Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Windows 2000 DNS Security
By Vivian Burns

GCNT practical assignment v2.1b

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002, Author retains full rights.

In Windows 2000, Microsoft made huge changes to its implementation of DNS. These changes provided markedly increased ease of use in some respects. In other ways, complications were also introduced. Factors that were previously nonexistent in infrastructure planning have now become vitally important, especially for organizations, like most, that run multiple operating systems. And Microsoft's implementation of DNS also has important implications for DNS security on the Internet as a whole. This paper looks at how these changes to DNS security work in the organization and on the Internet.

This paper will begin with an overview of the features of Microsoft DNS. Then it explains how security works in this implementation, including an explanation of how general DNS security concerns are addressed, and some how-to information. We will see how it conforms to RFCs, and what security concerns exist in a Microsoft-only DNS. Next, we will see how its security features compare with those in BIND DNS, as well as what interoperability concerns exist. Then, to bring it all together, we will look at how all of this information impacts the future of DNS security on the Internet as a whole.

Overview

DNS in Windows 2000 is markedly different from the version that came with Windows 4.0. The most fundamental change grabs your attention as soon as you upgrade a Windows server to a domain controller: DNS is required for a domain controller to work. The install asks for a DNS domain name, which it uses as the Windows domain name. If an authoritative name server for the domain cannot be located on the network, or if the server found does not support DNS dynamic update protocol, the install prompts you to install DNS server. The necessity for this has to do with a change that Microsoft made in their overall domain model: Windows no longer uses WINS for name resolution. Instead, Windows now uses DNS SRV records to locate domain controllers on the network.[1]

Windows 2000 does, however, provide means to handle older versions of windows that still require WINS for NetBIOS name resolution. It uses a record type called "WINS". This record provides the IP address of the WINS server or servers that can be consulted when a name is not found in the DNS. This record existed in NT 4.0, enabling DNS to perform WINS lookup on behalf of any DNS client. With this feature, Windows 2000 clients do not even need to be configured for NetBIOS name lookup. They use DNS as a locator service before trying WINS. This is a change from Windows 4.0, which performed a NetBIOS name lookup first and a DNS lookup second. [2]

This brings us to the second major change in the DNS – dynamic update. In the 2000 version of Microsoft DNS, dynamic update is turned on by default[3], although DNS can run without it. With dynamic update on, an administrator does not update the "A" and PTR records manually. Instead, Microsoft's Windows 2000 clients, when configured with a dynamic IP address, update their own "A" records, while DHCP updates their PTR records. For legacy clients, DHCP updates both the "A" and PTR records. Statically configured Windows 2000 clients update their own A and PTR records. Dynamic update, a proposed Internet standard, is outlined in RFC 2136. Microsoft claims compliance with 2136 for their implementation of this function. [4]

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Another proposed Internet standard that Microsoft followed is RFC 1995, incremental zone transfers.[5] Referred to as IXFR, the standard specifies a method for zone transfers that include only records that have changed since the last transfer. Implementations of DNS that do not follow this standard normally perform a transfer of the entire zone (AXFR), which can consume bandwidth unnecessarily. Where bandwidth or performance is a concern, such as over WAN links, the IXFR standard provides traffic relief.[6] Microsoft also claims compliance with a related standard, RFC 1996, called NOTIFY.[7] This standard specifies a protocol for prompt notification of changes. In other words, when the zone has changed, the primary notifies the secondary so that the secondary may initiate a zone transfer.[8]

However, these zone transfer protocols become less important in the context of the most significant option in Windows 2000 DNS: integration with Active Directory. Microsoft heavily encourages the use of this option. When you choose zone integration with Active Directory, DNS stores its records in AD, and DNS replication takes place as part of AD replication. Active Directory replication, since it is not specifically built for DNS, does not follow the IETF zone transfer standards cited above. AD replication also causes two more important changes to primary-secondary function. 1) Any DNS server can initiate and receive zone transfers, as secondaries usually do. 2) Record modifications can take place at any DNS server, as if they were all primaries. In this model of replication, therefore, the idea of a standard primary or secondary becomes irrelevant; all DNS servers are equals. Microsoft calls this arrangement multi-master replication. Active Directory uses multi-master replication for all services, so you can think of it as a side effect of the integration, rather than a model specifically designed for DNS. This multi-master replication provides fault tolerance for all data stored in Active Directory, including DNS data. Another result is some reduction in administrative overhead, as there is no need to set up DNS replication separately from Microsoft domain replication.[9]

An additional benefit of Active Directory integration is that each record possesses an Access Control List. As with all AD objects, administrators can have a fine-grained control over who can read, change, or delete any object or property of an object in the DNS. For more information about using Access Control Lists in Active Directory, read the chapter on Active Directory in the Microsoft's Windows 2000 server resource kit.[10]

Closely related to the Active Directory integration is Microsoft's secure dynamic update. The function of the secure update is to authenticate clients as they update their DNS records. Specifically, each client machine must be authenticated to the Active Directory before the server allows an update in this mode. More detailed information is provided in the following section.[11]

Another important change is support for the SRV record. Microsoft cites an Internet draft for their implementation of this record,[12] which has since become a proposed-standard RFC, number 2782.[13] Support of this record is particularly significant because Microsoft clients use this record to find authentication services.[14] The SRV record describes what services a server offers.[15] Some examples of SRV records and how they are used will be provided below.

Microsoft has provided other features, which have less relevance to security. This paper does not discuss these features detail. They include: 1) It supports UTF8 for host naming. UTF8 is an expanded character set[16] described in IETF draft standard RFC 2279.[17] Using it for host naming

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

is currently an Internet draft -- it is not yet an RFC. [18] This is an optional feature. [19] 2) Microsoft DNS now supports the extensions for Ipv6 (RFC 1886). [20] Ipv6 provides a larger address space by using longer addresses. 3) It supports round-robin load sharing (RFC 1794). [21] In this scheme, for names that map to multiple IP addresses, the server returns a different address for each request, rotating among those available. [21] 4) It supports negative caching, the retaining of non-existent host responses. [22]

Security features of Microsoft DNS

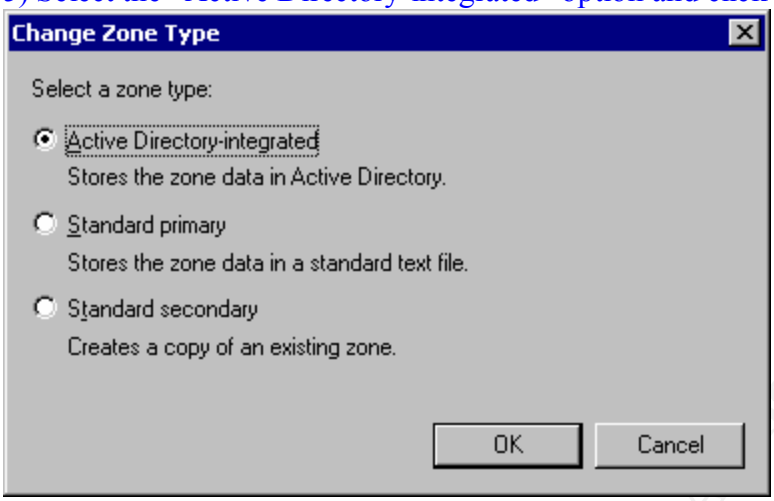
The Windows 2000 version of DNS is much more secure than the previous version. The first layer of security comes from Windows 2000 itself, which provides a more secure infrastructure than 4.0 did. You should begin your secure configuration of DNS by securing the operating system. If you need more information on this topic, you can consult the SANS reading room (www.sans.org). Second, DNS possesses security features of its own. This section focuses on what the security features are, how they work, and how one goes about activating and configuring them.

Active Directory Integration. Security for Windows 2000 DNS starts with Active Directory integration. In Windows 2000, when you create a zone you have three choices about what type of zone you are creating. You can create a standard primary, a standard secondary, or an Active Directory integrated zone. When you create a standard primary or secondary zone, the DNS server will follow the zone transfer RFCs that were mentioned in the section above. It will cause no interoperability problems with other DNS servers.

However, the most important security features function only when the DNS is integrated with Active Directory. Therefore, the first and most important action to take for security is to choose AD integration. And since Active Directory only runs on domain controllers, a secure DNS configuration must run on a domain controller. DNS can be installed on member servers, but since they do not run active directory, this configuration of DNS is less secure. Even the NSA recommends running Windows DNS with AD integration. [24] Some of the features that depend on AD integration are: Secure dynamic update, aging and scavenging, access control lists on individual records, and fault tolerance from multi-master replication. [23] Secure dynamic update, scavenging, where to edit access control lists, and how replication works will be discussed in the following pages.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

To integrate a zone with Active directory,
1) Open the DNS console. You can find it in the Administrative Tools menu.
2) Open the applicable DNS server, then the "Forward Lookup Zones" folder.
3) Click on the zone you want to integrate and select "Properties."
4) From the "General" tab, click the "Change" button.
5) Select the "Active Directory-integrated" option and click "OK".



This option is unavailable if the server is not a domain controller! It is grayed out.[25]

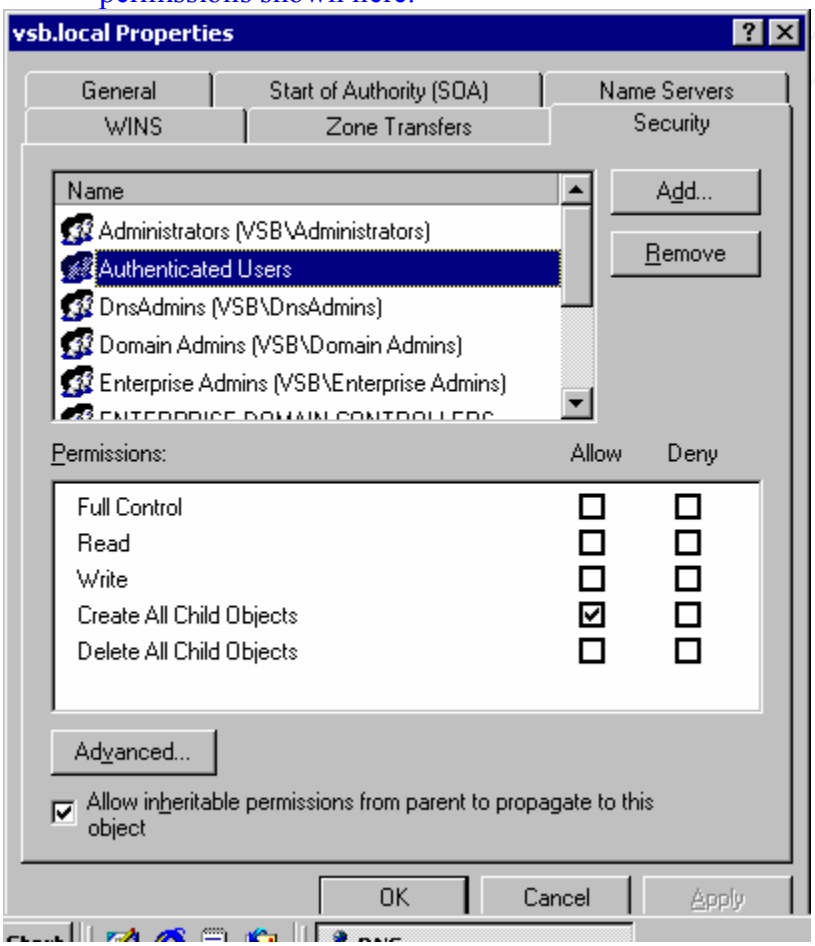
© SANS Institute 2000 - 2002 Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Active Directory itself provides a level of security not usually found in DNS implementations by virtue of its object-specific ACL's. This means that an administrator can control at any level of granularity the permissions to DNS objects and their properties.[26] In practice, since most records reach the DNS by way of dynamic update, their permissions will not be set individually. However, the permissions on the zone can be used for administrative control, as with other Active Directory objects.

To set zone ACLs,

1) As explained above, open properties for the applicable zone. If you want to set the ACL for a particular record, open the properties for the record.
2) On the "Security" tab, adjust the permissions as needed. Note that the "Advanced" button shows the more fine-tuned permissions that are used to build "read", "write", and so on, just as in Windows 4.0. However, the recommend method in almost all cases is to use the permissions shown here.

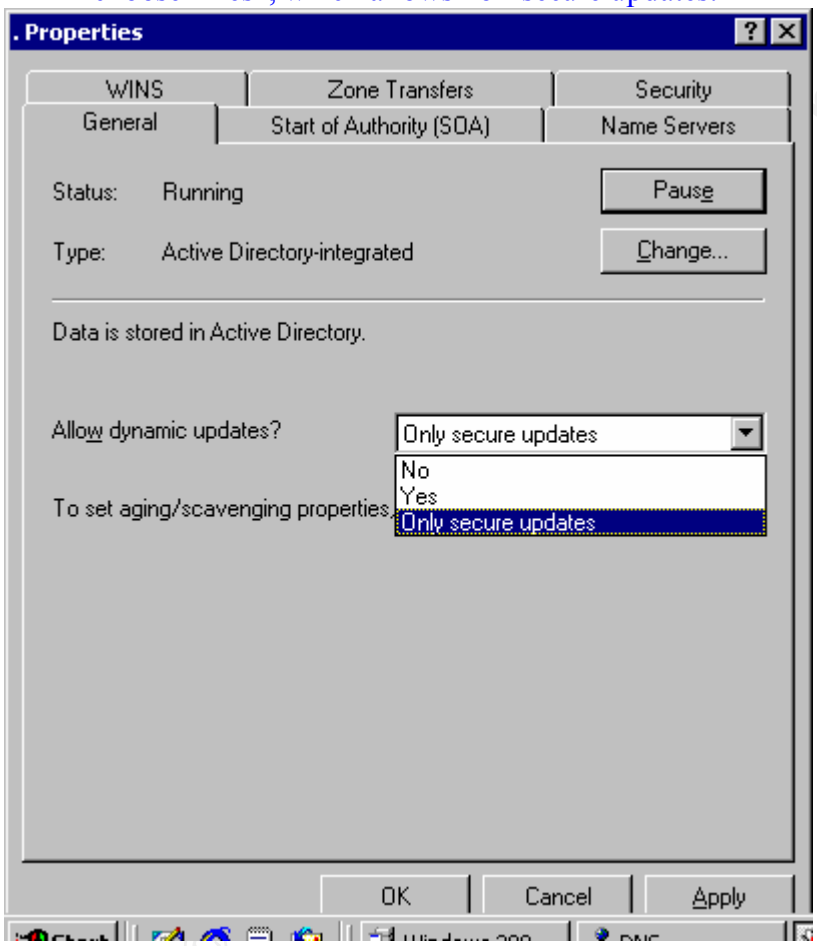


© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Secure Dynamic Update. Active Directory ACLs also participate in secure dynamic update. DNS uses AD ACLs to determine whether an update is authorized. [27] By default, any authenticated security principal may perform updates, as shown in screen shot above. Once a record has been created, only the creator of the record may update it. And by default for AD-integrated zones, dynamic updates are required to be secure. [28] That is to say, the server does not honor non-secure update requests. This is an appropriate setting for security purposes. If you have clients that cannot perform secure updates, you may need to configure DHCP to do their updates for them.

To secure dynamic updates,

1) As explained above, open properties for the applicable zone.
2) On the "General" tab, choose "Only secure updates" from the drop-down list. Do not choose "Yes", which allows non-secure updates.[29]

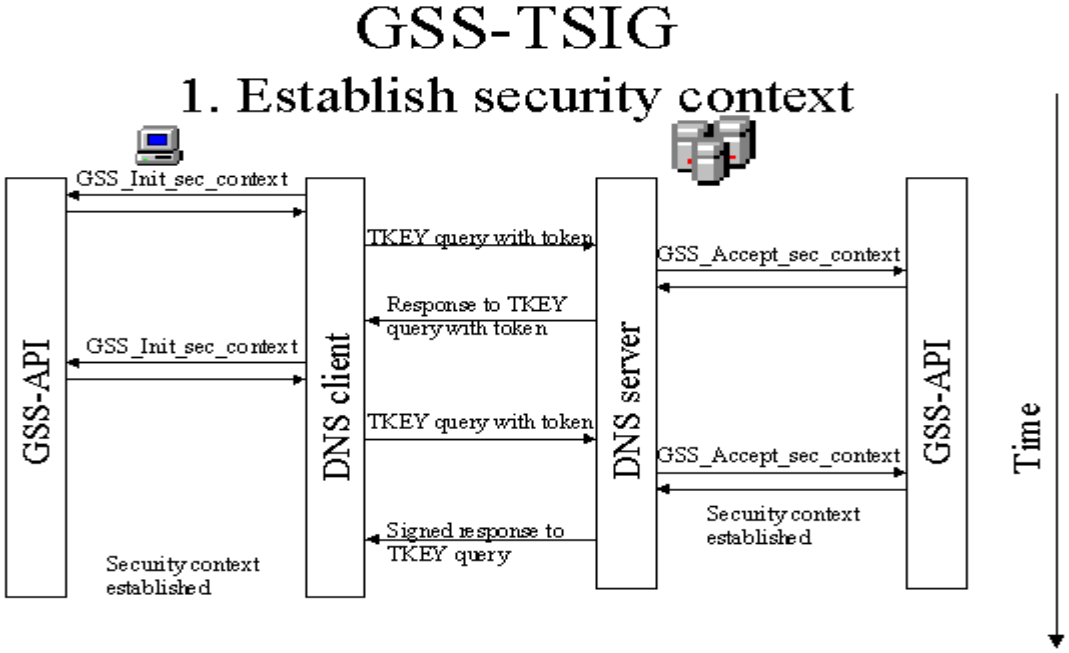


Microsoft does not use the IETF proposed standard for secure dynamic update, RFC 3007. That standard relies upon the DNS Security extensions outlined in RFC 2535, which Microsoft is not supporting either. Instead, Microsoft uses the Generic Security Service Application Program Interface (GSS-API) with TKEY, TSIG and Kerberos.[30] GSS-API is a set of programming calls, defined in RFC 2743, provided to the application developer in order to shield him or her from the details of a security mechanism. With this API, a developer does not have to know how the security mechanism works and does not have to create management structure to support them. The same calls may also be used for any security mechanism available on a system. [31] Microsoft

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

extended the GSS-API for use with TSIG. The extension algorithm, called GSS-TSIG, is described in the IETF draft draft-ietf-dnsext-gss-tsig-02, currently expiring September 2001. [32]

GSS-TSIG starts with the TKEY meta-resource record mentioned above. It is defined in RFC 2930 and is used to establish a shared secret between the resolver and the server. It is called a meta-resource record because it is used as a vehicle for DNS information, and is not kept in the DNS database. The specification requires that the record be kept in volatile storage (memory) only, until it expires. TKEY is valid only during a period specified by a particular start time and expiration time, so it is important for the two computers involved to synchronize their time. [33] In Windows 2000 systems, this is done automatically, depending on the PDC operations master at the root of a forest for authoritative time. [34] TKEY can operate in any one of five modes: server assignment, Diffie-Hellman, GSS-API, resolver assignment, and key deletion.[35]



For (Graphic from Stuart Kwan et al., "GSS-TSIG <draft-dnsext-gss-tsig-00.txt>" PowerPoint presentation given at July 2000 IETF proceedings)[36]

Windows secure DNS update, TKEY is used in GSS-API mode. To begin the process, the client machine uses the GSS_Init_sec_context call of the GSS-API, passing the name of the server and the name of the security mechanism, Kerberos. The called routine returns a context handle, which the client uses to keep track of the context being established. The routine also returns an output token and a result code, called a major_status. The output token is a string of information that the client must pass to the server. The client does not need to parse or run algorithms upon the information in the string. The major_status, on first call, returns the result GSS_S_CONTINUE_NEEDED. This means that the routine completed successfully but the security negotiation has not yet completed.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Next, the client sends a DNS query, type TKEY (249), to the server. For this type of query, the client puts the TKEY record in the additional records section of the DNS packet. Among other things, the TKEY record contains: a client-generated name for the security context being negotiated; the algorithm name, which informs the server that this negotiation is using GSS-TSIG; the TKEY mode field, which informs the server that this TKEY query uses mode three -- GSS-API mode; and in the data section, the output token returned by the GSS routine.

When the server receives the TKEY query, it sees that the client is using the GSS-TSIG algorithm and GSS_API mode. Based on this information, the server checks its mapping table. The mapping table keeps track of what context handles belong to what context names. The server checks to see if the name specified in the TKEY record exists in the mapping table. Since this is a new query, it does not exist. The server calls the GSS_Accept_sec_context routine from the GSS-API, passing in the input token. The routine returns a context handle, a GSS_CONTINUE_NEEDED major_status, and an output token. The server puts the context handle into its mapping table under the context name that the client specified.

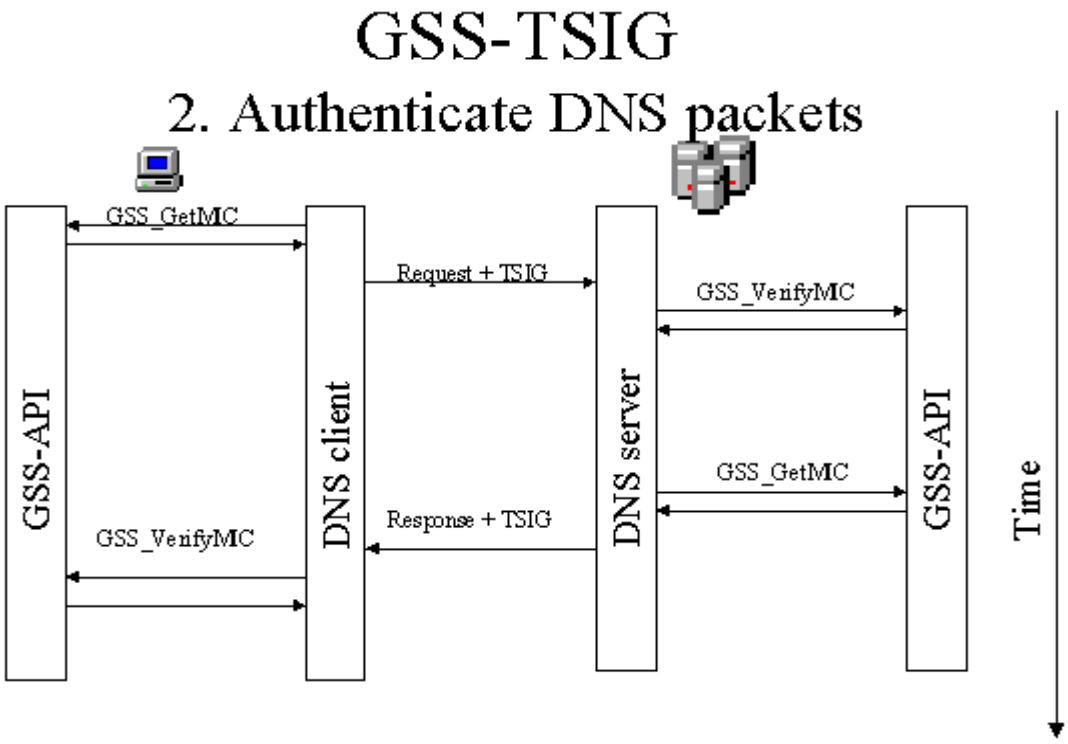
Next, the server sends a TKEY record to the client. This TKEY record, just like the TKEY from the client, is enclosed in a DNS packet. However, since this is a DNS response, the TKEY record is in the answer section. Again, the data section of the TKEY record consists of the output token return by the GSS routine.

When the client receives the server's token, it again calls GSS_Init_sec_context. Unlike the first call, when the client was initiating the process, this time it passes a token to the routine, as received from the server. If the routine returns a continue status, the whole cycle repeats as before, with the client sending its output token to the server, and the server processing that token before returning its own output token in a DNS response. The other possible outcome of the routine is that it returns GSS_S_COMPLETE, meaning that the client has successfully established a security context. The routine also returns an output token. The client must send it to the server in another TKEY request.

When the server receives the output token created during the client's completion, it finds the context name in its context table. It uses the context handle to call the acceptance routine, passing in the token supplied by the client. At that point the routine should end with a complete status, and a null token. This means that the context has successfully been established. The server does, however, need to respond to the client's last TKEY request.

The TKEY record for this last leg of the negotiation contains the same output token that the client passed to the server. The response also contains a TSIG (250) record in the additional records section. TSIG is another meta-record, which is not kept in the DNS database, but instead is kept in volatile storage until it is no longer useful. In the case of TSIG, it is kept during one DNS transaction. The purpose of a TSIG record is to provide a hash of the DNS message in which it is sent. The sender performs a hash on the entire message and certain TSIG fields before adding TSIG to the additional section, and the recipient removes TSIG before verifying the hash. The TSIG timer values are included in the hash to protect against replay attacks.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



(Graphic from Stuart Kwan et al., "GSS-TSIG <draft-dnsext-gss-tsig-00.txt>" PowerPoint presentation given at July 2000 IETF proceedings)[37]

To create the TSIG record, the server first calls GSS_GetMIC, passing in the entire outgoing message and appropriate TSIG fields. The routine returns the hash that the server includes in the TSIG record. The response is sent. When the client receives it, the client calls GSS_VerifyMIC, passing in the message minus TSIG and the TSIG hash. When the routine returns a complete major_status, the negotiation is complete. [38] At this point all communications between client and server can be secured by TSIG, as shown above.

This sequence of events, which is described in the GSS-TSIG draft, is not entirely compliant with the TSIG RFC. 2845 states that a server must not sign responses to unsigned requests.[39]

Now that we've seen the communication process for a secure update, we are still left with an important question: what is going on inside the black box of those GSS-API routines? We know already that the Windows 2000 client specifies the Kerberos security mechanism when it calls these routines. Let's briefly review how Kerberos works.

When any security principal wants to log in to a Microsoft domain, the first thing Windows must do is discover where the appropriate servers are. It does this by querying DNS for SRV records. When it boots, each server dynamically creates these records in the zone in active directory. SRV records advertise the existence of LDAP services, domain controller services (authentication), global catalog and Kerberos services. Active Directory is an LDAP compliant service, so it

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

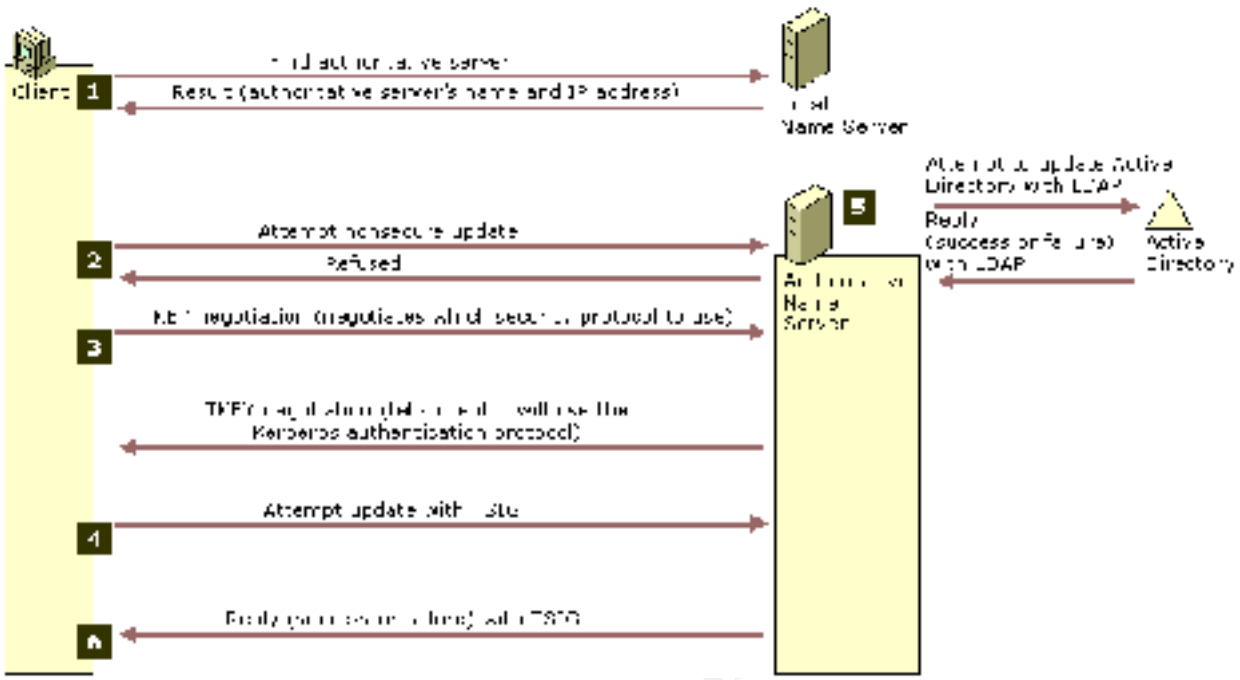
inserts a record having owner name of the form "_ldap._tcp.*DnsDomainName"*. Other owner names of records inserted by Microsoft servers are[40]:

Ldap server at a specific site:	_ldap._tcp.*SiteName*._sites.*DnsDomainName*
Domain controller:	_ldap._tcp.dc._msdcs.*DnsDomainName* and **_ldap._tcp.***DomainGuid***.domains._msdcs.***DnsForestName*
DC at a specific site:	_ldap._tcp.*SiteName*._sites. dc._msdcs *DnsDomainName*
Primary domain controller:	_ldap._tcp.pdc._msdcs.*DnsDomainName***.**
Global catalog server:	**_ldap._tcp.gc._msdcs.***DnsForestName* and **_gc._tcp.***DnsForestName*
GC server at a specific site:	**_ldap._tcp.***SiteName***._sites.gc._msdcs.***DnsForestName* and **_gc._tcp.***SiteName***._sites.***DnsForestName*
Kerberos server:	**_kerberos._tcp.***DnsDomainName* and **_kerberos._udp.***DnsDomainName*
Kerberos at a specific site:	**_kerberos._tcp.***SiteName***._sites.***DnsDomainName*
Kerberos w PK extensions:	**_kerberos._tcp.dc._msdcs.***DnsDomainName*
Above, at a specific site:	**_kerberos.tcp.***SiteName***._sites.dc._msdcs.***DnsDomainName*
Kerberos password change server: (DCs and KDCs)	**_kpasswd._tcp**.*DnsDomainName* and **_kpasswd._udp**.*DnsDomainName*

Note that a single domain controller might register several records, using various of these owner names, and that there will be multiple hosts corresponding to some of the owner names. For example, if a request contains the owner name "_ldap._tcp.dc._msdcs.example.com", the response may contain several records, each containing a different RDATA field.

Once a client machine has queried DNS for SRV records, it can use them to locate the Kerberos server nearest to itself. It then creates a secret key, based on the password, for communicating with the Kerberos server. It sends an encrypted request for a Ticket-Granting Ticket (TGT). On the other end, the Kerberos server looks up the secret key in the Active Directory, which stores the key with the other user ID information. It uses this shared secret to decrypt the message and sends back the TGT, encrypted with the shared secret. Then the client can use the key specified in the ticket to request a session ticket for the domain server.[40]

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



(Graphic from Microsoft Corporation, "Chapter 6 - Windows 2000 DNS", in the Windows 2000 Server TCP/IP Core Networking Guide of the Windows 2000 Server Resource Kit)[41]
When a machine wants to update its DNS record (or the DHCP server wants to update a record on behalf of a machine), the TGT probably is already established, based on the credentials for the machine itself. Note that the security principal is the machine and not the user. The first thing that Windows needs to do is locate a server that can accept the update. To locate the server, the client sends a request for a SOA record to DNS. The SOA record is supposed to list the primary DNS server for a zone. For AD-integrated zones, any domain controller running DNS can and will claim to be the primary. Next, the client sends a request to the server listed in the SOA to confirm that it is authoritative for the name to be updated. If so, the client sends an unencrypted update request. When that fails, the client uses TKEY negotiation to get a session ticket, which it can use for a TSIG secure update.[42]

That's more or less what's going on under the hood of GSS-API's communications with the server. As for the hash function performed by the GSS_GetMIC function, the TSIG RFC specifies that the HMAC-MD5 algorithm be used. [43] This algorithm performs a hash based on a shared secret. By the time this function is called, the shared secret has already been established. Note that this entire multi-layered security structure depends on the password of the security principal.

When all the steps have completed to enable a secure update, the client sends an update request to the server, according to RFC 2136. "Update" is a message format (opcode 5) in DNS, like Query is. It accordingly has its own rules about what it expects in DNS messages. The five parts of the message are: Header, zone, prerequisite, update, and additional data. The prerequisite section may list preconditions that must be met before the update can take place. These prerequisites cause checks for the existence of certain RR sets or names. The update section contains records to be added to or deleted from a zone. DNS uses values in the type and class

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

fields to identify the action to be taken upon the record. Once the update has been processed, DNS returns a response to the client.[44]

If the update succeeded, the process is complete. Otherwise, the client sends a request to DNS for NS records. It tries an update with the first server on the list, then the next, and so on until it meets with success. If all servers listed in NS records fail, the client repeats the request after 5 minutes. If the update still fails, the client tries again in another 10 minutes, then 50 minutes. [45]

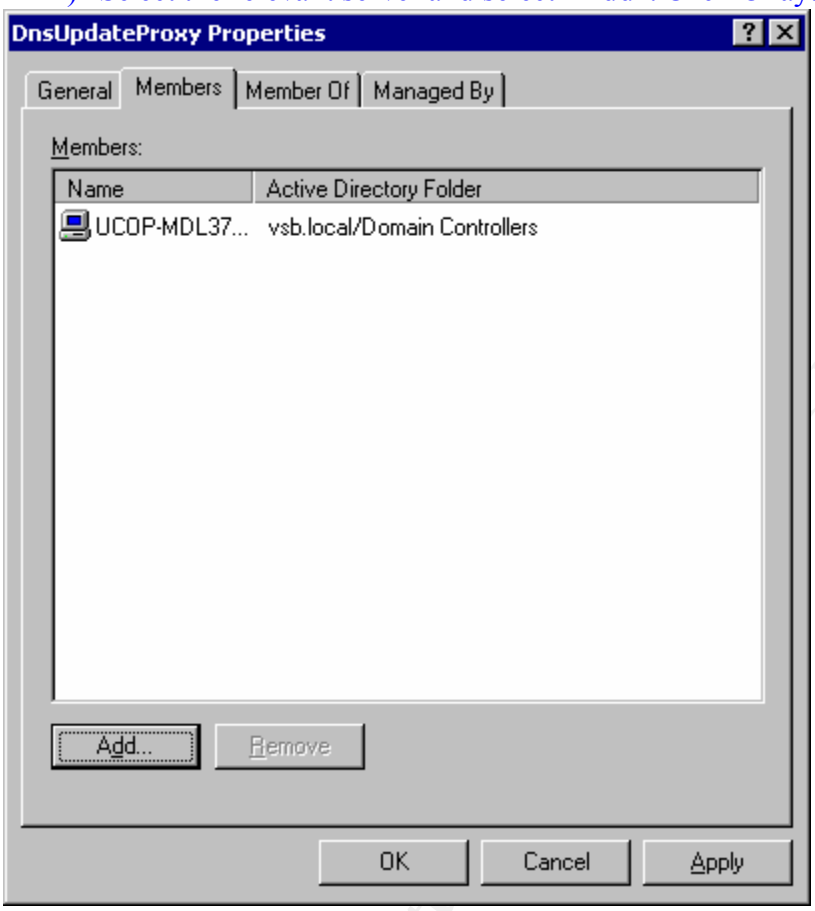
We're not quite done with secure dynamic update yet, though. There is one more wrinkle, which is extremely important for security. Remember that DHCP performs these updates for clients that are Windows 2000 unaware, and also updates PTR records for 2000 clients. This has important implications. Suppose that DHCP had updated a particular record, and later some other DHCP server needed to update it. The second DHCP server would be unable to perform this change because the record belongs to the first server. Microsoft has provided a convenient means to solve this problem. It's called the DNSUpdateProxy group. This built-in group has a special function. When you put a Windows server in the group, the records that it creates in DNS have no owner, and therefore can continue to be updated by other DHCP servers. Not until a host that does not belong to the group updates the record will it have an owner. This means that any authenticated host can take possession of the record. Even the A record that DHCP creates for itself is ownerless, so it's important to manually change the security on that record. If you have DHCP installed on a domain controller, there's an additional problem: all of the SRV and CNAME records that the server adds to DNS will be ownerless. To prevent this, you must not install DHCP on a domain controller if you are using the DNSUpdateProxy group. [46]

© SANS Institute 2000 - 2002 All rights reserved

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

To add a DHCP server to the DNSUpdateProxy Group,

1) In the "Active Directory Users and Computers" console, open the users folder.
2) Double-click the "DNSUpdateProxy" group.
3) Choose the "Members" tab and click the "Add" button.
4) Select the relevant server and select "Add". Click Okay. [47]



There is another way to go around the problem: eliminate DHCP from the update process. Configure all publicized nodes with static addresses. For many organizations, this is not a big change, since they use dynamic addresses mainly for user systems. When you configure Windows 2000 servers with static IP addresses, they update their own A and PTR records without making use of the DHCP server[48]. Unless peer-to-peer sharing is necessary on your network, you can leave the workstations out of DNS altogether – configure them not to update their DNS records. This scheme provides several benefits: 1) You escape the effort and fallibility of manual zone updates for your servers when they change 2) You minimize the information that DNS offers to potential attackers, as you remove all but the information needed for correct functioning of your systems. 3) By choosing not to support peer-to-peer networking, you may also keep network traffic lower. 4) You escape the security concerns associated with DHCP updates, which were described above.[49]

Multi-master replication may be looked upon as one of the more controversial aspects of Microsoft DNS. It does not comply with RFC 1996 (Notify), even though Microsoft claims compliance with this RFC. It states "The zone's servers must be organized into a dependency

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

graph such that there is a primary master, and all other servers must use AXFR or IXFR either from the primary master or from some slave which is also a master. No loops are permitted in the AXFR dependency graph." [50] In Microsoft multi-master replication, loops are permitted in the replication dependency graph. [51] This method of replication also fails to comply with best practices as outlined in BCP 16. That RFC discourages the use of more than one primary because, in most scenarios, such configurations lead to inconsistencies in the data. [52] However, in Microsoft's replication model, the data is "guaranteed" to converge on the same set of values. This does not mean the replicas are guaranteed to be consistent at any particular point in time. Only if no changes are made and the systems are left to themselves do they actually converge on a matching set of data. Microsoft calls this state of affairs "loose consistency." The tradeoff is greater availability, as the naming system in particular and Active Directory in general gain a fault tolerance not found in the traditional model.

So how does Microsoft deliver on this guarantee of loose consistency? For starters, AD updates do not depend on timestamps. Instead, each server assigns a local set of Update Sequence Numbers to object attributes as updates occur. Then, each domain controller tracks the highest sequence number that it has already received from each of its replication partners. When a server requests an update, it sends this High-Watermark so that the partner need only send updates of those objects possessing higher sequence numbers. Since all Update Sequence Numbers are stored with their associated objects, this eliminates the need for a transaction log.

Because each server assigns its own sequence numbers, comparisons of them cannot be used to resolve conflicts between near-simultaneous updates on multiple servers. The replication system requires other means for resolving update conflicts. What the Active Directory does is to use volatility as the primary criteria for decision, and time secondarily. In other words, when an attribute has been modified on two servers, the value for the instance that was modified a greater number of times is retained. If volatility is equal, time of update is considered. And in those instances where even the time is equal, Active Directory bases its decision on the Server DSA – a fairly arbitrary number.

Two more important aspects to understand about Active Directory replication are these: First, replication takes place as a pull process. In other words, a server that is the source of an update does not initiate the replication. Instead, each server is responsible for requesting updates from its own replication partners. This is comparable to DNS secondary operation, as secondaries always initiate zone transfers. Second, domain controllers do not receive updates directly from every other domain controller. Instead, changes are replicated in a hierarchy that roughly corresponds to the AD tree. Changes from a domain controller on one branch may be replicated up the tree before coming back down another branch to a controller there. This hierarchy is similar to standard DNS, except that updates are propagated both up and down the structure. [53]

Aging and scavenging. To assist with data integrity, Microsoft's DNS also provides a feature called aging and scavenging. The purpose of this function is to automatically delete dynamically added DNS resource records when they have not been renewed for a specified period of time. These records are called "stale" records, and they may occur when a computer disconnects without deleting its RRs. Manually added records may be included in this process also, but the

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

administrator must manually modify the time stamp to make that happen. Let's look at how aging and scavenging works.

When DHCP or a client requests creation of a new resource record, DNS assigns a timestamp, based on the local server time. After that, a Windows 2000 client sends a refresh request every 24 hours while it is running, or whenever it reboots. The refresh request updates the timestamp with the current time.

At the moment of resource record creation, the timer on the no-refresh interval starts. During the no-refresh interval, DNS will only update a time stamp if the record is modified. For example: if a record is created at 3:00 pm on Monday, and the no-refresh interval is set for seven days (the default), a client cannot refresh the record's timestamp until the following Monday at 3 pm. The timestamp may only be updated if the record is actually changed. If the record is modified, it receives a new timestamp, and the no-refresh interval starts again at zero. In the example, if the record is changed on Tuesday, the no-refresh interval is reset and the record cannot be refreshed until the following Tuesday.

Once the no-refresh interval has passed, the timer on the refresh interval starts. During this time, a record can be refreshed. If the record is refreshed and therefore receives a new timestamp, this restarts the no-refresh interval for that record. If the record is not refreshed, it continues to be eligible for refresh for the duration of the refresh interval. The default is seven days. If the entire refresh period passes without any refresh taking place, the record becomes eligible for scavenging. That is to say, it is a stale record.

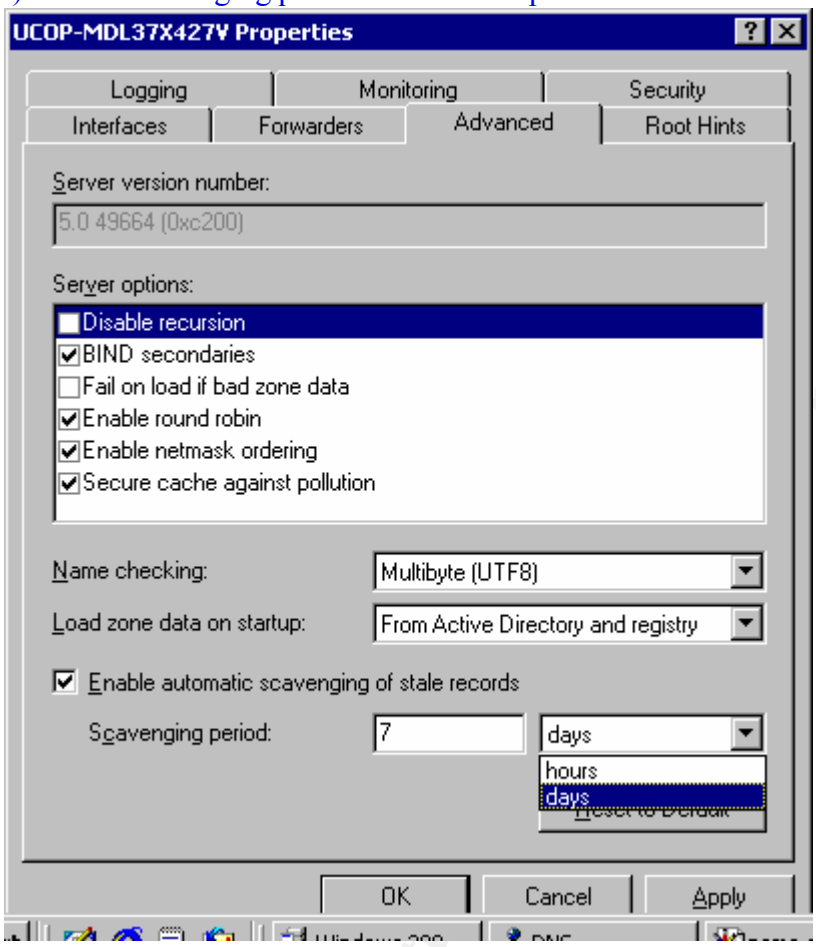
In the example, if a record is created at 3:00 pm on Monday, and no updates take place for a week, the following Monday the no-refresh interval passes and the refresh interval begins. If no refresh takes place during the next week, that record becomes a stale record, eligible for scavenging. If at any time during the two weeks the record is updated, the record receives a new timestamp and the no-refresh begins anew. Or, if the record is updated during the second week, it also escapes staleness. This means that a record must be as old as the length of the no-refresh interval plus the length of the refresh interval, and untouched for the length of the refresh interval, to become stale. If a client created a record, sent refreshes every day for six days, and then quit, the record would receive no refreshes and would become stale.

When the scavenging actually happens depends on the scavenging period. This period specifies how often the server checks for stale records. The default is seven days. Since the server is not constantly scavenging, but instead performs this function on a periodic basis, a record is not necessarily immediately deleted when it becomes stale. It is deleted the next time scavenging takes place. In the example, the server might be scavenging every Thursday. In that case, the record would exist in DNS for the three additional days between Monday when it became stale and Thursday when the server actually performed scavenging. The record can still be refreshed during this period and escape scavenging.[54]

Aging and scavenging is disabled by default. To take advantage of its benefits, an administrator must enable it. The administrator also needs to set the refresh interval, the no-refresh interval, and the scavenging period.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

To enable scavenging and set the scavenging period,
1) In the DNS console, right-click the relevant server and choose “Properties.“
2) On the Advanced tab, check “Enable automatic scavenging of stale records”.
3) Set the scavenging period from the drop-down list and its corresponding text box.[55]

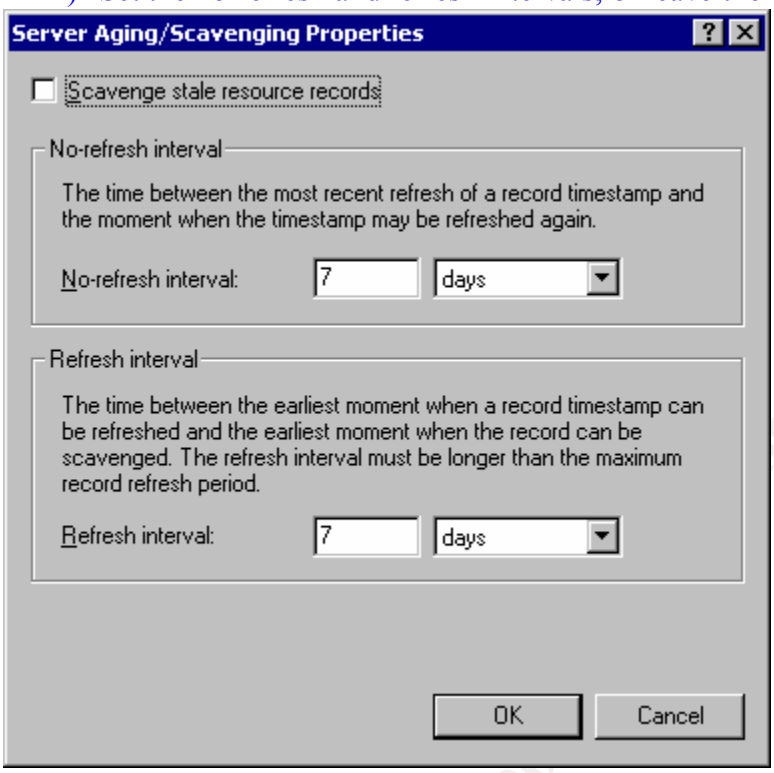


© SANS Institute retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

To set scavenging intervals,

1) In the DNS console, right-click the relevant zone and select "Properties."
2) On the General tab, click the "Aging" button.
3) Check "Scavenge stale resource records".
4) Set the no-refresh and refresh intervals, or leave them at default.[56]

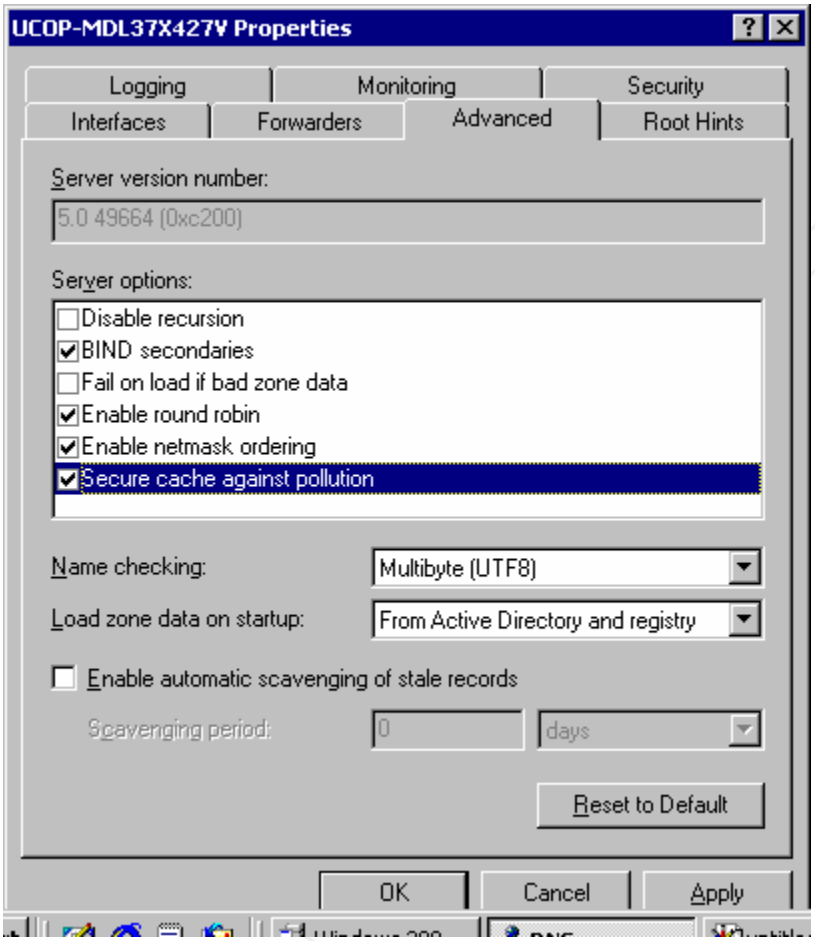


One more note on aging scavenging: When you enable it on a zone that already contains records, those records are not aged automatically. DNS only ages records that are added after you enable the option. However, you can use the command line tool Dnscmd to force scavenging of the old records. To do so, type Dnscmd <ServerName> /AgeAllRecords <ZoneName>.[57]

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Additional security features: Additional security features Microsoft DNS provides include:

- An option to "secure cache against pollution." When this option is selected, DNS discards additional responses – records provided that were not asked for in the request. For example, if the server requested Microsoft.com and got back a response that included Netscape.com in the additional section, it would discard the Netscape record. To secure the cache, open the relevant server in the DNS console. Then click "Secure cache against pollution" on the Advanced Tab.[58]

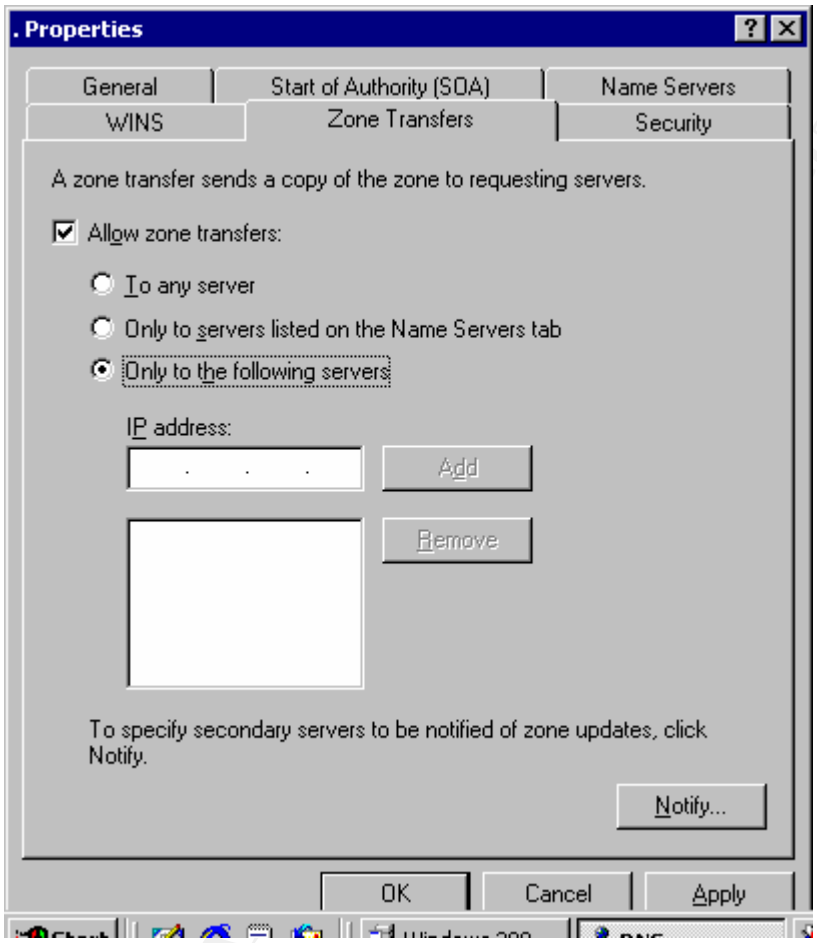


- Secure zone transfers, for Active Directory integrated zones only. As previously discussed, Active Directory-integrated zones do not use standard transfer mechanisms, but replicate with the rest of Active Directory on the basis of changed properties. Active Directory replication is automatically secured by Kerberos. [59]

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

- Restriction of zone transfers. That is, you can specify which servers are allowed to receive zone transfers. It is highly advisable to make use of this feature, as you usually do not want to allow potential attackers to have all of your zone information at once.

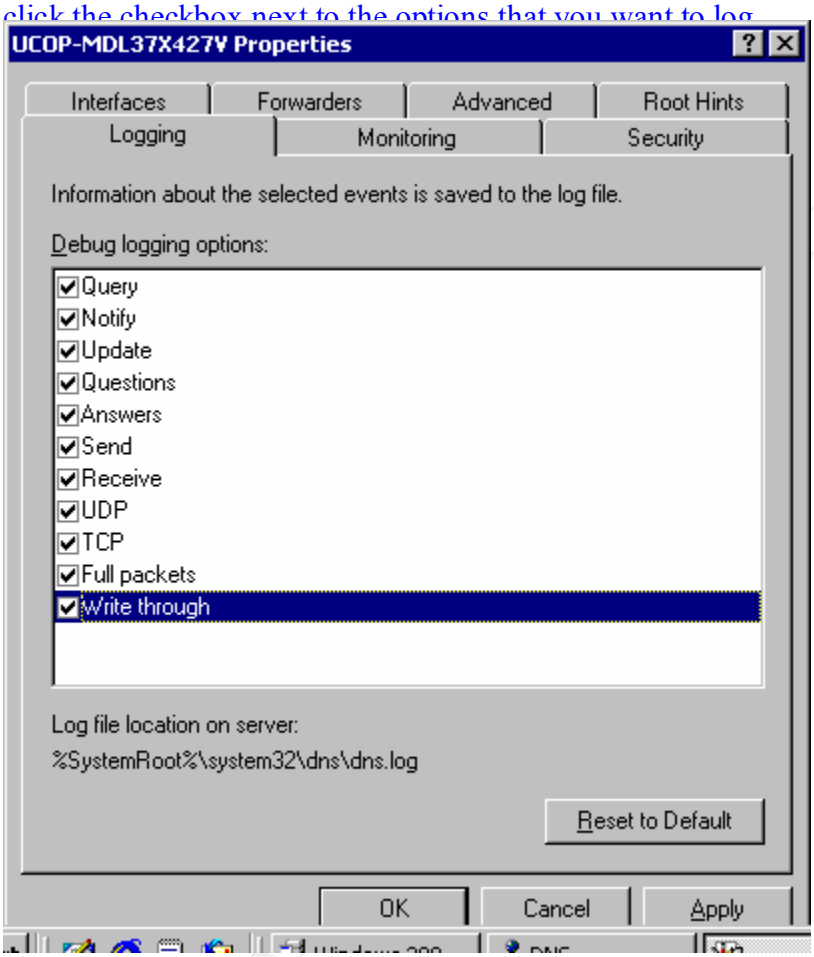
To restrict transfers, open the relevant zone and click the Zone Transfers. You can select "Only to servers listed on the Name Servers tab" or you can select "Only to the following servers" and specify which IP addresses are permitted. Other Windows 2000 DNS servers in the same zone will already be listed on the Name Servers tab, so most of the time this is sufficient. If you have BIND secondaries, you will need to list them on the Zone Transfers tab.



© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

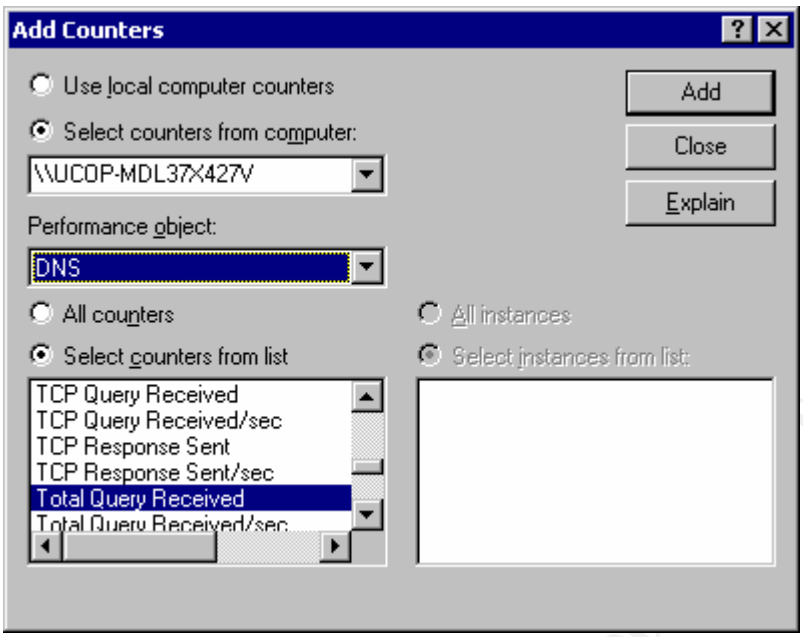
- A logging feature that enables you to log queries, notifications from other DNS servers, updates received, contents of question sections, contents of answer sections, number of queries sent and / or received, number of UDP requests received, number of TCP requests received, number of packets sent, and number of packets written back to the zone. As you can imagine, turning on these logging options increases the load on the server, so it's not advisable to use them all the time. Nevertheless, this feature can be a useful tool to find out what's going on. The logging information is sent to *windir*/system32/DNS/DNS.log. [60]

To turn on logging, open the relevant server in the DNS Console. Select the logging tab. Then click the checkbox next to the options that you want to log.



© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

- Another option for investigating some of the same information is to use the performance monitor. Microsoft has provided a whopping 62 performance counters for DNS. They include counters related to zone transfers, memory use, dynamic updates, notifications, queries, TCP and UDP packets, and WINS. [61]



- You can prevent the server resolver from accepting responses from servers it did not query. You need to change the registry, because the default is to accept these responses. To change the default, add the following registry entry: [62]

Path: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DnsCache\Parameters
Key: QueryIpMatching
Value: 1
Type: REG_DWORD

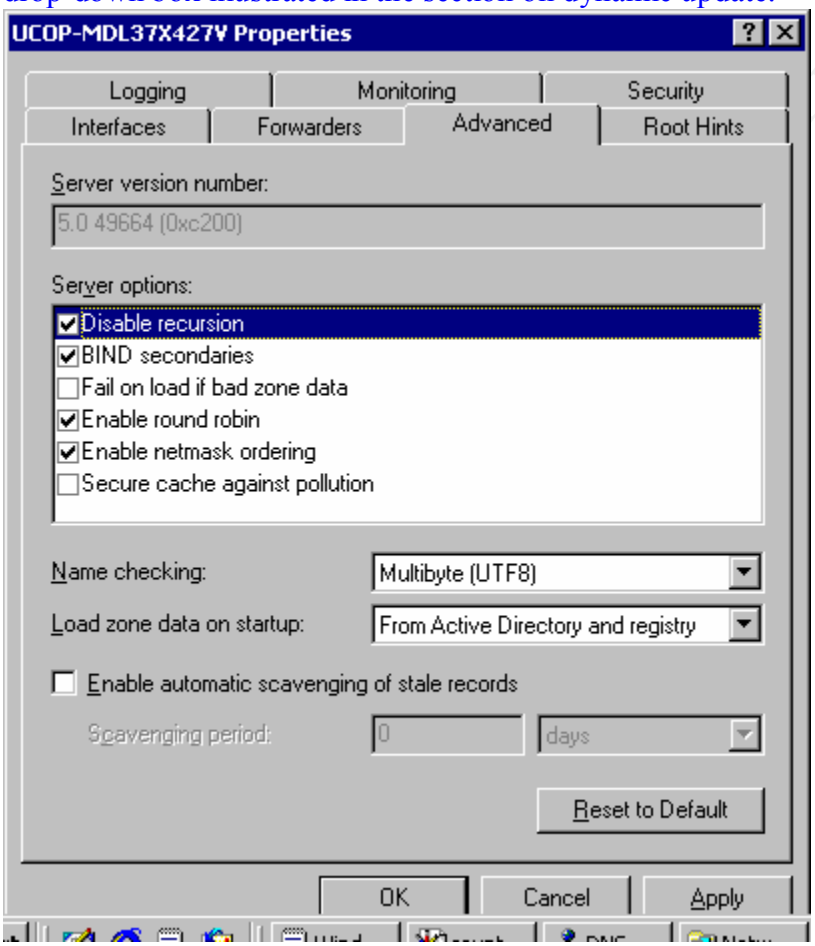
Best practices for configuring Microsoft DNS

Now that we've seen what security features Microsoft has built in to the DNS, let's look at how you can configure it further for security. We've already got a good start on following the general recommendations of Mr. DNS himself, Cricket Liu. His first recommendation is that you should install the most recent version of your DNS software. Of course, by installing Windows 2000 you are doing that. But be sure to apply your service packs, too. DNS is only as secure as the platform it runs on. Next, he recommends that you restrict zone transfers. We saw that Microsoft provides this option in the DNS console. He recommends that zone transfers be authenticated with TSIG. This particular recommendation is followed in spirit, but not practice, in pure Microsoft zones. Because zone transfers take place via AD replication, they do not use TSIG, but they are nevertheless secured. Microsoft does not provide secure zone transfers for non-integrated zones, so the best you can do to follow this recommendation is to use integrated zones. Next, he recommends that dynamic updates be restricted. Microsoft's security on updates effects this restriction.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

1) The next recommendation is slightly more involved. He suggests that you protect against spoofing by rejecting recursive queries from the Internet or rejecting all queries from the Internet, and rejecting additional data. Note that Internet queries must be allowed at least for an Internet-authoritative domain. This might imply domain splitting, where you have an internal DNS server that serves only internal requests, and an external server that serves external requests and reveals more limited information. If you split your domain like this, the external server should be non-recursive and the internal server should not answer any queries from the Internet. The NSA also adds that you should turn off dynamic updates on the external server. [63]

To configure for zone splitting, configure your external server as non-recursive. You can find this setting on the Advanced tab of the server properties. Also turn off dynamic update, using the drop-down box illustrated in the section on dynamic update.



Use your firewall to prevent inbound traffic to the internal DNS server. Configure the external server as a forwarder for the internal one. You find this option in the forwarders tab.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

RFC 2182, BCP 16, describes the IETF recommendations for best practices for DNS configuration. They include the following:

2) You should have at least two name servers available to the Internet from your zone, and they should be on separate subnets, preferably in separate physical locations. An effective approach is to identity a trusted organization of similar size, and swap secondary zones with them. In this scenario, each organization provides a secondary server for the other organization's zones. The purpose of setting up this way is to ensure availability in case the primary server is unreachable.

Some have argued that this type of redundancy is unnecessary. They say that if the network that a primary server is on is unreachable, name service is unnecessary because the machines the name server points to will probably be unreachable. There are some flaws in this argument. The first is that the name server may be unavailable even though the network itself is available, such as when the server's hardware fails or the DNS server is under attack. The second flaw concerns Internet performance in the situation where the network is unreachable. Some resolvers do not cache the fact of a failure to reach a domain server. In that case, there will be unnecessary traffic trying to reach the unavailable name server. Also, there are likely to be at least some hosts referenced by the name server that may not be on the unreachable network.

3) The IETF recommends that organizations have at least three servers, with at least one being away from the others, as described above. Note that this is not a requirement, but a recommendation. The DNS specification requires at least two. But if you follow recommendation number two, and also split your zone, you need at least three name servers, with the third one being internal.

4) Server addresses listed in DNS should be reachable from any client requesting the address records. This means that you should restrict what records appear on a name server used by hosts on the Internet. Usually, It's good practice to only include machines that are meant to be public, like your web server and your mail server. When following this recommendation, don't forget to pay attention to the NS records for the zone. Just as with other records, you should not list NS records pointing to servers that the external host cannot reach.

5) You can set up your servers so that even though all of them are authoritative, only one or two are listed. The unlisted servers are used for all local zones, and are called stealth servers. [64]

6) The last two recommendations are from Microsoft. For large organizations, you may not want your DNS administrators to control all of your zones. The purpose of this recommendation is to explain how you can securely assign separate groups for particular zones or sets of zones. First, create as many groups as you need for the different sets of permissions that you want to assign. Put all of the groups into the DNS Administrators group. Give the DNS Admins group read permissions for the entire domain. Then, in each zone, give full permissions to the particular group that is assigned to control that zone. [65]

7) You may need to reserve certain names for the user of only certain groups of users. To do so, create the name in the DNS console. Then modify its ACL so that only a particular security principal can change it. [66]

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Comparison with security features in BIND

Now that we've seen the security features provided in Microsoft's latest implementation of DNS, efforts to put it into practice in the real world immediately raise the question, how does its security compare with that of other implementations of DNS? And how well does it interoperate with them? How do you integrate Microsoft DNS into existing DNS structures? To answer these questions, we'll take a look at the latest version of BIND, the reference implementation of DNS. BIND is also the most popularly used implementation.

It was mentioned earlier that Microsoft does not support the DNS Security extensions outlined in RFC 2535 or the secure update outlined in 3007. BIND 9 does support DNSSec extensions.[67] This set of extensions provides a protocol for data origin authentication and transaction authentication. In other words, a DNS requestor can verify the source of a response, and can verify that the data has not been tampered with since it was sent from the source. DNSSec authentication is based on public-private key signatures.

To produce a digital signature, the procedure is this: First, the signer creates a hash of the data to be signed, using a one-way mathematical algorithm that produces a random-looking string of characters. Then the signer encrypts the hash value, using the private key. The public-private key algorithms ensure that once data has been encrypted using a private key, the only way to decrypt is to use the corresponding key. Therefore, successful decryption using a particular public key proves that the data was encrypted with the corresponding private key. To validate a digital signature, therefore, the following steps take place: First, one generates a hash of the data to be validated. Then the digital signature is decrypted using the public key. If the generated hash matches the result of the decryption, the signature is validated. You have proved that the data has not been altered since it was signed and that a holder of the private key signed it.

For DNS, here's how it works: Each zone owns at least one private-public key pair that it uses for signing. The public key is stored in a DNS resource record, type KEY (25). Storing it in DNS enables everyone to obtain it by means of a DNS request. The RDATA part of the record includes a flags field that designates what algorithm uses the key. For example, the DNSSec specification requires the DSA algorithm. Other algorithms that may be designated include RSA/MD5, Diffie-Hellman and elliptic curve crypto.

The server uses its private key to create a digital signature for each RR set in a zone. The signature is stored in a DNS resource record, type SIG (24). A SIG record must exist for each RR set. When a client requests an RR set, the server includes the SIG record in its response. Then the client uses the public key obtained from the KEY record to validate the records, using the method described above.

To secure the message in transit, the server can also sign the DNS message itself. To do so, it uses its private key to create a digital signature, based on the entire message except the parts that are changed by the creation of the signature. Then it includes the signature as a SIG record at the end of the additional records section. This is called a SIG(0).[68]

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

As is usual with public-private key schemes, this brings up some key-related questions. How is the private key kept secure? How is the public key certified to belong to the asserted owner? The RFC recommends that the private key be kept offline in order to keep it secure. It suggests that zone signing take place on a stand-alone server so that there is no chance of network intrusion. Once the zone is signed, it can be manually transferred onto the server that does the work of name service. For SIG(0), a server should use a private key owned by the host instead of the zone key.

The question of public key certification is slightly more complicated. To prove that a public key has not been tampered with, it must be signed with a SIG record like any other RR. The difference is that the key owner does not sign the KEY record; the parent zone does. To validate the signature on a key, the resolver must get the public key from the parent zone. This raises the question of how we know that the parent zone's key is legitimate. Again, the parent key possesses a SIG record signed by the next zone up in the hierarchy, and so on up to the root. To be sure of the security of a DNS record, a resolver must check all signatures until it comes to one that was generated by a trusted signer. The resolver can only trust a signer if the local configuration has specified the key as trusted. This requires a pre-existing copy of the public key on the local system. The process of obtaining a trusted public key has to have taken place separately, and how it happens is not specified by the RFC.[69]

To evaluate how this security mechanism compares with what Microsoft has provided, it might be helpful to take a look at what has been said by the developers of DNSSec. In June 2001, Ed Lewis in RFC 3130 summarized the working group meeting held at the latest IETF conference. The RFC stated that "The current collective wisdom is that DNSSEC is 1) important, 2) a buzz-word, 3) hard, 4) immature." The group includes representatives from NLnet Labs, Verisign, the Foundation for Internet Infrastructure, the Root Server System Advisory Committee, the Collaborative Advanced Interagency Research Network, NIST, the US Defense Information Systems Agency, the RIPE NCC, Network Associates, the IP.6 domain, and the Topology Based Domain Search. All of these organizations are performing work toward the goal of moving the DNS Security toward Draft Standard status. The consensus in this group is that one more iteration of the standard in proposed status is still needed. Also, interoperability between two implementations, required for draft status, cannot yet take place because there is only one implementation yet in existence – BIND. [70]

There is one more problem dogging the implementation of DNSSec. That concerns signing secure zones. Since each zone has to be verified by its parent, if there is any non-secure zone between a secured zone and the root, the secured zone looks like an insecure zone to the rest of the Internet. Local resolvers may be configured to trust the secured zone's key, but obviously no zone can configure all the resolvers that may ever query them. So far, the only way to avoid this problem is to have DNSSec deployed on the entire Internet, which doesn't look like it's happening anytime soon. [71]

As for secure dynamic updates, BIND uses "Simple Secure Updates." This protocol is described in RFC 3007, which came out in November of 2000. In Simple Secure Update, an update message must be signed by a TSIG or SIG(0) that can be validated by the server. This standard requires that the private key for a zone resides online so that SIG records can be generated on the fly when updates take place. [72]

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

BIND 9, in addition to DNSSec, and Simple Secure Update, supports SRV records, fast zone transfers, negative caching, and incremental zone transfers. Zone transfers can also be secure, using TSIG.[73]

Microsoft DNS with AD	BIND 9	Microsoft DNS w/out AD
Secure dynamic update using GSS-TSIG	"Simple secure update" as described in RFC 3007.	Not available
Aging and scavenging	No equivalent security	Not available
No equivalent security	DNS responses may be validated using DNSSec; the standard is immature and difficult to use	No equivalent security
Access control lists on individual records	No equivalent security	No equivalent security
Multi-master fault tolerance	DNS configured according to IETF best practice provides some fault tolerance	DNS configured according to IETF best practice provides some fault tolerance
Secure zone transfers using Kerberos	Uses SIG(0) or TSIG for secure zone transfers	No equivalent security
Restriction of zone transfers	Available option.	Available option.

Putting all this information together for comparison, it looks like you get reasonably good security with either Microsoft DNS or BIND, as long as you integrate Microsoft DNS with Active Directory. However, the BIND security mechanisms require more work to set up correctly.

If for any reason you cannot use Active Directory integration for Microsoft DNS, you are better off using BIND. BIND's secure updates, secure zone transfers, authenticated responses and restricted zone transfers make it unquestionably more secure than unintegrated Microsoft DNS.

Interoperability

Even though most organizations won't want to set up Microsoft DNS without AD integration, it's also fair to say that many organizations need to make it work with other, less secure DNS software. This need brings up the next topic: interoperability. Many organizations need naming services for systems running operating systems other than Windows, and many organizations already have non-Microsoft DNS servers running. Let's take a look at what the issues are and recommendations for dealing with them.

To understand interoperability, we need first of all to understand which features of Microsoft DNS are proprietary or depart from standards common to other DNS software. It was already mentioned that Microsoft's multi-master replication does not meet standards for best practices. But this replication method raises concerns about more than replication loops or data integrity. It eliminates the possibility that an Active Directory zone could play secondary to a non-Microsoft primary. The reason is that Microsoft gives you an either-or situation. Your zone can be configured as one of AD integrated, standard or secondary, but not both integrated and standard

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

or secondary. Therefore, as soon as you specify a zone as secondary, you lose integration and all of its benefits.

Next, there are concerns about Kerberos authentication. Microsoft has followed IETF standards in building their Kerberos implementation, but their authentication method requires extensions that do not follow the practices of other implementations. In "Kerberos interoperability Issues", Paul B. Hill of MIT states, "The Windows 2000 authorization data is ignored by current UNIX implementations. Although Microsoft has released information about their use of the authorization field it appears that the Kerberos community is precluded from writing any code that can use this information in any way." This might cause problems when non-Microsoft systems need to dynamically update their DNS records on a secure server.[74]

Another proprietary aspect is the WINS implementation. Some DNS servers may have problems with the WINS records during zone transfers. In order to avoid these problems, Microsoft has provided an option to use WINS records locally only. To set this option, open the zone properties and click the WINS tab. Select the "Do not replicate this record" check box. [75]

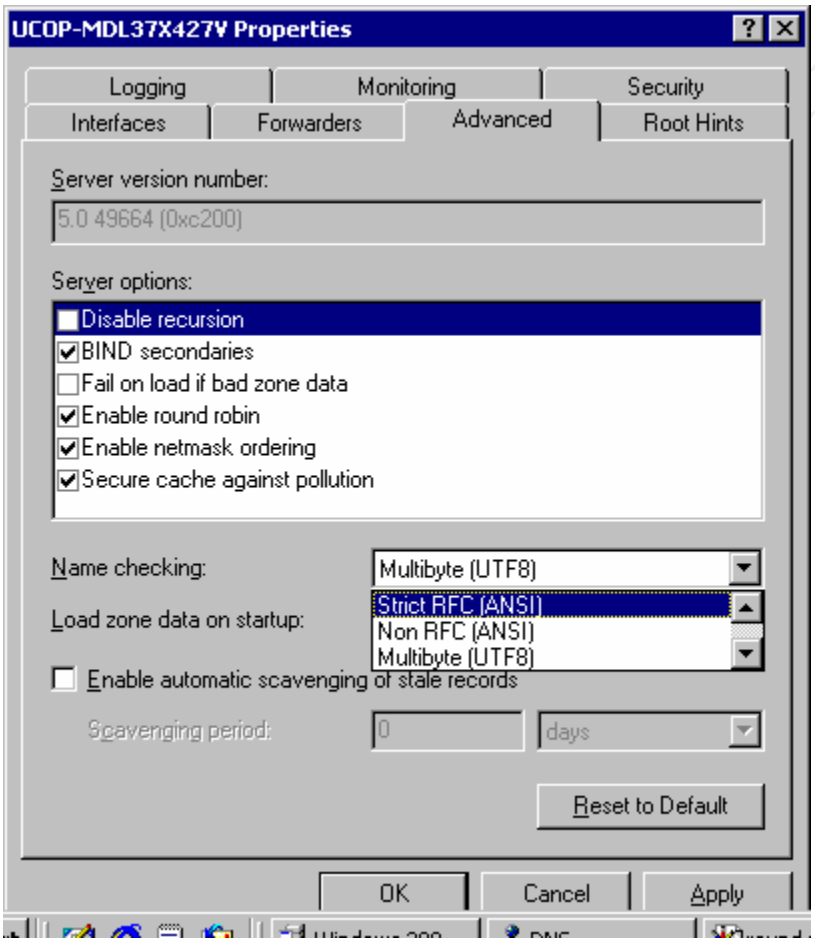
Even after you've set this option, you may have a problem. Clients may get varying answers if the primary hosts WINS records and the secondary does not. A way to configure your domain to get around this problem is to create a WINS referral zone. What that means is that you create a child zone that is authoritative for the records in WINS only. For example, if your top zone is called example.org, you could create a WINS referral zone called wins.example.org. Your first step is to set up a Windows 2000 DNS server with no records in it except the WINS records. All of the lookups that come to the DNS server will be forwarded to WINS. Then, in the parent zone, you set up a delegation to the wins zone. This enables clients to find the WINS zone that you set up. Last, you configure your clients to include the wins.example.org zone in their DNS suffix search list. This will cause the client to search the WINS zone when a name is not found elsewhere.

Another slight issue has to do with underscores. This merits some explanation, as there has been quite a bit of confusion about Microsoft's use of underscores in SRV records. The confusion stems from misinterpretation of RFC 1123. That RFC restricts host names to the characters a-z, 0-9, and hyphen characters.[76] The concept that clears up the confusion is the distinction between host names and DNS labels. In DNS A records[77] and also in SRV records[78], host names show up in the RDATA section of the record. The underscores that some have complained about are part of the "owner name" part of the record. There is no requirement to name any host according to these owner names. This means using underscores in owner names is irrelevant to host naming and to compliance with RFC 1123. [79] In fact, Microsoft's implementation of SRV owner names is compliant with the format outlined in the relevant RFC, 2782, which specifies underscores. [80] Also, RFC 2181 states that DNS labels may be composed of any binary string. [81] However, not all implementations of DNS comply with 2181. There are even servers out there that balk at the use of characters not specified in RFC 952, an older host-naming standard. They may be unable to store the records, even if they can accept them during zone transfers. In that case, you should change the "check-name" setting from the default, fail, to the warn or ignore setting. [82]

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

The UTF-8 option brings up another character-use issue. This option allows you to have an expanded character set for host naming, one that includes international characters. It includes characters from most of the world's written languages. This might be useful for organizations that have NetBIOS host names with characters not normally allowed as Internet host names. It would allow them to go on using the same names, where otherwise they might need to change them to conform to 1123 standards. However, if you choose to use UTF-8, you lose interoperability, since other DNS software is not implementing this Internet draft yet. Note that UTF-8 for host naming is an Internet draft, and not yet an RFC.[83]

To enable UTF-8 name checking, open the server properties and click the advanced tab. Select "Multibyte (UTF8)" from the drop-down list. For better interoperability, choose the "Strict RFC (ANSI)" setting.



The meaning of the choices offered is as follows:
Strict RFC (ANSI) allows A-Z, a-z, hyphen, and underscore (_) as the first character.
Non RFC (ANSI) allows all characters that Strict does, and allows underscore anywhere in a name.
Multibyte (UTF-8) allows characters specified by the UTF-8 character set.
"Any character" Allows any character, including UTF-8 characters.[84]

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Concern number six: Whenever you convert an AD integrated server to a standard primary, the DNS records are deleted from all DNS for that zone, not only the server converted. This is only a problem when a secondary has been configured to accept transfers from one of the other masters. In that case, you would be left with an orphan secondary until you reconfigured it to point to the new primary. [85] To avoid the problem, do not convert an integrated zone to a primary unless you are sure no secondaries depend on other DNS servers in the zone. Even better, do not use secondaries at all.

Now that we've looked at the features that affect integration, let's take a look at the options for integrating Microsoft DNS into the organization. You can try to use only Microsoft; have it act as primary for zones where non-Microsoft DNS servers are secondary; or segregate Microsoft DNS into its own zone. I am leaving out the option to set up Microsoft servers as secondary because, as previously discussed, that option is unreasonably insecure.

If you use only Windows DNS, of course, you don't worry about compatibility with other DNS servers. You can use all of Windows' new security features without breaking any DNS functionality. This is the easiest route if you are setting up a new name space or if you haven't invested heavily in other DNS servers. [86] There is at least one compatibility issue in this case, though: DNS clients. If you have non-Windows systems that need to store records in DNS, they cannot achieve secure dynamic update on your Windows DNS because they do not have a Windows account. [87] ISC hopes to eventually include GSS-TSIG in BIND clients, but they have neither the funding nor sufficient information from Microsoft to do so yet. [88] You will need to manually update the non-Windows records. That means you need to set up a separate zone that does not accept updates. Since you need to create a separate zone, it makes more sense to use one of the other options. The only case when you should use only Windows DNS is when all of the systems that need to update DNS run Windows.

Your second option would be to integrate Windows DNS into a zone that includes other types of servers. In that case, you need to make the other servers secondary to Microsoft AD integrated zones to retain security, since AD integrated zones cannot act as secondaries. When you choose this configuration, Windows performs transfers to secondaries using IXFR and fast transfer if possible, but not secure transfer. You will have to deal with the following concerns: 1) As mentioned above, you cannot use this option if you have non-Windows hosts that need to update DNS, unless you make the zone static. 2) You may have WINS problems with your secondaries. If you need to run WINS, it's probably better not to use this option, since you end up having to create a sub-zone for referral. 3) You need to be sure your secondaries can store SRV records, and UTF-8 if you are using it.

Tao Zhou explains in "Integrating UNIX DNS with Windows 2000 in Windows 2000 Magazine for February 2000 that you can use BIND's nsupdate utility to find out whether a server supports SRV records. First, create a test domain in the target server. Make sure dynamic update is on and supported from your client system. Start nsupdate and issue the following commands:

"> update add webdev.acme.com. 86400 IN A 192.168.1.10

> update add _http._tcp.acme.com. 3600 IN SRV 0 1 80 webdev.acme.com.
> "

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

These commands first add an A record for a web server, and then add the SRV record for the web service. The blank line sends the commands to the server. After you send these commands, you can use nslookup to see whether the records have both been added.

Here's the recommended option: You can skip the hassles caused by using Windows servers in the same zone as others if you segregate the AD-integrated zones from the others. The recommended method is to create a sub-domain that contains only Windows DNS. All Windows systems can be assigned names from this space. For example, if your top-level zone is named example.org, you might create a zone named windows.example.org. When you assign user names, they will look like userid@windows.example.org. All of the Windows systems can be members of the sub-domain. With this scheme, none of your other servers have to replicate with or update to Windows, which prevents compatibility problems.

To set this up, install your Windows 2000 domain controller with DNS running on it, using the sub-domain name you chose. The domain you set up will be a Windows root domain. Note that a Windows root domain is not the same thing as a DNS root domain. Then have all of your Windows 2000 systems join this domain. Set up a delegation to the sub-domain on your parent DNS server. The example.org domain would have a delegation for windows.example.org. This enables resolvers to find your sub-domain. For reverse lookups, you should assign IP addresses from a separate scope, too. What you might need to do is put the Windows machines on a separate subnet. That way, the reverse lookups for that subnet can be delegated to the WIndows servers, just as the forward lookups are.

You do also have the option of using only BIND for DNS. If you do, you need to make sure your server supports SRV records to support Microsoft servers. Dynamic update is helpful, but not required. You can add the SRV records manually. If your server does not support SRV records, you need to delegate the following zones to one that does:
_tcp.<*domain name*>, _udp.<domain name>,
_msdcs.<Active Directory domain name> and
_sites. <Active Directory domain name>.[89]

Security for the Internet
Microsoft has created a secure DNS. It includes features to secure almost every aspect of naming services. In the year and a half since Windows 2000 was released, no vulnerabilities specific to this DNS have been reported. However, it's only secure in the context of Microsoft systems, since non-Windows systems cannot be authenticated to DNS.

This is a problem. DNS is not a self-contained system; the whole Internet uses it. That means lots of different kinds of systems! To secure the entire DNS, cooperation is required. There must be open standards so that products on any system can implement security that will work with other product's security. What Microsoft has done is created a closed system. They have not cooperated with the work that the IETF is doing for DNS security, and they have not provided a standard that others can write to, either. The result for the Internet is to delay and obstruct overall DNS security.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Appendices

Appendix A - How dynamic update uses class and type to specify action required

The following table is taken directly from RFC 2136:

Table Of Metavalues Used In Prerequisite Section

CLASS	TYPE	RDATA	Meaning
ANY	ANY	empty	Name is in use
ANY	rrset	empty	RRset exists (value independent)
NONE	ANY	empty	Name is not in use
NONE	rrset	empty	RRset does not exist
zone	rrset	rr	RRset exists (value dependent)

When a client sends an update request, it may send resource records in the prerequisite section. All of the resource records must belong to the same zone. Each resource record in the prerequisite section represents a condition that must be met before the update can take place. The table above summarizes how these resource records represent a prerequisite condition. For example, the class and type of the prerequisite record are both "ANY" and the record contains no data, the prerequisite is that the name specified in the RR is in use as a name in the zone, regardless of what type or RDATA may be found in the relevant records. If the class is "ANY" and the type is specified, the prerequisite is that a record(s) exist that match the name and type in the prerequisite. If the class is "NONE" and the type is "ANY" the prerequisite is that the name in the RR is not used as a name in any records in the zone. If the class is "NONE" and the type is not is specified, the prerequisite is that no records exist with the specified name and type. And last, a prerequisite record may specify class, type and rdata when the prerequisite is that all three must be matched in some zone record.

To specify what update actions should be taken, a client sends resource records in the update section. The server process them according to the following table:

CLASS	TYPE	RDATA	Meaning
ANY	ANY	empty	Delete all RRsets from a name
ANY	rrset	empty	Delete an RRset
NONE	rrset	rr	Delete an RR from an RRset
zone	rrset	rr	Add to an RRset

To specify that the server should delete resource records of a particular name, the client sends a resource record with class and type "ANY" and RDATA section empty. The client can specify the type if it wants to delete records of a particular type for a name. The client can also specify RDATA, but in that case the class is supposed to be set to NONE. When the client wants to add a record, it specifies the class, type and rdata in addition to the name.[90]

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Appendix B – Using DNSCMD and IPCONFIG[91]

Dnscmd is a command line tool that enables you to accomplish, in scripts, the same administration tasks that you do using the DNS console. You may find dnscmd.exe on the Windows 2000 Server CD. To use it, type "Dnscmd" follow by the name of the DNS server that you want to connect to, and one of the commands in the following table. If you are running Dnscmd on the server you want to work with, you can use "." to specify the server. Each of the syntax illustrations shown here is taken from the command line help.

/Info [<Property>]: Output shown with no properties specified. Properties can be: BootMethod, RpcProtocol, LogLevel, EventlogLevel, NoRecursion, ForwardDelegations, ForwardingTimeout, IsSlave, SecureResponses, RecursionRetry, RecursionTimeout, MaxCacheTtl, MaxNegativeCacheTtl, RoundRobin, LocalNetPriority, AddressAnswerLimit, BindSecondaries, WriteAuthorityNs, NameCheckFlag, StrictFileParsing, UpdateOptions, DisableAutoReverseZones, SendPort, NoTcp, XfrConnectTimeout, DsPollingInterval, DsTombstoneInterval, ScavengingInterval, DefaultAgingState, DefaultNoRefreshInterval, DefaultRefreshInterval	Server info: ptr = 00075AE8 server name = ucop-mdl37x427v version = C2000005 DS container = c Configuration: dwLogLevel = 0000000 dwDebugLevel = 0000000 dwRpcProtocol = FFFFFFF dwNameCheckFlag = 0000000 cAddressAnswerLimit = 0 dwRecursionRetry = 3 dwRecursionTimeout = 15 dwDsPollingInterval = 300 Configuration Flags: fBootMethod = 3 fAdminConfigured = 1 fAllowUpdate = 1 fDsAvailable = 1 fAutoReverseZones = 1 fAutoCacheUpdate = 0 fSlave = 0 fNoRecursion = 0 fRoundRobin = 1 fLocalNetPriority = 1 fStrictFileParsing = 0 fLooseWildcarding = 0 fBindSecondaries = 1 fWriteAuthorityNs = 0 Aging Configuration: ScavengingInterval = 0 DefaultAgingState = 0 DefaultRefreshInterval = 168 DefaultNoRefreshInterval = 168 ServerAddresses: Addr Count = 1 Addr[0] => 192.168.21.38

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

	ListenAddresses: NULL IP Array. Forwarders: NULL IP Array. forward timeout = 5 slave = 0
/Config [<ZoneName> |..AllZones] <Property> <Value> Zone properties: /SecureSecondaries /AllowUpdate /Aging /RefreshInterval /NoRefreshInterval	This command enables you to set the value of a server or zone property. The available property list is the same as for the Info command except that you don't have BootMethod, and you do have five zone-specific options as listed at left. Example: Dnscmd . /config example.org scavenginginterval 7 Note that the commands are case insensitive.
/EnumZones [/primary | /Secondary | /Cache | /Auto-Created] [/Forward | /Reverse] This command lists the zones on a server. You can narrow the list to primaries, secondaries, forward zones, etc. Example output with no options on:	Zone Count = 8. 0 file Up=0 1 DS Up=2 0.in-addr.arpa 1 file Rev Auto Up=0 127.in-addr.arpa 1 file Rev Auto Up=0 255.in-addr.arpa 1 file Rev Auto Up=0 example.org 1 DS Up=2 vsb.burns.fremont.ca.us 1 DS Up=2 vsb.local 1 DS Up=2
/Statistics [<Filter> | /Clear] Filters available: 00000001 -- Time 00000002 -- Query 00000004 -- Query2 00000008 -- Recurse 00000010 -- Master 00000020 -- Secondary 00000040 -- Wins 00000100 -- Update 00000200 -- SkwanSec	Example: dnscmd . /statistics 00000002 DNS Server . statistics: Queries and Responses: ---------- Total: Queries Received = 0 Responses Sent = 0 UDP: Queries Recvd = 0 Responses Sent = 0 Queries Sent = 0 Responses Recvd = 0

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

00000400 -- Ds 00010000 -- Memory 00100000 -- PacketMem 00040000 -- Dbase 00080000 -- Records 00200000 -- NbstatMem	TCP: Client Connects = 0 Queries Recvd = 0 Responses Sent = 0 Queries Sent = 0 Responses Recvd = 0
/clearcache	Takes no parameters and does exactly what it says.
/WriteBackFiles [<ZoneName>]	Writes the zone datafile. Example: dnscmd . /writebackfiles example.org
/startscavenging	You can force scavenging to start using this command.
/ResetListenAddresses [<ListenAddress>]	Use this command to specify which interface a server should listen on.
/ResetForwarders [<IPAddress>] ...] [/[No]Slave] [/TimeOut <Time>]	Set DNS servers to forward recursive queries to. Example: dnscmd . /resetforwarders 192.168.21.99 /timeout 10
/ZoneInfo <ZoneName> [<Property>] Properties are as listed in the /config command.	dnscmd . /zoneinfo vsb.local Zone query result: Zone info: ptr = 000754A8 zone name = vsb.local zone type = 1 update = 2 DS integrated = 1 data file = (null) using WINS = 0 using Nbstat = 0 aging = 0 refresh interval = 168 no refresh = 168 scavenge available = 3511718 Zones Masters NULL IP Array. Zone Secondaries NULL IP Array. secure secs = 0
/ ZoneAdd <ZoneName> <ZoneType> [<Options>] The type is /primary, /secondary, or /dsprimary. Options are for filename and admin email.	Example: dnscmd . /zoneadd test.org /secondary 192.168.21.99
/ZoneDelete <ZoneName>	Example: dnscmd . /zonedelete test.org /f

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

[/DsDel] [/f] DsDel deletes the zone from the directory. The /f option is to force the delete.	DNS Server . deleted zone test.org: Status = 0 (0x00000000)
/ZonePause <ZoneName>	Self-explanatory
/ZoneResume <ZoneName>	Self-explanatory
/ZoneReload <ZoneName>	Reloads the zone from disk
/ZoneWriteBack <ZoneName>	Writes the zone to disk
/ZoneRefresh <ZoneName>	Force transfer from master.
/ZoneUpdateFromDs <ZoneName>	Refresh zone from directory services.
ZoneResetType <ZoneName> <Property> [<Options> The property is primary, secondary, etc. Options are for overwriting DS or DNS.	Example: dnscmd . /zoneresettype example.org /dsprimary
> /ZoneResetSecondaries <ZoneName> [<Security>] [<SecondaryIPAddress>] ...] [<Notify>] [<NotifyIPAddress>] ...]	Example: dnscmd /zoneresetsecondaries example.org /securelist 192.168.21.99 /notify
/ZoneResetScavengeServers <ZoneName> [<Server IPs>]	Specify what server should scavenge the zone.Example: dnscmd . /zoneresetscavengeservers example.org 192.168.21.99
/ZoneResetMasters <ZoneName> [<Server IPs>]	Specify masters for a secondary. Example: Dnscmd . /zoneresetmasters example.org 192.168.21.99
/EnumRecords <ZoneName> <NodeName> [<DataOptions>] [<ViewOptions>] Display RRs for a zone. Options are /Type, /Authority, /Glue, /Additional, /Node, /Child, /StartChild, /Continue, (on full buffer condition), /Detail	dnscmd . /enumrecords example.org @ Returned records: @ 3600 NS ucop-mdl37x427v.vsb.local. 3600 SOA ucop-mdl37x427v.vsb.local. administrator.vsb.local. 1 900 600 86400 3600
/RecordAdd <Zone> <NodeName> [/Aging] [<Ttl>] <RRType> <RRData>	Example: dnscmd . /recordadd example.org bill.example.org. A 192.168.21.45

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

/RecordDelete <Zone> <OwnerName> <RRType> <RRData> [/f]	Example: dnscmd . /recorddelete example.org bill.example.org. A 192.168.21.45
/NodeDelete <Zone> <NodeName> [/Tree] [/f]	dnscmd . /nodedelete example.org bill Are you sure you want to delete node? (y/n) y DNS Server . deleted node at bill.example.org: Status = 0 (0x00000000)
/AgeAllRecords <ZoneName> [<NodeName>] [/Tree]	Force aging on specified node(s)

Ipconfig also can be useful. In Windows 2000, you can flush the DNS cache with it.

Example: ipconfig /flushdns

You can also display the cache:

Example: ipconfig /displaydns

I flushed the cache, then visited Google, without doing a search. Here is the resulting display:

```
ns1.google.com.
----------------------------------------------------
  Record Name . . . . . : NS1.google.com
  Record Type . . . . . : 1
  Time To Live  . . . . : 86359
  Data Length . . . . . : 4
  Section . . . . . . . : Answer
  A (Host) Record . . . : 216.239.32.10


localhost.
----------------------------------------------------
  Record Name . . . . . : localhost
  Record Type . . . . . : 1
  Time To Live  . . . . : 31524516
  Data Length . . . . . : 4
  Section . . . . . . . : Answer
  A (Host) Record . . . : 127.0.0.1


ns4.google.com.
----------------------------------------------------
  Record Name . . . . . : NS4.google.com
  Record Type . . . . . : 1
  Time To Live  . . . . : 86359
  Data Length . . . . . : 4
  Section . . . . . . . : Answer
  A (Host) Record . . . : 216.239.38.10
```

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

ns2.google.com.

Record Name . . . . . : NS2.google.com
Record Type . . . . . : 1
Time To Live . . . . : 86359
Data Length . . . . . : 4
Section . . . . . : Answer
A (Host) Record . . . : 216.239.34.10

www.google.com.

Record Name . . . . . : www.google.com
Record Type . . . . . : 1
Time To Live . . . . : 127
Data Length . . . . . : 4
Section . . . . . : Answer
A (Host) Record . . . : 216.239.37.100

Record Name . . . . . : google.com
Record Type . . . . . : 2
Time To Live . . . . : 127
Data Length . . . . . : 4
Section . . . . . : Authority
NS Record . . . . . : NS2.google.com

Record Name . . . . . : google.com
Record Type . . . . . : 2
Time To Live . . . . : 127
Data Length . . . . . : 4
Section . . . . . : Authority
NS Record . . . . . : NS1.google.com

Record Name . . . . . : google.com
Record Type . . . . . : 2
Time To Live . . . . : 127
Data Length . . . . . : 4
Section . . . . . : Authority
NS Record . . . . . : NS3.google.com

Record Name . . . . . : google.com
Record Type . . . . . : 2
Time To Live . . . . : 127
Data Length . . . . . : 4
Section . . . . . : Authority
NS Record . . . . . : NS4.google.com

Record Name . . . . . : NS2.google.com
Record Type . . . . . : 1
Time To Live . . . . : 127
Data Length . . . . . : 4
Section . . . . . : Additional
A (Host) Record . . . : 216.239.34.10

Record Name . . . . . : NS1.google.com
Record Type . . . . . : 1

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Time To Live . . . . : 127
Data Length . . . . . : 4
Section . . . . . : Additional
A (Host) Record . . . : 216.239.32.10

Record Name . . . . . : NS3.google.com
Record Type . . . . . : 1
Time To Live . . . . : 127
Data Length . . . . . : 4
Section . . . . . : Additional
A (Host) Record . . . : 216.239.36.10

Record Name . . . . . : NS4.google.com
Record Type . . . . . : 1
Time To Live . . . . : 127
Data Length . . . . . : 4
Section . . . . . : Additional
A (Host) Record . . . : 216.239.38.10

1.0.0.127.in-addr.arpa.

Record Name . . . . . : 1.0.0.127.in-addr.arpa
Record Type . . . . . : 12
Time To Live . . . . : 31524516
Data Length . . . . . : 4
Section . . . . . : Answer
PTR Record . . . . . : localhost

ns3.google.com.

Record Name . . . . . : NS3.google.com
Record Type . . . . . : 1
Time To Live . . . . : 86359
Data Length . . . . . : 4
Section . . . . . : Answer
A (Host) Record . . . : 216.239.36.10

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Appendix C - SRV, TKEY and TSIG records

From RFC 2782, here is the format of the SRV RR, whose DNS type code is 33:[92]

_Service._Proto.Name TTL Class SRV Priority Weight Port Target

Service is the name of the service, such as ldap or http. The underscore is included to avoid creating the same names as host DNS names.

Proto is the name of the protocol, _TCP or _UDP.

Name is the domain name, such as example.org.

TTL is the usual DNS time to live

Class, for SRV records is always IN.

Priority is a number from 0 – 65535. This field enables the administrator to specify which host should be tried first. When an SRV request returns multiple records, a client should attempt contact with the host having the lowest-numbered priority field.

Weight is a number from 0-65535. When each of a set of records has the same priority, the client should rely on the weight to determine which host to contact first. Otherwise, this field is used for the server determine which records should have a more frequently be selected. Usually 0.

Port is the tcp or udp port used by the service.

Target is the domain name of the host running the service. To specify the absence of a service, this field may be "."

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

From RFC 2930, here is the format of the TKEY meta resource record: (type 249): [93]

NAME	The name that identifies the key
TTYPE	= 249
CLASS	ignored; SHOULD be 255
TTL	SHOULD be zero
RDLEN	size of RDATA
RDATA	includes the following
Algorithm:	see below
Inception:	see below
Expiration:	see below
Mode:	see below
Error:	see below
Key Size:	the size of the key exchange data field in octets.
Key Data:	the material being transmitted
Other Size:	not used
Other Data:	not used

The *algorithm* name looks like a domain name e.g. sample-alg.org.

The *inception* and *expiration* times are the validity interval for the TKEY. Each specifies a time in number of seconds since 1/1/80.

The *mode* field is a numeric value from 1-5. Each of the values represents one of these modes:

1 server assignment
2 Diffie-Hellman exchange
3 GSS-API negotiation
4 resolver assignment
5 key deletion

The server identifies errors using the *error* code field. The TKEY specific codes are:

16 BADSIG (TSIG)
17 BADKEY (TSIG)
18 BADTIME (TSIG)
19 BADMODE
20 BADNAME
21 BADALG

© SANS Institute 2000 - 2002 Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

From RFC 2845, here is the format of the TSIG meta resource record: [94]

NAME: Identification of the key used.
TYPE: 250
CLASS: ANY
TTL: 0
RdLen: Length of RDATA
RDATA:
- Algorithm Name: Name of the algorithm in domain name syntax, see above.
- Time Signed: stated in seconds since 1-Jan-70.
- Fudge: seconds of error permitted in Time Signed.
- MAC Size: number of octets in MAC.
- MAC: The product of the hash algorithm
- Original ID: original message ID
- Error: see below.
- Other Len: length, in octets, of Other Data.
- Other Data: empty unless Error == BADTIME

TSIG-specific RCODES:
- ERROR = 16 (BADSIG)
- ERROR = 17 (BADKEY)
- ERROR = 18 (BADTIME)

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Appendix D - Performance Counters

The following performance counters are available for use with DNS, according to Microsoft's Performance Counters Reference.[95]

Counter Name	Description	Counter Type
AXFR Request Received	Shows the total number of full zone transfer requests received by the master DNS server.	PERF_COUNTER_RAWCOUNT
AXFR Request Sent	Shows the total number of full zone transfer requests sent by the secondary DNS server.	PERF_COUNTER_RAWCOUNT
AXFR Response Received	Shows the total number of full zone transfer responses received by the secondary DNS server.	PERF_COUNTER_RAWCOUNT
AXFR Success Received	Shows the total number of successful full zone transfers received by the secondary DNS server.	PERF_COUNTER_RAWCOUNT
AXFR Success Sent	Shows the total number of successful full zone transfers of the master DNS server.	PERF_COUNTER_RAWCOUNT
Caching Memory	Shows the total amount of caching memory used by the DNS server.	PERF_COUNTER_RAWCOUNT
Database Node Memory	Shows the total database node memory used by the DNS server.	PERF_COUNTER_RAWCOUNT
Dynamic Update NoOperation	Shows the total number of No-operation/empty dynamic update requests received by the DNS server.	PERF_COUNTER_RAWCOUNT
Dynamic Update NoOperation/sec	Shows the rate at which No-operation/empty dynamic update requests are received by the DNS server.	PERF_COUNTER_COUNTER
Dynamic Update Queued	Shows the total number of dynamic updates that are queued by the DNS server.	PERF_COUNTER_RAWCOUNT
Dynamic Update Received	Shows the total number of dynamic update requests that are received by the DNS server.	PERF_COUNTER_RAWCOUNT
Dynamic Update Received/sec	Shows the rate at which dynamic update requests are received by the DNS server.	PERF_COUNTER_COUNTER
Dynamic Update Rejected	Shows the total number of dynamic updates rejected by the DNS server.	PERF_COUNTER_RAWCOUNT
Dynamic Update TimeOuts	Shows the total number of dynamic update timeouts of the DNS server.	PERF_COUNTER_RAWCOUNT
Dynamic Update Written to	Shows the total number of dynamic updates written to the	PERF_COUNTER_RAWCOUNT

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Database	database by the DNS server.	
Dynamic Update Written to Database/sec	Shows the rate at which dynamic updates are written to the database by the DNS server.	PERF_COUNTER_COUNTER
IXFR Request Received	Shows the total number of incremental zone transfer requests received by the master DNS server.	PERF_COUNTER_RAWCOUNT
IXFR Request Sent	Shows the total number of incremental zone transfer requests sent by the secondary DNS server.	PERF_COUNTER_RAWCOUNT
IXFR Response Received	Shows the total number of incremental zone transfer responses received by the secondary DNS server.	PERF_COUNTER_RAWCOUNT
IXFR Success Received	Shows the total number of successful incremental zone transfers received by the secondary DNS server.	PERF_COUNTER_RAWCOUNT
IXFR Success Sent	Shows the total number of successful incremental zone transfers of the master DNS server.	PERF_COUNTER_RAWCOUNT
IXFR TCP Success Received	Shows the total number of successful TCP incremental zone transfers received by the secondary DNS server.	PERF_COUNTER_RAWCOUNT
IXFR UDP Success Received	Shows the total number of successful UDP incremental zone transfers received by the secondary DNS server.	PERF_COUNTER_RAWCOUNT
Nbstat Memory	Shows the total Nbstat memory used by the DNS server.	PERF_COUNTER_RAWCOUNT
Notify Received	Shows the total number of notifies received by the secondary DNS server.	PERF_COUNTER_RAWCOUNT
Notify Sent	Shows the total number of notifies sent by the master DNS server.	PERF_COUNTER_RAWCOUNT
Record Flow Memory	Shows the total record flow memory used by the DNS server.	PERF_COUNTER_RAWCOUNT
Recursive Queries	Shows the total number of recursive queries received by the DNS server.	PERF_COUNTER_RAWCOUNT
Recursive Queries/sec	Shows the rate at which recursive queries are received by the DNS server.	PERF_COUNTER_COUNTER
Recursive Query Failure	Shows the total number of recursive query failures.	PERF_COUNTER_RAWCOUNT
Recursive Query Failure/sec	Shows the rate of recursive query failures.	PERF_COUNTER_COUNTER
Recursive Send TimeOuts	Shows the total number of recursive query sending timeouts.	PERF_COUNTER_RAWCOUNT

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Recursive TimeOut/sec	Shows the rate of recursive query sending timeouts.	PERF_COUNTER_COUNTER
Secure Update Failure	Shows the total number of secure update failures of the DNS server.	PERF_COUNTER_RAWCOUNT
Secure Update Received	Shows the total number of secure update requests received by the DNS server.	PERF_COUNTER_RAWCOUNT
Secure Update Received/sec	Shows the rate at which secure update requests are received by the DNS server.	PERF_COUNTER_COUNTER
TCP Message Memory	Shows the total amount of TCP message memory used by the DNS server.	PERF_COUNTER_RAWCOUNT
TCP Query Received	Shows the total number of TCP queries received by the DNS server.	PERF_COUNTER_RAWCOUNT
TCP Query Received/sec	Shows the rate at which TCP queries are received by the DNS server.	PERF_COUNTER_COUNTER
TCP Response Sent	Shows the total number of TCP responses sent by the DNS server.	PERF_COUNTER_RAWCOUNT
TCP Response Sent/sec	Shows the rate at which TCP responses are sent by the DNS server.	PERF_COUNTER_COUNTER
Total Query Received	Shows the total number of queries received by the DNS server.	PERF_COUNTER_RAWCOUNT
Total Query Received/sec	Shows the rate at which queries are received by the DNS server.	PERF_COUNTER_COUNTER
Total Response Sent	Shows the total number of responses sent by the DNS server.	PERF_COUNTER_RAWCOUNT
Total Response Sent/sec	Shows the rate at which responses are sent by the DNS server.	PERF_COUNTER_COUNTER
UDP Message Memory	Shows the total UDP message memory used by the DNS server.	PERF_COUNTER_RAWCOUNT
UDP Query Received	Shows the total number of UDP queries received by the DNS server.	PERF_COUNTER_RAWCOUNT
UDP Query Received/sec	Shows the rate at which UDP queries are received by the DNS server.	PERF_COUNTER_COUNTER
UDP Response Sent	Shows the total number of UDP responses sent by the DNS server.	PERF_COUNTER_RAWCOUNT
UDP Response Sent/sec	Shows the rate at which UDP responses are sent by the DNS server.	PERF_COUNTER_COUNTER

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

WINS Lookup Received	Shows the total number of WINS lookup requests received by the DNS server.	PERF_COUNTER_RAWCOUNT
WINS Lookup Received/sec	Shows the rate at which WINS lookup requests are received by the DNS server.	PERF_COUNTER_COUNTER
WINS Response Sent	Shows the total number of WINS lookup responses sent by the DNS server.	PERF_COUNTER_RAWCOUNT
WINS Response Sent/sec	Shows the rate at which WINS lookup responses are sent by the server.	PERF_COUNTER_COUNTER
WINS Reverse Lookup Received	Shows the total number of WINS reverse lookup requests received by the DNS server.	PERF_COUNTER_RAWCOUNT
WINS Reverse Lookup Received/sec	Shows the rate at which WINS reverse lookup requests are received by the DNS server.	PERF_COUNTER_RAWCOUNT
WINS Reverse Response Sent	Shows the total number of WINS Reverse lookup responses sent by the DNS server.	PERF_COUNTER_RAWCOUNT
WINS Reverse Response Sent/sec	Shows the rate at which WINS Reverse lookup responses are sent by the server.	PERF_COUNTER_COUNTER
Zone Transfer Failure	Shows the total number of failed zone transfers of the master DNS server.	PERF_COUNTER_RAWCOUNT
Zone Transfer Request Received	Shows the total number of zone transfer requests received by the master DNS server.	PERF_COUNTER_RAWCOUNT
Zone Transfer SOA Request Sent	Shows the total number of zone transfer start of authority (SOA) requests sent by the secondary DNS server.	PERF_COUNTER_RAWCOUNT
Zone Transfer Success	Shows the total number of successful zone transfers of the master DNS server.	PERF_COUNTER_RAWCOUNT

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Appendix E - Selected DNS Registry Entries

The following registry entries, may be of interest when securing DNS:

Key: AutoConfigFileZones
Path: HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters
Type: REG_DWORD
Value: 0 | 1 | 2 | 3
Default: 1
Determines whether DNS server updates RRs on primary zones when the local fully qualified domain name changes.
0 = No updates
1 = Update zones that permit dynamic updates
2 = Update zones that do not permit dynamic updates
3 = Update all primary zones.
The documentation of this registry key is found at
http://www.microsoft.com/windows2000/techinfo/reskit/en/regentry/94015.htm

Key: StrictFileParsing
Path: HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters
Type: REG_DWORD
Value: 0 | 1
Default: 0
Determines whether DNS allows records that violate RFCs.
0 = Errors are logged and allowed
1 = Zone does not load when the records are encountered.
You can change this value using the "Fail on load if bad zone data" option on the advanced tab of the server properties. The documentation for this registry key is found at
http://www.microsoft.com/windows2000/techinfo/reskit/en/regentry/46768.htm

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Key: UpdateOptions
Path: HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters
Type: REG_DWORD
Value: Ranges 0x0–0x80000000
Default: 0x30F
Prevents dynamic update for specified types of records. This entry is a bitmask. To disable dynamic update for types listed, set the specified bits to 1.

Value	Meaning
0x0	DNS dynamic update does not restrict any record types.
0x1	SOA (Start of Authority) records.
0x2	NS (name server) records.
0x4	Delegation NS records.
0x8	Server host records.
0x100	On secure dynamic update, exclude SOA records.
0x200	On secure dynamic update, exclude root NS records.
0x30F	On standard dynamic update, exclude NS, SOA, and server host records. On secure dynamic update, exclude root NS and SOA records. Allow delegations and server host updates.
0x400	On secure dynamic update, exclude delegation NS records.
0x800	On secure dynamic update, exclude server host records.
0x1000000	DS peer records.
0x80000000	Disable DNS dynamic update.

The documentation for this registry entry can be found at:
http://www.microsoft.com/windows2000/techinfo/reskit/en/regentry/94016.htm

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

CLIENT BEHAVIOR. The next set of registry entries apply to client behavior:

Key: DefaultRegistrationRefreshInterval
Path: HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
Type: REG_DWORD
Value range: 0x0–0xFFFFFFFF (seconds)
Default: 0x15180 (86,400 seconds = 24 hours)
By default, clients register their DNS records every 24 hours. You can change this frequency by specifying an interval in this registry key.
Information regarding this registry entry can be found at:
http://support.microsoft.com/support/kb/articles/Q246/8/04.ASP

Key: UpdateSecurityLevel
Path: HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
Type: REG_DWORD
Value range: 0 | 16 | 256
Default: 0
Determines whether the DNS client uses standard secure dynamic update.
0 = Send secure only when non-secure is refused
16 = Non-secure only
256= Secure only
Documentation for this registry key can be found at:
http://www.microsoft.com/windows2000/techinfo/reskit/en/regentry/94187.htm

Key: DisableDynamicUpdate
Path: HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces*interface-name*
Type: REG_DWORD
Values: 0 | 1
Default: 0
Disables dynamic update registration on a specified interface.
0 enables, 1 disables.
Documentation for this registry entry can be found at:
http://www.microsoft.com/windows2000/techinfo/reskit/en/regentry/94184.htm

Key: DisableReplaceAddressesInConflicts
Path: HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
Type: REG_DWORD
Values: 0 | 1
Default: 0
Prevents the client from overwriting an existing RR when an address conflict exists during insecure dynamic update.
0 = overwrite
1 = do not overwrite
Documentation for this registry entry can be found at:
http://www.microsoft.com/windows2000/techinfo/reskit/en/regentry/94185.htm

Key: DisableReverseAddressRegistrations
Path: HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
Type: REG_DWORD
Values: 0 | 1
Default: 0
Disables client registration of PTR records.
0 = register PTR records
1 = do not register
Documentation for this registry entry can be found at:
http://www.microsoft.com/windows2000/techinfo/reskit/en/regentry/94186.htm

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Appendix F - Microsoft compliance with RFCs

Microsoft claims compliance with the following RFCs, according to the on-line help: [96]

1034 Domain Names -- Concepts and Facilities
1035 Domain Names -- Implementation and Specification
1123 Requirements for Internet Hosts -- Application and Support
1886 DNS Extensions to Support IP Version 6
1995 Incremental Zone Transfer in DNS
1996 A Mechanism for Prompt DNS Notification of Zone Changes
2136 Dynamic Updates in the Domain Name System (DNS UPDATE)
2181 Clarifications to the DNS Specification
2308 Negative Caching of DNS Queries (DNS NCACHE)

Filename	Title
Draft-ietf-dnsind-rfc2052bis-02.txt	A DNS RR for Specifying the Location of Services (DNS SRV)
Draft-skwan-utf8-dns-02.txt	Using the UTF-8 Character Set in the Domain Name System
Draft-ietf-dhc-dhcp-dns-08.txt	Interaction between DHCP and DNS
Draft-ietf-dnsind-tsig-11.txt	Secret Key Transaction Signatures for DNS (TSIG)
Draft-ietf-dnsind-tkey-00.txt	Secret Key Establishment for DNS (TKEY RR)
Draft-skwan-gss-tsig-04.txt	GSS Algorithm for TSIG (GSS-TSIG)

My notes:

Draft-ietf-dnsind-rfc2052bis-02.txt has since become RFC 2782.
Draft-skwan-utf8-dns-02.txt is now draft-skwan-utf8-dns-06.txt expires 11/01.
Draft-ietf-dhc-dhcp-dns-12.txt expired 3/01.
Draft-ietf-dnsind-tsig-11.txt has since become RFC 2845.
Draft-ietf-dnsind-tkey-00.txt has since become RFC 2930.
Draft-skwan-gss-tsig-04.txt is now draft-ietf-dnsext-gss-tsig-02.txt, expires 9/01.

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Appendix G - Format of WINS and WINS-R Resource Records

A WINS resource record has the following syntax, according to the Microsoft Resource Kit: [97]

<domain> <class> WINS [*<TTL>*] *<Local> <LookupTimeout> <CacheTimeout> <IP address of WINS server>*

domain	@
class:	IN
TTL.	Usual meaning.
Local	Whether the record should be replicated
LookupTimeout	Time in seconds to try WINS lookup before giving up.
CacheTimeout.	Time in seconds that WINS responses may be cached
WINSServers.	IP addresses of the WINS servers.

Example: @ IN WINS LOCAL 5 3600 172.16.72.3

The WINS-R resource record is used for reverse lookups. It has the following syntax:

<domain> <class> WINSR [*<TTL>*] *<Local> <LookupTimeout> <CacheTimeout> <NameResultDomain>*

domain	@
class	IN
TTL	Usual meaning.
Local	Whether the record is replicated.
LookupTimeout	Time that a lookup waits before it giving up.
CacheTimeout	Time that WINS-R responses may be cached.
NameResultDomain	Domain to append to returned NetBIOS names.

Example: @ IN WINS-R LOCAL 5 3600 reskit.com.

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

List of References / Notes:

Instead of only attaching notes to quoted material, I am following standard writing practice by attaching a full reference to any information that needs backing.

1. Microsoft Corporation, "Chapter 6 - Windows 2000 DNS" (hereafter referred to as "DNS chapter"), excerpted from the Windows 2000 Server TCP/IP Core Networking Guide in the Windows 2000 Server Resource Kit, http://www.microsoft.com/TechNet/prodtechnol/windows2000serv/reskit/tcpip/part2/tcpch06.asp? as of 7/30/01.
2. Ibid.
3. Microsoft Corporation, "Dynamic Update Is Set to "None" for a DNS Root Zone" http://support.microsoft.com/support/kb/articles/Q232/1/87.ASP as of 7/30/01.
4. Microsoft, DNS chapter, op.cit79.
5. Windows 2000 Server online help, DNS/Concepts/Resources/Additional Resources/DNS RFCs.
6. M. Ohta, "Incremental Zone Transfer in DNS", RFC 1995, ftp://ftp.isi.edu/in-notes/rfc1995.txt as of 7/30/01.
7. Server help, DNS RFCs, op. cit.
8. P. Vixie, "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, ftp://ftp.isi.edu/in-notes/rfc1996.txt as of 7/30/01.
9. Microsoft, DNS chapter, op.cit.
10. Microsoft Corporation, "Chapter 12 – Access control" excerpted from the Windows 2000 Server resource kit, Distributed Systems Guide http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/distsys/part2/dsgch12.asp as of 7/30/01.
11. Microsoft, DNS chapter, op.cit.
12. Server help, DNS RFCs, op. cit.
13. A. Gulbrandsen et al., "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, ftp://ftp.isi.edu/in-notes/rfc2782.txt as of 7/30/01.
14. Microsoft Corporation, "Chapter 11 – Authentication" (Authentication chapter), excerpt from the Windows 2000 Server Distributed Systems Guide in the resource kit, http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/distsys/part2/dsgch11.asp? as of 7/30/01.
15. RFC 2782, op.cit.
16. Microsoft, DNS chapter, op.cit.
17. F. Yergeau, "UTF-8, a transformation format of ISO 10646", RFC 2279, http://www.ietf.org/rfc/rfc2279.txt as of 7/30/01.
18. Stuart Kwan et al., "Using the UTF-8 Character Set in the Domain Name System", UTF8 draft, http://search.ietf.org/internet-drafts/draft-skwan-utf8-dns-06.txt as of 7/30/01.
19. Microsoft, DNS chapter, op.cit.
20. Server help, DNS RFCs, Op. Cit.
21. Microsoft, DNS chapter, op.cit.
22. Ibid.
23. Ibid.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

24. NSA, "Windows 2000 Security Recommendation Guide", http://nsa2.www.conxion.com/win2k/index.html as of 7/30/01.
25. Windows 2000 server online help, DNS/Configure Zone Properties/Change the Zone Type.
26. Microsoft, Authentication chapter, op. cit.
27. Microsoft, DNS chapter, op.cit.
28. You can verify this by creating a new Active Directory-integrated zone. The property sheet automatically sets the update to secure only.
29. Windows 2000 server online help, DNS/How to/ Manage Zones/Allow only secure dynamic updates.
30. Microsoft, DNS chapter, op.cit.
31. J. Linn, "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, http://www.ietf.org/rfc/rfc2743.txt as 0f 7/30/01.
32. Stuart Kwan et al., "GSS Algorithm for TSIG (GSS-TSIG)" (GSS-TSIG draft), http://www.ietf.org/internet-drafts/draft-ietf-dnsext-gss-tsig-02.txt as 0f 7/30/01.
33. D. Eastlake, 3rd, "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, http://www.ietf.org/rfc/rfc2930.txt as of 7/30/01.
34. Microsoft Corporation, "How to Configure an Authoritative Time Server in Windows 2000", http://support.microsoft.com/support/kb/articles/Q216/7/34.ASP as of 7/30/01.
35. RFC 2930, op.cit.
36. Graphic from Stuart Kwan et al., "GSS-TSIG <draft-dnsext-gss-tsig-00.txt>" PowerPoint presentation given at July 2000 IETF proceedings, http://www.ietf.org/proceedings/00jul/SLIDES/dnsext-gss-tsig/sld004.htm as of 8/1/01.
37. Graphic, ibid., sld005.
38. Information in preceding several paragraphs: GSS-TSIG draft, op.cit.
39. P. Vixie et al., "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, ftp://ftp.isi.edu/in-notes/rfc2845.txt as of 7/30/01
40. All owner names in the following table are described in Microsoft Corporation, "SRV Resource Records", an excerpt from the Windows 2000 Server Resource Kit, http://www.microsoft.com/windows2000/techinfo/reskit/en/distrib/dsbc_nar_sdns.htm as of 7/30/01.
41. Graphic: Microsoft, DNS chapter, op.cit.
42. Microsoft, Authentication chapter, op.cit.
43. RFC 2845, op.cit.
44. P. Vixie, Editor, "Dynamic Updates in the Domain Name System (DNS UPDATE)" RFC 2136 ftp://ftp.isi.edu/in-notes/rfc2136.txt as of 7/30/01.
45. Microsoft Corporation, "Windows 2000 DNS" white paper, http://www.microsoft.com/windows2000/techinfo/howitworks/communications/nameadrmgmt/w2kdns.asp as of 7/30/01.
46. Microsoft, DNS chapter, op.cit.
47. Ibid.
48. Ibid.
49. Morgan Stern, "Windows 2000 DNS Integration", http://www.lucent.com/knowledge/documentdetail/0,1494,inContentId+8844-inLocaleId+1,00.html as of 8/1/01.
50. RFC 1996, op.cit.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

51. Microsoft Corporation, "Chapter 6 – Active Directory Replication"(Replication), an excerpt from the Windows 2000 Server Distributed Systems Guide in the Resource Kit, http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/distsys/part1/dsgch06.asp? as of 8/1/01.
52. R. Elz, "Selection and Operation of Secondary DNS Servers", RFC 2182, ftp://ftp.isi.edu/in-notes/bcp/bcp16.txt as of 8/1/01.
53. Microsoft, Replication, op.cit.
54. Microsoft, DNS chapter, op.cit.
55. Windows 2000 Server online help, DNS/How To/Use Aging and Scavenging/Enable automatic scavenging of stale resource records.
56. Windows 2000 Server online help, DNS/How To/Use Aging and Scavenging/Set aging/scavenging properties for a selected zone.
57. Microsoft Corporation, "How to Convert Static DNS Records to Dynamic Records", http://support.microsoft.com/support/kb/articles/Q264/8/63.ASP, as of 8/1/01.
58. Windows 2000 server online help, DNS/How To/Optimize Servers/To Secure Server Cache Against Names Pollution.
59. Microsoft, Replication, op. cit.
60. Microsoft, DNS chapter, op. cit.
61. Microsoft Corporation, "DNS Object", excerpted from Windows 2000 Performance Counters Reference in the Windows 2000 Server Resource Kit, http://www.microsoft.com/windows2000/techinfo/reskit/en/counters/counters1_jxpr.htm as of 8/1/01.
62. Microsoft, DNS chapter, op. cit.
63. Cricket Liu, "Securing an Internet Name Server," Powerpoint presentation, http://www.microsoft.com/windows2000/techinfo/reskit/en/counters/counters1_jxpr.htm as of 8/1/01.
64. RFC 2182, op. cit.
65. DNS white paper, op.cit.
66. Microsoft, DNS chapter, op. cit.
67. Internet Software Consortium, Bind 9.1.3, http://www.isc.org/products/BIND/bind9.html as of 7/25/01.
68. D. Eastlake, "Domain Name System Security Extensions", RFC 2535, ftp://ftp.isi.edu/in-notes/rfc2535.txt
69. Ibid.
70. E. Lewis, "Notes from the State-Of-The-Technology: DNSSEC", RFC 3130, ftp://ftp.isi.edu/in-notes/rfc3130.txt
71. E. Lewis, "DNS Security Extension Clarification on Zone Status", RFC 3090, ftp://ftp.isi.edu/in-notes/rfc3090.txt
72. B. Wellington, "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, ftp://ftp.isi.edu/in-notes/rfc3007.txt
73. Bind 9.1.3, op. cit.
74. Paul B. Hill, "Kerberos interoperability issues" http://www.usenix.org/events/lisa-nt2000/hill/hill_html/ as of 8/2/01.
75. Microsoft, DNS chapter, op. cit.
76. R. Braden, Editor, "Requirements for Internet Hosts -- Application and Support", RFC 1123, http://www.ietf.org/rfc/rfc1123.txt as of 8/2/01.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

77. P. Mockapetris, "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION", RFC 1035, http://www.ietf.org/rfc/rfc1035.txt, as of 8/2/01.
78. RFC 2782, op. cit.
79. RFC 1123, op. cit.
80. RFC 2782, op. cit.
81. R. Elz et al., "Clarifications to the DNS Specification", RFC 2181, http://www.ietf.org/rfc/rfc2181.txt as of 8/2/01.
82. Tao Zhou, "Integrating UNIX DNS with Windows 2000", in Windows 2000 Magazine for February 2000, http://www.winntmag.com/Articles/Index.cfm?ArticleID=7874&pg=2 as of 8/2/01.
83. Stuart Kwan et al., "Using the UTF-8 Character Set in the Domain Name System" http://search.ietf.org/internet-drafts/draft-skwan-utf8-dns-06.txt as of 8/2/01
84. Microsoft, DNS chapter, op. cit.
85. Microsoft, DNS chapter, op. cit.
86. Morgan Stern, "Windows 2000 DNS Integration" http://www.lucent.com/knowledge/documentdetail/0,1494,inContentId+8844-inLocaleId+1,00.html as of 8/2/01.
87. Paul B. Hill on the NTBugTraq mailing list, 3.10.2000, http://packetderm.cotse.com/mailing-lists/ntbugtraq/2000/Mar/0021.html as of 8/5/01.
88. David R. Conrad on the BIND users mailing list, 12/14/99, http://www.isc.org/ml-archives/bind-users/1999/12/msg00532.html as of 8/5/01.
89. Zhou, op. cit., remove "&pg=2" at end of URL.
90. RFC 2136, op. cit.
91. All the information in this appendix is taken from command line help for the tools and from actual use of the tools.
92. RFC 2782, op. cit.
93. RFC 2930, op. cit.
94. RFC 2845, op. cit.
95. Microsoft Corporation, "DNS Object", in Windows 2000 Performance Counters reference, http://www.microsoft.com/windows2000/techinfo/reskit/en/counters/counters1_jxpr.htm as of 8/7/01.
96. Microsoft Corporation, "DNS RFCs" http://www.microsoft.com/windows2000/en/server/help/sag_DNS_add_standards.htm as of 8/7/01.
97. Microsoft, DNS chapter, op. cit.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Other Resources:

"Domain Name Service", chapter 16 from *Microsoft Windows 2000 TCP/IP Protocols and Services Technical Reference* by Thomas Lee and Joseph Davies, http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/itsolutions/network/deploy/confeat/domain.asp as of 8/7/01.

Stuart Kwan on NTBugTraq mailing list, 2/28/00, why Microsoft did not use DNSSec, http://ntbugtraq.ntadvice.com/default.asp?pid=36&sid=1&A2=ind0002&L=NTBUGTRAQ&P=R6854 as of 8/7/01.

DNS and DDNS resources at Labmice, http://www.labmice.net/networking/DNS.htm as of 8/7/01.

RIPE presentation by David Conrad at the 7/3/00 meeting, "BIND update", includes some information about GSS-TSIG interoperability, http://www.ripe.net/ripe/meetings/archive/ripe-36/presentations/bind_update/index.html as of 8/7/01.

Entire text of all Windows 2000 resource kit books can be found at http://www.microsoft.com/windows2000/techinfo/reskit/en/default.asp as of 8/7/01.

Levon Esibov, "Windows 2000 DNS", powerpoint presentation given at MIT Kerberos Workshop, http://web.mit.edu/pismere/MSR-Summer-2000/DAY1_Finished/KerberosWorkshop_W2K_DNS/default.htm as of 8/14/01.

© SANS Institute 2000 - 2002 Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.