) may have references to other resources (such as images, scripts, or stylesheets), which the client will then make further requests for.



HTTP Methods

- HTTP methods (sometimes called "verbs") are used as the first line of an HTTP request to specify what action to take on the server
- Methods are categorized as "Safe" or "Unsafe":

"Safe" methods:	"Unsafe" methods:
• GET	• POST
• HEAD	• PUT
• OPTIONS	• DELETE
• TRACE	

HTTP Methods

HTTP methods, sometimes called "verbs", make up the first line of an HTTP request. They specify what action to take on a specified resource on the server. For example, "`GET /file.html HTTP/1.1`" tells the server to GET the file at "/file.html" using HTTP version 1.1.

Methods can be categorized as either "safe" or "unsafe", based on whether or not they would change anything on the server (such as changing a file). Safe methods are not supposed to change any state on the server, making them "safe" to call without accidentally performing actions (such as making a purchase or deleting a record). Unsafe methods are meant to be used in cases that would change the server's state, such as submitting a form. Below is an overview of a few of the most important methods:

Safe methods:

GET: Requests a copy of the specified resource. This is the most commonly used method, which simply downloads a file (typically a web page). Any data that is passed along to the resource is included as part of the URL, such as: http://www.example.com/page.html?data=value

HEAD: Requests a copy of the HTTP headers that would be sent if a GET request was issued for this particular resource. This is used to obtain information about a resource (such as when it was last modified) without actually downloading the resource itself.

OPTIONS: Returns the HTTP methods that the server supports for the specified URL, or that the server itself supports if "*" is requested.

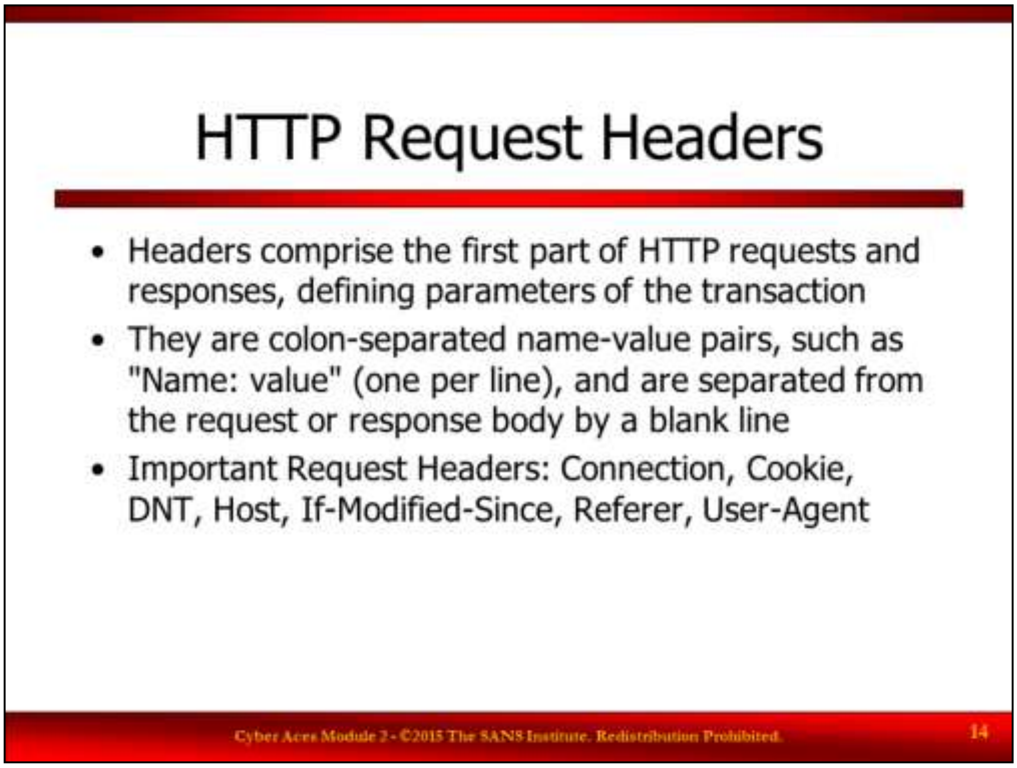
TRACE: Echoes back the request, which can be used to debug connection problems between the client and the server. Trace should generally be disabled on web servers to mitigate a potential XSS attack vector.

Unsafe methods:

POST: Submits data to the specified resource for processing. This is used to submit forms to complete a transaction (such as logging in to a server, making a purchase, or posting a comment on a blog), and can also be used to upload files to the server. The data is sent as part of the request body, instead of as part of the URL, and can therefore be of an arbitrary length.

PUT: Uploads a copy of a resource to the server. This is not widely used on standard HTTP servers.

DELETE: Deletes the specified resource from the server. This is not widely used on standard HTTP servers.



HTTP Request Headers

- Headers comprise the first part of HTTP requests and responses, defining parameters of the transaction
- They are colon-separated name-value pairs, such as "Name: value" (one per line), and are separated from the request or response body by a blank line
- Important Request Headers: Connection, Cookie, DNT, Host, If-Modified-Since, Referer, User-Agent

Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 14

HTTP Request Headers

HTTP Headers comprise the first part of HTTP requests and responses, and are used to define additional parameters of the transaction (beyond the initial method or status code). They are colon-separated name-value pairs, such as "Name: value", one per line. They are separated from the request or response body by a blank line. While headers are considered optional (except for the "Host" header), they often contain important and useful information. Let's review a small sampling of important headers used as part of HTTP requests:

Request Headers:

Connection: Specifies whether the client would prefer for the server to close the connection when it is done or keep it open (which can be useful for requesting multiple resources concurrently).

Cookie: Contains any existing cookies previously set by the server. If the client is sending two cookies, one called "name1" with a value of "value1" and one called "name2" with a value of "value2", the header would look like: "`Cookie: name1=value1; name2=value2;`"

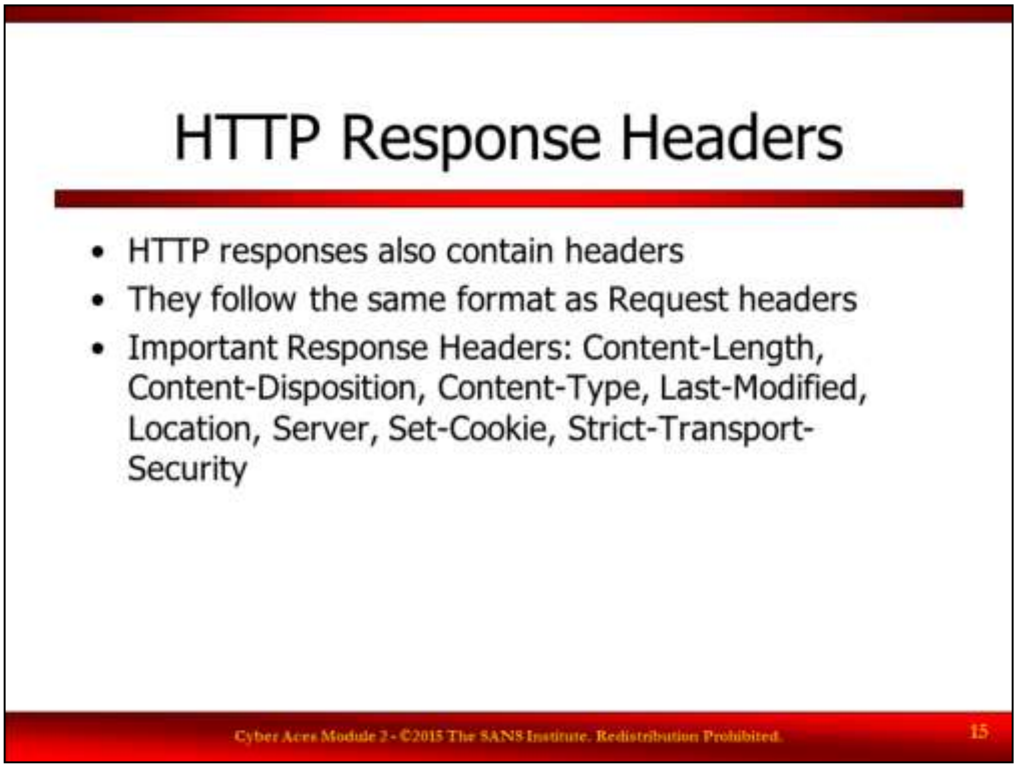
DNT: Specifies that the client wishes to opt-out of being tracked by the web application. This is a new specification that has not yet been widely adopted.

Host: Specifies the domain name of the server to request the resource from. As of HTTP 1.1, this header is required for all requests. The use of this header allows for a single server to serve pages for multiple domains, which is called "virtual hosting".

If-Modified-Since: Specifies that the server should only send the specified resources if it has not been modified since the specified date (if it hasn't been modified, the server will answer with status code 304). This is commonly used by web browsers to only download a new copy of a resource if it is newer than the copy that it has cached.

Referer: Specifies the URL of the resource that led to this request. For example, if a user clicked a link leading to the page that is being requested, the referer header will contain the URL of the page that contained the link. The referer is also set by browsers when requesting resources on a page, such as images. Servers can use this to track users across a site, or to find out how they first reached their site. Clicking a link in a set of search results, for example, will usually set the referer header to the path of the search, often containing the search terms. The referer is not set when following a link from an HTTPS site to an HTTP site. Please note that "referer" is intentionally spelled wrong, as that is the spelling in the specification.

User-Agent: Contains information about the client software making the request, such as the name and version of the web browser being used.



HTTP Response Headers

- HTTP responses also contain headers
- They follow the same format as Request headers
- Important Response Headers: Content-Length, Content-Disposition, Content-Type, Last-Modified, Location, Server, Set-Cookie, Strict-Transport-Security

Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 15

HTTP Response Headers

HTTP responses also contain headers, which follow the same format as Request headers. Let's review a small sampling of important headers used as part of HTTP responses:

Content-Length: The length (or size) of the response specified in bytes.

Content-Disposition: Specifies that a file should be downloaded instead of displayed inline, and also specifies what file name should be used when downloading the file.

Content-Type: Specifies the MIME type of the resource being sent. Most types of files have an associated MIME type, such as "text/plain" for a plain text file, "text/html" for an HTML document, and "image/jpeg" for a JPEG image. The MIME type "application/octet-stream" is used for generic binary (non-text) files.

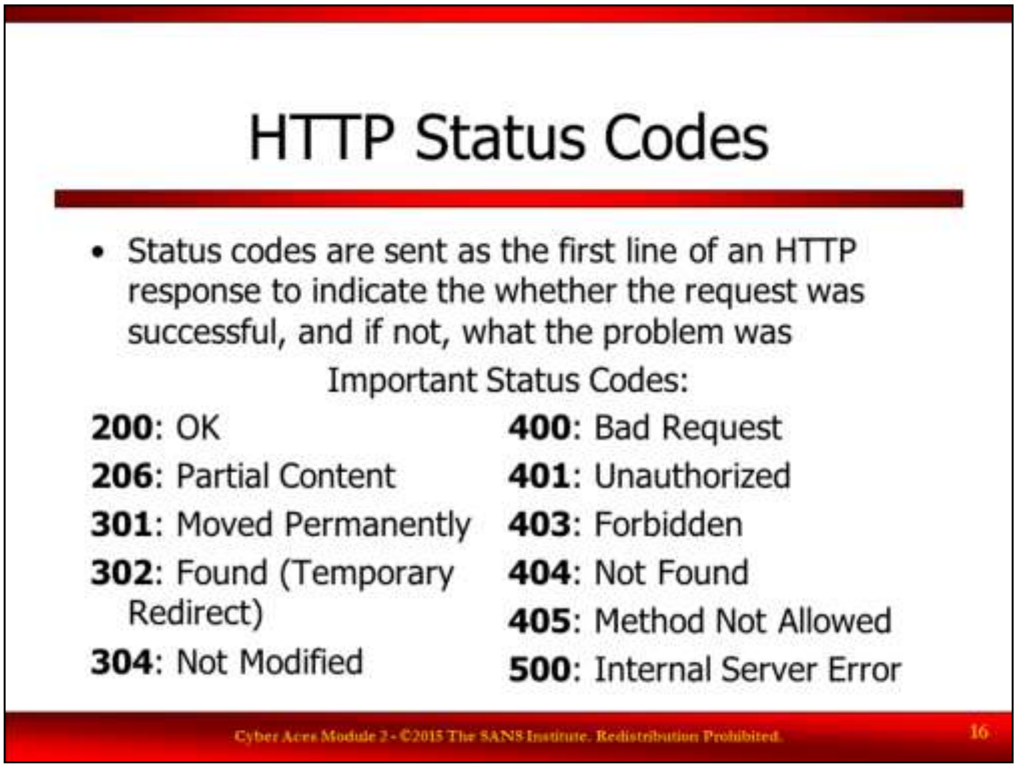
Last-Modified: Specifies the date the resource being requested was last modified.

Location: When redirecting a request to another location, this header specifies the location the client should be sent to.

Server: Contains information about the server software sending the response, such as the name version of the server being used.

Set-Cookie: Sets an HTTP cookie in the client's browser.

Strict-Transport-Security: Specifies that the client should only connect to this server using HTTPS, never with an unencrypted HTTP connection. This is a new standard designed to help prevent certain types of attacks against secure websites. Most browsers support it now, but not many servers use it yet.


HTTP Status Codes

- Status codes are sent as the first line of an HTTP response to indicate the whether the request was successful, and if not, what the problem was

Important Status Codes:

200: OK	**400**: Bad Request
206: Partial Content	**401**: Unauthorized
301: Moved Permanently	**403**: Forbidden
302: Found (Temporary Redirect)	**404**: Not Found
	405: Method Not Allowed
304: Not Modified	**500**: Internal Server Error

Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 16

HTTP Status Codes

HTTP status codes are sent as the first line of an HTTP response. They are used to indicate to the client whether the request was successful, and if not, what the problem was. Status codes are split into groups based on whether they indicate success (2XX), redirection (3XX), a client error (4XX), or a server error (5XX). While there are many status codes defined, only a small useful are commonly seen during most transactions. Let's review a few of the most important ones:

200: The request was successful, and the server is sending the requested resource as part of the response.

206: The server is only delivering the part of the resource that the client requested. This is used for clients to be able to resume interrupted downloads, for example, without having to start over.

301: This redirects the client to a different URL, and indicates that the new URL should always be used in the future.

302: This redirects the client to a different URL, and indicates that the redirect is not permanent (meaning the old URL should still be used). Status codes 303 and 307 are better suited for this purpose, but most implementations use 302.

304: The resource has not been modified since the client last requested it (typically sent in response to an "If-Modified-Since" header).

400: The client's request was malformed, and the server could not understand it.

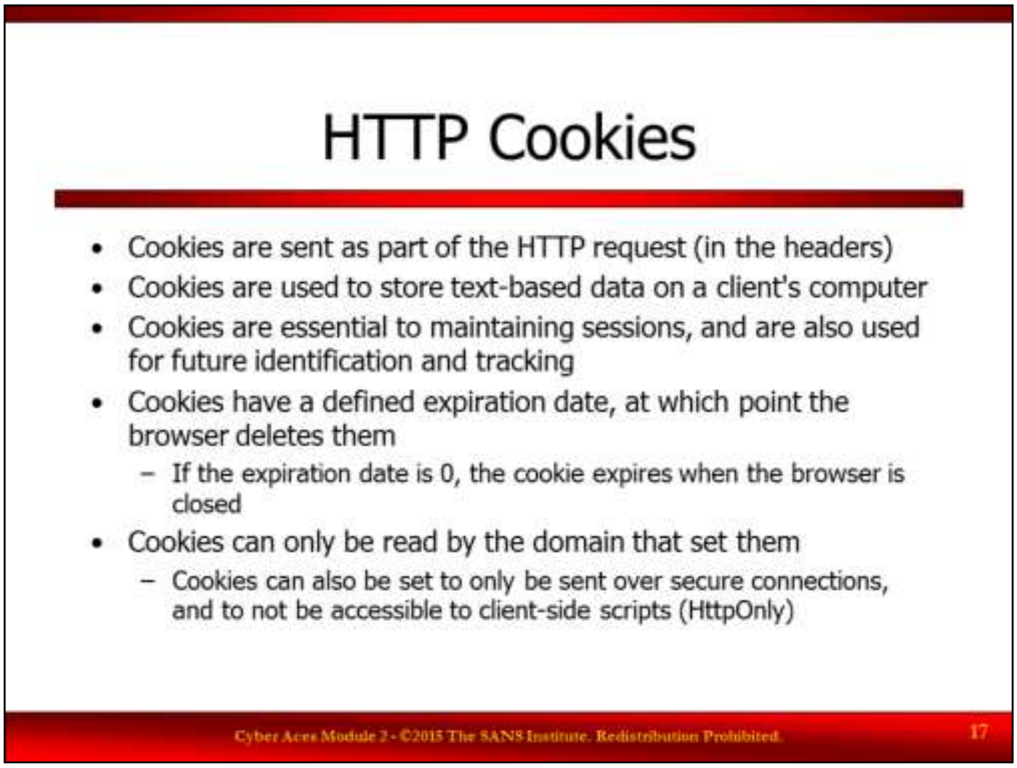
401: The client needs to supply authentication credentials, most likely a username and password. There are two types of authentication supported. Basic authentication simply encodes the username and password using base64 encoding, which can easily be decoded by an attacker. Digest authentication uses an MD5-based challenge-response. Note that most popular web applications use their own authentication methods using web forms and cookies, not this mechanism.

403: The client is not authorized to access this resource, and providing authentication will not help. This is typically used to restrict access by IP address.

404: The requested resource could not be found on the server. There is also a related status code, 410, which indicates a resource no longer exists, though it is not commonly used.

405: The HTTP method requested is not allowed. For example, using GET on a form which requires POST could trigger this, and many servers disable TRACE to mitigate a potential XSS attack vector.

500: The server has suffered some sort of internal error that is preventing it from responding to the request. This if often triggered by misconfiguration on the server, but it could be triggered by a number of events.



HTTP Cookies

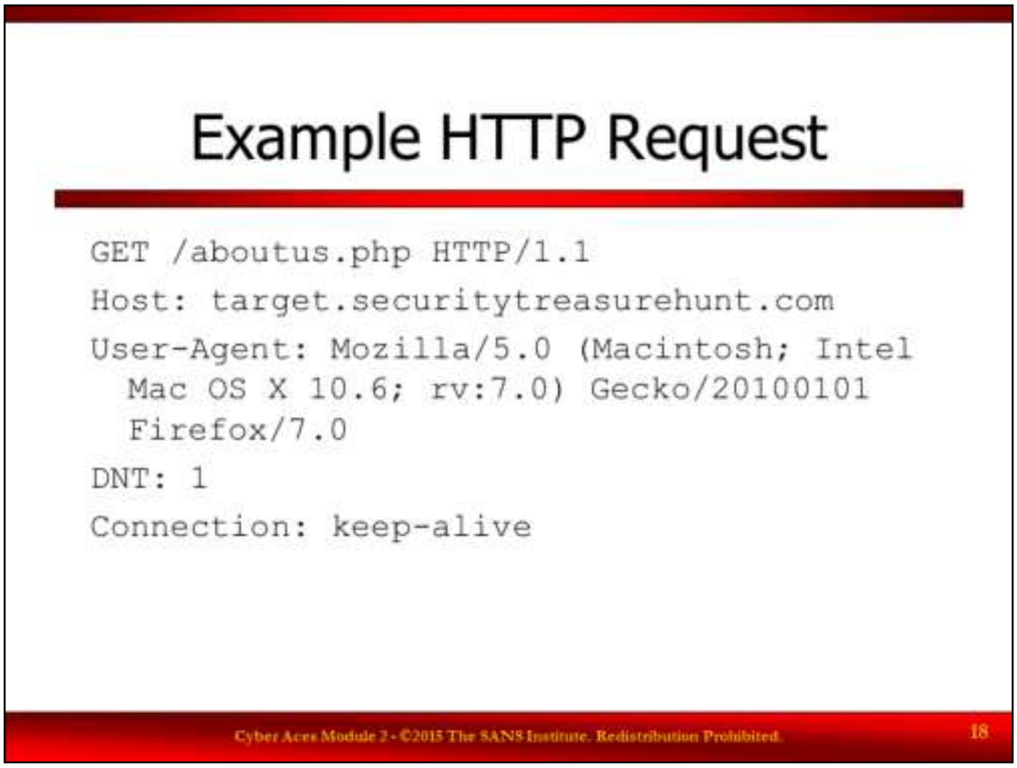
- Cookies are sent as part of the HTTP request (in the headers)
- Cookies are used to store text-based data on a client's computer
- Cookies are essential to maintaining sessions, and are also used for future identification and tracking
- Cookies have a defined expiration date, at which point the browser deletes them
  - If the expiration date is 0, the cookie expires when the browser is closed
- Cookies can only be read by the domain that set them
  - Cookies can also be set to only be sent over secure connections, and to not be accessible to client-side scripts (HttpOnly)

17

HTTP Cookies

Cookies are used to store data on a client's computer, and are included in the HTTP headers of each request sent to the same server. Since HTTP is a stateless protocol, cookies are essential to maintaining session state between a client and the server. They are also used to identify particular clients on future visits, and to track particular clients across one or more web sites. For this reason, there are privacy concerns surrounding cookies, and most browsers provide an easy way to clear stored cookies. Cookies can only store plain text, not executable code.

Cookies have a defined expiration date (which could be far in the future), at which the point the browser deletes them. If the expiration date is 0, the cookie expires when the browser is closed (this is called a "session cookie"). Cookies are subject to the "same-origin policy", meaning they can only be read by the domain that set them. Modern browsers support a flag indicating that a particular cookie can only be sent over an SSL-encrypted connection (the "secure" flag), and most browsers also support a new flag that blocks cookies from being read by client-side scripting languages (the "HttpOnly" flag).

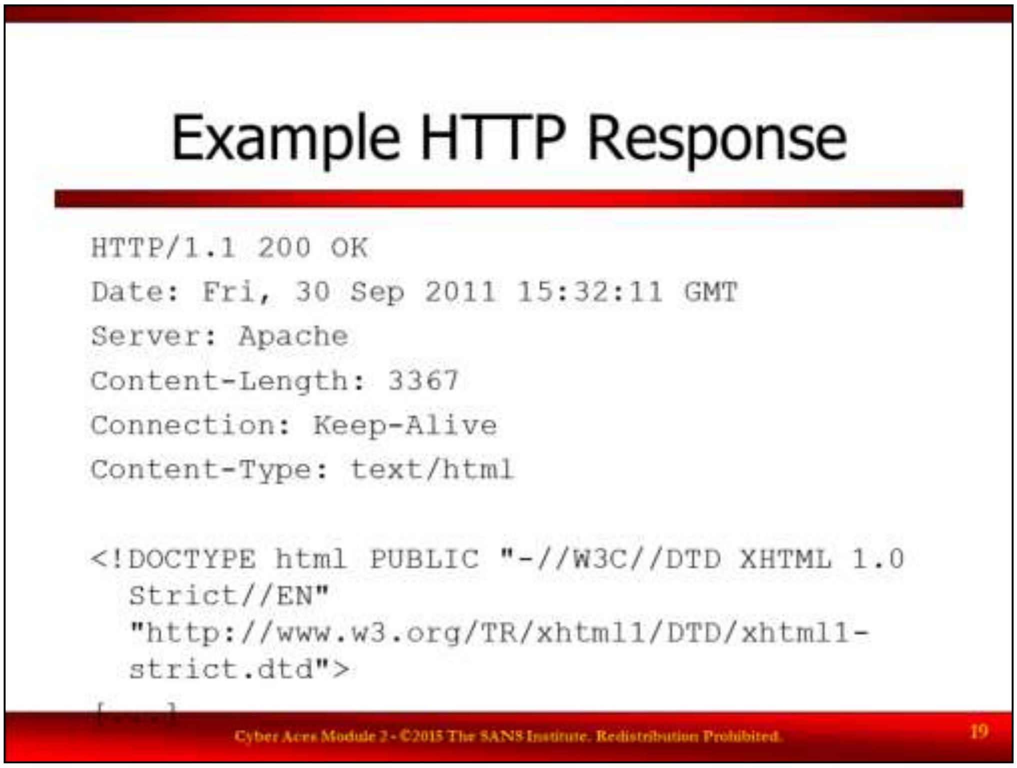


Example HTTP Request

```
GET /aboutus.php HTTP/1.1
Host: target.securitytreasurehunt.com
User-Agent: Mozilla/5.0 (Macintosh; Intel
  Mac OS X 10.6; rv:7.0) Gecko/20100101
  Firefox/7.0
DNT: 1
Connection: keep-alive
```

Example HTTP Request

Above is an example HTTP request made by the Firefox browser. Note that we've removed a few of the unimportant headers for brevity. The URL being requested is "http://target.securitytreasurehunt.com/aboutus.php". The first line contains the request method ("GET") followed by a space, then the path to the resource being requested ("/aboutus.php"), another space, and the protocol and version ("HTTP/1.1"). The next line contains the "Host" header, indication the name of the host ("target.securitytreasurehunt.com"). The third line contains the "User-Agent" header, which contains the client's browser's user agent string (above, the client is running Firefox 7.0 on Mac OS X 10.6). Finally, we have the "DNT" header (telling the server not to track the client's activities), and the "Connection" header (telling the server to keep the connection open). The request ends with a blank line, telling the server that the request is complete.



Example HTTP Response

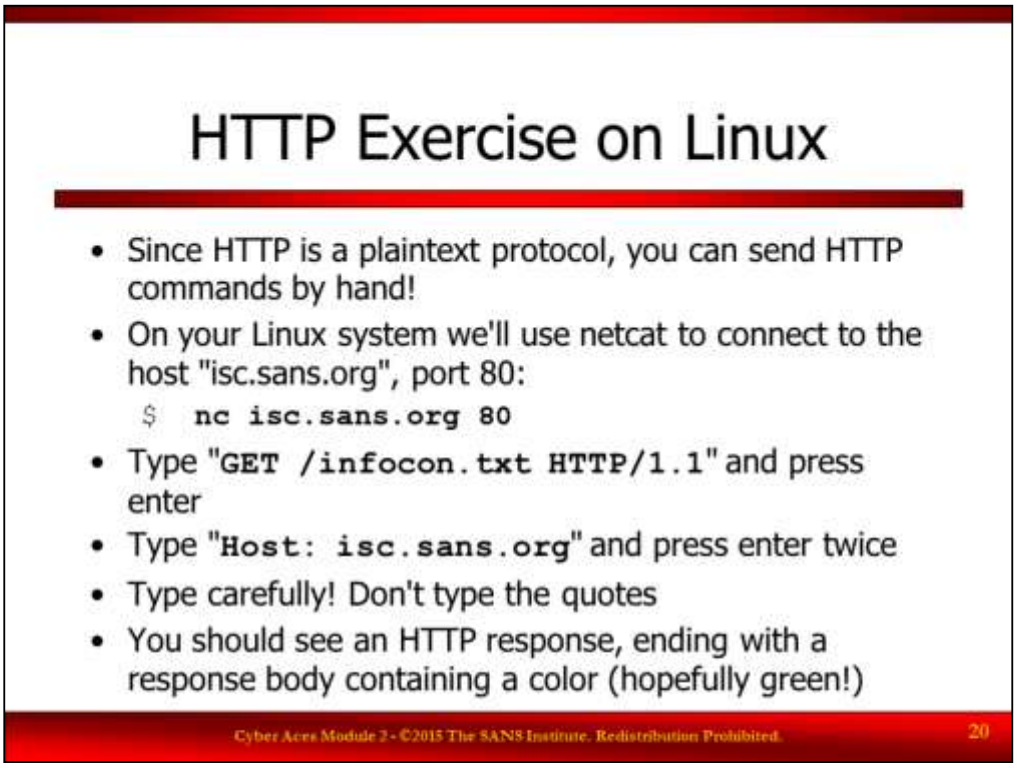
```
HTTP/1.1 200 OK
Date: Fri, 30 Sep 2011 15:32:11 GMT
Server: Apache
Content-Length: 3367
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
  Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
  strict.dtd">
[...]
```

Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 19

Example HTTP Response

Here is the HTTP response associated with the request on the previous page. Again, a few unimportant headers have been removed for brevity. The first line contains the protocol and version (HTTP/1.1), followed by a space, the numerical status code, another space, then the description of the status code. The "Date" header indicates the date that the server is fulfilling the request. The third line contains the server header, which in this case simply shows "Apache" (the most commonly used web server on the Internet). Note that while the "Server" header often contains full version information, it can be configured to reveal much less detail for security purposes, as is the case above. The "Content-Length" header indicates the size of the response in bytes, so the browser knows how much data to expect (this is particularly useful for file downloads, to allow for a status bar). The "Connection" header confirms the server's intention to keep the current TCP connection open for subsequent connections. The "Content-Type" header specifies the MIME type of the response, allowing the browser to decide how to handle it. There is then a blank line, indicating the headers are over, and the rest of the data received is the resource that was requested.



HTTP Exercise on Linux

- Since HTTP is a plaintext protocol, you can send HTTP commands by hand!
- On your Linux system we'll use netcat to connect to the host "isc.sans.org", port 80:

```
$  nc isc.sans.org 80
```

- Type "**`GET /infocon.txt HTTP/1.1`**" and press enter
- Type "**`Host: isc.sans.org`**" and press enter twice
- Type carefully! Don't type the quotes
- You should see an HTTP response, ending with a response body containing a color (hopefully green!)

20

HTTP Exercise on Linux

While your Web Browser typically issues these commands for you, you can type HTTP commands in the terminal on your Linux system. Try this:

Open a "Terminal" in Linux

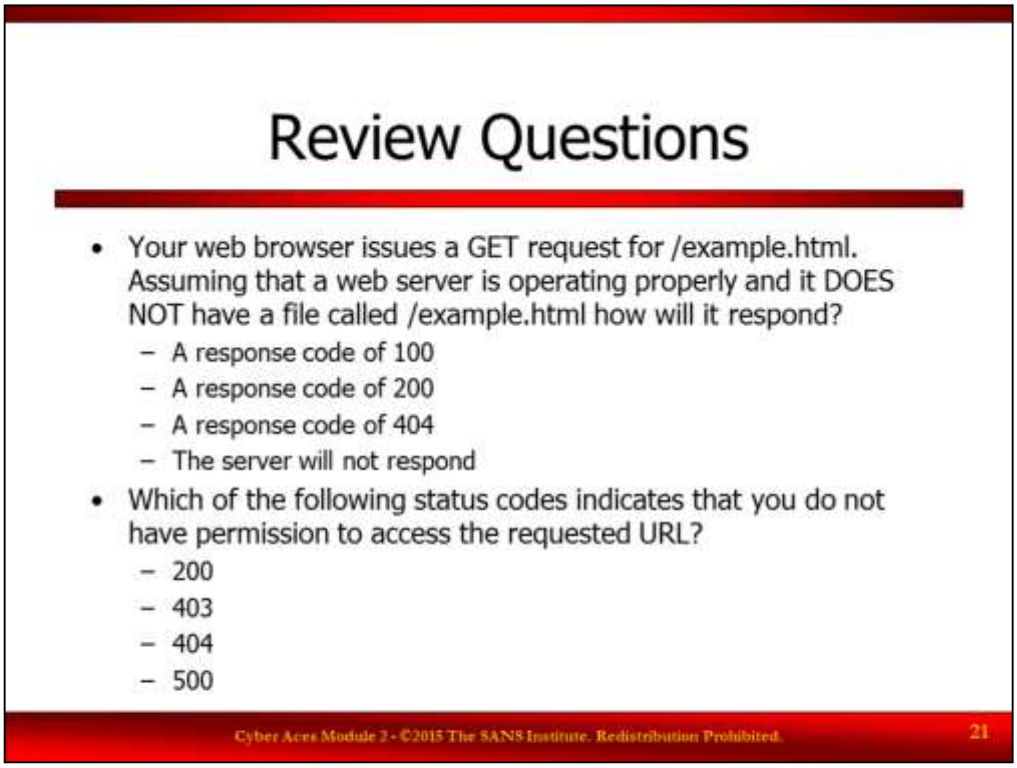
Type "**`nc isc.sans.org 80`**" and press enter

Type "**`GET /infocon.txt HTTP/1.1`**" and press ENTER (Note: Case is Important!)

Type "**`Host: isc.sans.org`**" and press enter

Press Enter Again.

Using netcat, your computer will download the document containing the current status of the internet from the Internet Storm Center. HOPEFULLY you got back an HTTP Response of 200 and several pieces of information from the server. At the bottom of that response is the requested document "infocon.txt". Hopefully the bottom of the document contained the word "green" indicating that all is well in on the Internet. This is the same thing your browser would display if you visited the address http://isc.sans.org/infocon.txt. For more information on the InfoCon, check out http://isc.sans.org with a normal browser.



Review Questions

- Your web browser issues a GET request for /example.html. Assuming that a web server is operating properly and it DOES NOT have a file called /example.html how will it respond?
  - A response code of 100
  - A response code of 200
  - A response code of 404
  - The server will not respond
- Which of the following status codes indicates that you do not have permission to access the requested URL?
  - 200
  - 403
  - 404
  - 500

Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 21

Review Questions

Your web browser issues a GET request for /example.html. Assuming that a web server is operating properly and it DOES NOT have a file called /example.html how will it respond?

•A response code of 100

•A response code of 200

•A response code of 404

•The server will not respond

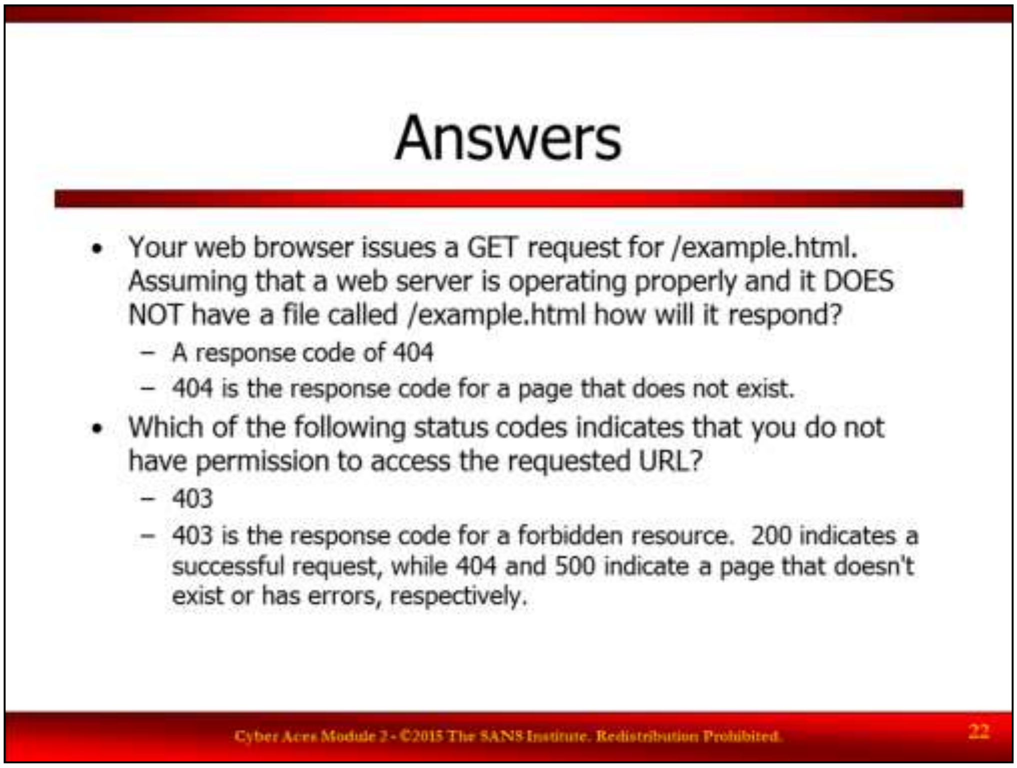
Which of the following status codes indicates that you do not have permission to access the requested URL?

•200

•403

•404

•500


Answers

- Your web browser issues a GET request for /example.html. Assuming that a web server is operating properly and it DOES NOT have a file called /example.html how will it respond?
  - A response code of 404
  - 404 is the response code for a page that does not exist.
- Which of the following status codes indicates that you do not have permission to access the requested URL?
  - 403
  - 403 is the response code for a forbidden resource. 200 indicates a successful request, while 404 and 500 indicate a page that doesn't exist or has errors, respectively.

22

Answers

Your web browser issues a GET request for /example.html. Assuming that a web server is operating properly and it DOES NOT have a file called /example.html how will it respond?

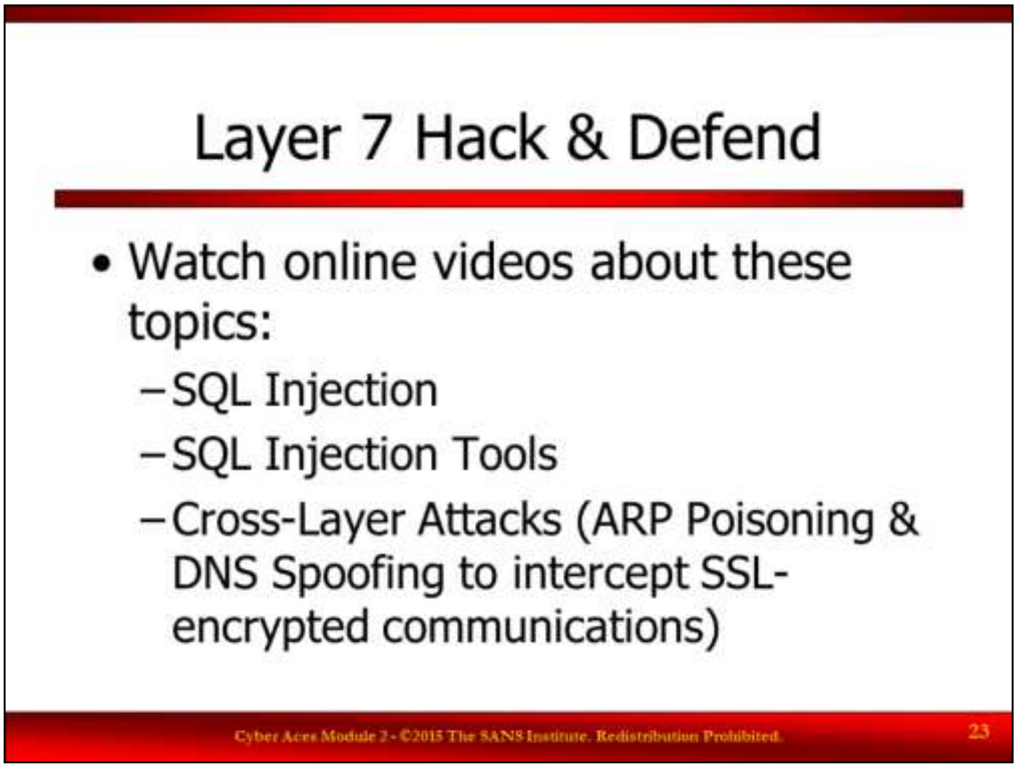
•A response code of 404

•404 is the response code for a page that does not exist.

Which of the following status codes indicates that you do not have permission to access the requested URL?

•403

•403 is the response code for a forbidden resource. 200 indicates a successful request, while 404 and 500 indicate a page that doesn't exist or has errors, respectively.



Layer 7 Hack & Defend

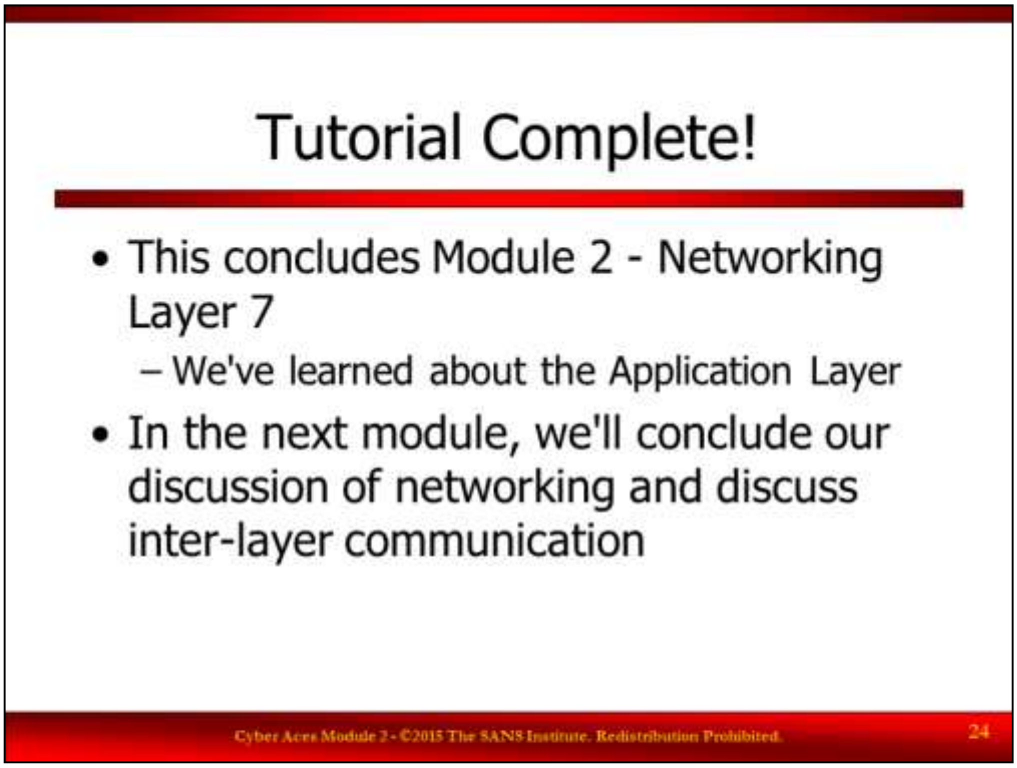
- Watch online videos about these topics:
  - SQL Injection
  - SQL Injection Tools
  - Cross-Layer Attacks (ARP Poisoning & DNS Spoofing to intercept SSL-encrypted communications)

Cyber Aces Module 2 - ©2015 The SANS Institute. Redistribution Prohibited. 23

Layer 7 Hack & Defend

Watch online videos about these topics:

- SQL Injection:
  - http://www.youtube.com/watch?v=h-9rHTLHJTY
  - http://www.youtube.com/watch?v=jMQ2wdOmMIA
- SQL Injection Tools:
  - Sqlmap - http://www.binarytides.com/sqlmap-hacking-tutorial/
  - Havif - http://hackyshacky.com/blog/havij-sql-injection-tutorial/
- Cross-Layer Attacks (ARP Poisoning & DNS Spoofing to intercept SSL-encrypted communications):
  - https://www.youtube.com/watch?v=GHphDapYp1w



Tutorial Complete!

- This concludes Module 2 - Networking Layer 7
  - We've learned about the Application Layer
- In the next module, we'll conclude our discussion of networking and discuss inter-layer communication

24

Tutorial Complete

This concludes the discussion about Layer 7, the Application Layer. In the next tutorial we'll conclude our discussion of networking and discuss inter-layer communication