A NGSSoftware Insight Security Research Publication

Hackproofing Lotus Domino Web Server

David Litchfield (david@nextgenss.com)
21st October 2001
www.nextgenss.com

Introduction

Brief
This document describes how to secure the web service that comes with Lotus Domino. It is written to show Lotus Domino administrators how an attacker would attempt to subvert the security of a Domino Web server and provide insight into the mind and modus operandi of a Domino hacker. The attacks are explained in detail to aid understanding and include information on how to prevent these attacks. Some of the preventative measures explained here may require upgrades or the application of security patches, whereas others simply require a modification in the Domino Directory or tightening of access control lists

What is Lotus Domino and what are Notes Databases
Lotus Domino is an Application server designed to aid workgroups and collaboration on projects. It provides services such as SMTP, POP3, IMAP, LDAP, HTTP and a Notes database server. All information is stored in Lotus Notes databases. For example when a mail is sent to a Notes user it is stored in a Notes database. When users use Domino's web service they navigate through Notes databases. These databases are not of the relational type such as Oracle or Microsoft's SQL server, but rather it is document based. Documents are grouped into views and are edited and created through the use of forms.

The logical structure of a Notes Database could be described as in Figure 1:

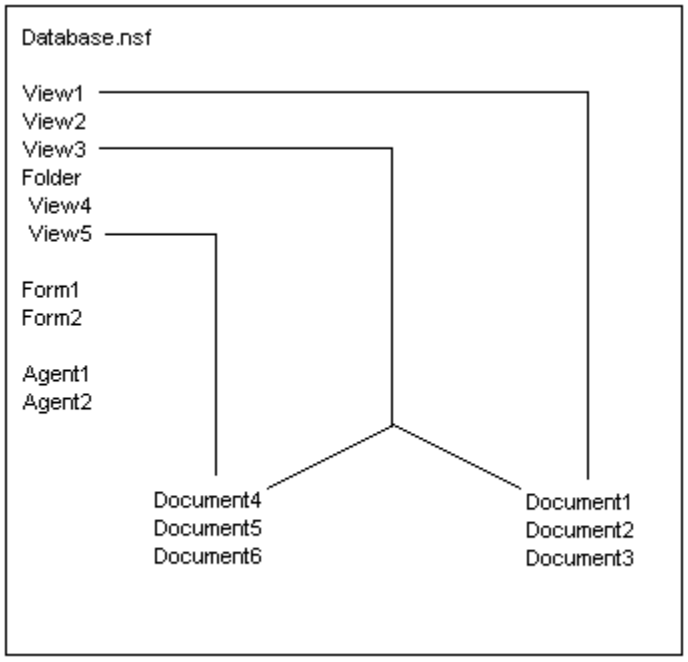

Fig1. Logical Structure of a Notes Database

On the left hand side there are views, folders, forms and agents. Folders group views together and views contain documents. As we can see in the diagram View1 can see documents 1 through 3*, View 5 can see documents 4 through 6*, whereas View3 can see all the documents. Views and folders can be considered as high level database objects and documents as low level

database objects. Forms and agents are more high level objects and are used to manipulate the low level documents.

* As we will see later on in this document this is not strictly speaking true. Just because a document is not listed as being in a view it can still be accessed through that view which opens up a security hole.

Notes databases are given the file extension .nsf and examples are names.nsf, the Domino Directory and log.nsf, the Notes Log. Notes databases can also have a .box file extension, and these usually indicate that they are mailboxes. When Domino is installed databases are stored in the lotus\domino\data directory. Non-database data such as HTML files are stored in the lotus\domino\data\domino\html directory. Further to Notes databases there are Notes template files that have a .ntf file extension and these are stored in the directory same directory as the Notes databases. (This has some relevance to the Domino web server's security discussed further on into this document.)

Notes Databases and the Domino Web Server

Users interact with a Notes database over the web using commands in the query string of a URL. For example to open a database a user would request

http://server/statrep.nsf?OpenDatabase

It is important to note that query strings can be denoted with an exclamation mark (!) in Domino. For example

http://server/statrep.nsf!OpenDatabase
is equivalent to
http://server/statrep.nsf?OpenDatabase

Anyone setting up IDS systems for Domino should be aware of this feature. Besides the OpenDatabase command there are many more: OpenServer, OpenNavigator, ReadEntries, OpenView, ReadViewEntries, OpenDocument, EditDocument, CreateDocument, DeleteDocument, SaveDocument, ReadDesign, OpenForm, ReadForm, OpenAgent, SearchView, OpenIcon, OpenAbout and OpenHelp are some of the more common ones. Some of these have a major impact of the applications security, some only if the databases Access Control Lists are set improperly. We will examine these commands and describe what they do and why they can offer an avenue of attack by a potential hacker. Throughout this document we'll be using the Statistics Reporting database, statrep.nsf, as an example. We choose this database as, by default, there are no ACLs set on this database. In other words the default access is set to manager meaning that anonymous web users have complete control over this database - in itself a security risk.

Application attacks

Commands which act on the server as a whole

The OpenServer command
Example: http://server/?OpenServer
This command will list the databases on a Domino Server. This has security implications because being able to list the databases on the system could alert an attacker to the presence of more sensitive databases on the system - for example "customers.nsf". By default database browsing is not allowed and a request for the OpenServer command will elicit a 403 Forbidden response from the server. If the command is successful and you want to disable it open names.nsf and edit the Servers document in the Server view. From the Internet Protocols tab set "Allow HTTP

Clients to browse databases" to No. This command doesn't affect a single database - it is a server-wide issue.

Commands which act on a database

The OpenNavigator command
Example: http://server/statrep.nsf/home?OpenNavigator
Notes databases can have a navigator - similar to a user interface that exists to aid users to navigate to data and documents. Every database comes with a default navigator - called $defaultNav. This navigator simply shows a list of visible views and folders. When combined with the $defaultNav, the OpenNavigator command opens a security hole as it presents a list of visible views in the database:

http://server/statrep.nsf/$defaultNav?OpenNavigator

Note that the default Navigator will not show a list of hidden views - though there are other ways to get this information - see the section on Database Structure Enumeration.

There is information freely available on how to prevent access to the default Navigator on the Notes web site (www.notes.com) - however their suggested fix doesn't work and can be easily by-passed. It describes how to create a URL to Redirection mapping (this part is right). Here is their fix:

Open the servers view and then click on the Actions menu bar item then select Web -> Create a URL Mapping/Redirection. This will open up the Mapping/Redirection form. On the Basics tab you want to set up a "Url -> Redirection" action. If the server in question is a virtual server from the site information tab enter its IP address and optionally a comment. In the mapping tab enter in the "Incoming URL path" edit box "*/*.nsf/$defaltNav*". In the "Redirection URL string" edit box enter a url where you'd have the person redirected to - for example /. You need not enter anything in the "Administration" tab. Once this has been done save and close the document and issue from the Domino console the command tell http restart for the changes to take effect.

There are two things wrong with setting the Incoming URL path to */*.nsf/$defaltNav*

Firstly, if we make a request for the database using the ReplicaID

such as http://server/8000000000000000/$defaultNav

then the pattern matching will be broken.

Secondly if we substitute any characters for their hex encoded equivalent,

for example /foo.nsf/$%64efaultNav

then, again, the pattern matching is broken.

A Domino administrator needs to create a URL redirection mapping for every possibility and when you consider /$%44efaultNav works just as well as /$%64efaultNav you have to take into case sensitivity. Because of this it would be far too impracticle to have a mapping for every variant. It is suggested therefore that only the first two characters be taken into consideration - $d.

This way only 8 mappings need to be created:

/%24D
/%24d
/%24%64
/%24%44
/$d
/$D
/$%64
/$%44

Note that if you substitute the leading slash with %2F or %5C the redirection mapping still works:

http://server/foo.nsf%2f$defaultNav
produces a 500 Unable to process request response,

whereas

http://server/foo.nsf%5C$defaultNav

performs the redirection.

NGSSoftware Insight Security Research have also tested variants of double URL encoding and UTF-8 encoding and these seem not to work - i.e. an attacker cannot get access to the default Navigator. If you have a normal database view which starts with the characters "$d" then this fix will prevent access to this view from over the web as any request that contains with "/$d" will be redirected. To work around this you could set up an alias for this view. Once the changes have been made from the Domino Server console issue the command "tell http restart" to load these changes.

Having said all this if access control lists are set properly on the database and its objects then even if someone were able to access the default navigator then this pose little risk.

The ReadEntries command
Example: http://server/statrep.nsf?ReadEntries
This command returns an XML listing of visible views. This XML listing contains information such as the views' Universal NoteID (UNID) and the views' name. This command exists to allow Java applets to work out what view are accessible for its use. Essentially this command returns to an attacker the same information as a request to the default Navigator so if you've prevent access to the default Navigator but not prevent access to ReadEntries then it was for naught. Like with the default Navigator fix a URL to Redirection mapping should be created taking into consideration all possibilities. However in this case you must use ! and ? and R but also note that this will also prevent the ReadForm, ReadViewEntires and ReadDesign commands and therefore may not be viable. It is better to secure a database and its objects using proper ACLs. Note that if clients are to access this database using Java applets preventing the use of the ReadEntries command may cause the applets to cease form working.

Commands which act on a View

The OpenView command
Example: http://server/statrep.nsf/view?OpenView
This command lists documents within a view. Normally this command does not present a problem

and is integral to the way Domino operates.

The ReadViewEntries command
Example: http://server/statrep.nsf/view?OpenView
Like the ReadEntries command the ReadViewEntries command returns an XML listing of documents within a view and is used by client-side Java applets to make use of documents in the database.

The ReadDesign command
Example: http://server/statrep.nsf/view?ReadDesign
Like the ReadEntries command the ReadDesign command returns an XML listing, but this command returns a description of the structure of the view.

Commands which act on Documents

The OpenDocument command
Example: http://server/statrep.nsf/view/doc?OpenDocument
The OpenDocument command opens a document in read mode.

The EditDocument command
Example: http://server/statrep.nsf/view/doc?EditDocument
If the ACLs on the document allow it, the EditDocument command opens a document in write mode. EditDocument returns an HTML form with fields in the document that can be edited. However, it is possible to change the other fields too. By saving the HTML page and editing the source an attacker can add the name of the field to be changed as a form input field and then submit the changes.

For example, consider a document that had two fields, "foo" and "bar". On requesting the EditDocument command the HTML form returned only allowed the editing of the "foo" field. By adding the line:

<INPUT TYPE="text" NAME="bar" VALUE="I shouldn't be able to change this but I can">

and then submitting the form both the "foo" and "bar" fields are changed. This of course only happens if the user has the permissions to edit that particular field.

The SaveDocument command
Example: http://server/statrep.nsf/view/doc?SaveDocument
If the source of the HTML form is viewed from the EditDocument command, you can see that the form action is to the SaveDocument command. Providing the ACLs allow it on submitting the EditDocument form the SaveDocument command is executed.

The DeleteDocument command
Example: http://server/statrep.nsf/view/doc?SaveDocument
As you would expect the DeleteDocument command deletes the document. If you don't want attackers stripping documents from your system it is advisable to setup ACLs correctly.

The CreateDocument command
Example: http://server/statrep.nsf/view/form?CreateDocument
Notes Forms are used to create documents in a Notes database. If you open a form over the web and HTML form is returned with an action of CreateDocument.

For all these commands that act on Documents ensure that the ACLs are set correctly to prevent unauthorized tampering.

Special Database Objects
Notes database have special database objects. For example the default Navigator, $defaultNav, and the $searchForm template are two examples. Most of these have no security ramifications though it is helpful to know they exist and how they are used.

The $icon object
Example: http://server/statrep.nsf/$icon?OpenIcon
This is the icon for the database.

The $help object
Example: http://server/statrep.nsf/$help?OpenHelp
The $help object describes how to use the database.

The $about object
Example: http://server/statrep.nsf/$about?OpenAbout
This object opens a page that tells you "about" the database.

The $icon object
Example: http://server/statrep.nsf/$icon
This is the icon for the database.

The $first object
Example: http://server/statrep.nsf/view/$first?OpenDocument
This opens the first document in a given view.

The $defaultform object
Example: http://server/statrep.nsf/$defaultform?OpenForm
This returns the first form found in the database.

Database Structure Enumeration

Enumerating Views, Forms, Agents and Special Database Objects
Objects in the database are tracked in many ways. There are Universal Note IDs (UNIDs) that are 32 character long strings. For example the following URL:

http://server/statrep.nsf/0a89ad68dd8b0187852561780077caf0/b937c60966dbb9dd80256a7400059cc6

shows a view with a UNID of 0a89ad68dd8b0187852561780077caf0 and a document with a UNID of b937c60966dbb9dd80256a7400059cc6.

Another way to reference a database object is with its NoteID which is a much smaller number. For example, the NoteID of the view above has a NoteID of 136. A NoteID is basically the postion of the object in the database file, somewhat akin to a file pointer. As far as the author can tell there is no way to get the NoteID of a view remotely. However, for all Notes databases on all servers the way NoteIDs are issued is standard as higher level objects such as forms and views reside at roughly the same location between two database files.

For any view, hidden or visible, form or agent the NoteID issued to that object is issued starting at around 0x11A and increments in 4. Try it out:
Make 10 requests

http://server/statrep.nsf/11A
http://server/statrep.nsf/11E
http://server/statrep.nsf/122
http://server/statrep.nsf/126
http://server/statrep.nsf/12A
http://server/statrep.nsf/12E
http://server/statrep.nsf/132
http://server/statrep.nsf/136
http://server/statrep.nsf/13A
http://server/statrep.nsf/13E

and so on.

If the NoteID exists the page is returned. It may be a special database object such as $icon or $help, it might be a form or it might be a view. If the NoteID doesn't exist then the server will return an error, "Invalid or nonexistent document". There is one exception to this - if the server returns an "Unknown Command Exception" error then the NoteID belongs to an Agent - you can't call an Agent with its NoteID. By going all the way up from 0x11A to 0xFFF in increments of 4 (only 953 requests) you'll have found the NoteID of every hidden view, visible view, agent, form and special object - thus the higher level database structure can be enumerated.

Enumerating Documents
Documents also have a NoteID. To get the NoteID of a document open a view with the ReadViewEntries command.

http://10.1.1.28/statrep.nsf/3.+Events/?ReadViewEntries

The NoteID of every document in this view will be returned in the XML listing. One of the peculiar things with Domino is that if you reference a document that exists in one view you can access it by requesting it through another view.

For example go to

http://server/statrep.nsf/136?ReadViewEntries

select a NoteID at random (136 is the NoteID of the "3. Events" view) and then request it like so:

http://server/statrep.nsf/136/8F6

The Event document will be returned. Now request:

http://server/statrep.nsf/$Alarms/

Normally with statrep.nsf there will be no documents in that view. However - now request

http://server/statrep.nsf/$Alarms/8F6

and the Event document is returned!

This means that it is possible to gain access to documents in one view through another and

therefore it is possible to enumerate all the documents in a database by selecting 1 view and counting upwards from 0x8F6 in increments of 4. (As far as the author can tell document NoteIDs start at 0x8F6)

http://server/statrep.nsf/$Alarms/8F6
http://server/statrep.nsf/$Alarms/8FA
http://server/statrep.nsf/$Alarms/8FE
http://server/statrep.nsf/$Alarms/902
http://server/statrep.nsf/$Alarms/906
http://server/statrep.nsf/$Alarms/90A
and onwards...

When a 500 Invalid or nonexistent message is returned all of the documents have been enumerated. If the server returns a 404 it usually means the document no longer exists - for example it has been deleted.

By-passing ACLs set on views
The fact that you can request a document in one view through another view opens up a security hole. If ACLs are set on a view to give say only admins access, then any documents in that view may be access directly through another view - thus bypassing the access control. This works because the permissions allow access to the database, the fake view and the document in question. At each stage where permission checking takes place the authorization process succeeds. If the request was made through the real view then authorization would fail. Remember by requesting a NoteID you're simply asking for the contents from a position in the database file.

To prevent against this, you must ensure that not only do you set ACLs on views but also on the documents that view is supposed to protect.

Features of Domino that lead to security vulnerabilities

Domino comes, by default with a Web Administrator database, webadmin.nsf. Anonymous users are not allowed to access this database, but they are allowed to access the template from which webadmin.nsf is derived, webadmin.ntf. Webadmin.ntf contains the same functionality as it's database child and resides in the same directory. Recalling from the introduction, database files (those with a .nsf, .ns4 and .box file extention) are served from the lotus\domino\data directory. Everything else is served from the lotus\domino\data\domino\html directory. Therefore if a user were to request

http://server/webadmin.ntf

Domino would search in the latter of these two locations, but as webadmin.ntf actually exists in the former location the server cannot find the file and returns a 404 file not found message. However, there are a couple of features of Domino that can allow a user to trick domino into searching in the right directory for webadmin.ntf. Firstly is the use of ReplicaIDs. A ReplicaID is used to maintain a database's consistency - if the database exists in two locations i.e. on two separate servers then Domino uses the ReplicaID to keep track of changes. It is completely legal to make a request to Domino for a database using its ReplicID.

For example, on the author's system the ReplicaID of names.nsf is 80256A7100183ABF and access could be had to names.nsf through any of the following requests.

http://server/80256A7100183ABF/
http://server/__80256A7100183ABF.nsf/
http://server/__80256A7100183ABF.ns4/
http://server/__80256A7100183ABF.box/

Each of the requests will cause Domino to look in the lotus\domino\data directory for a file with this ReplicaID. As names.nsf is created afresh with each new install the ReplicaID of the author's names.nsf file will probably be different to everyone else's names.nsf. However, the Web Administrator template file, webadmin.ntf is not created from scratch. It is shipped whole as part of the install and therefore the ReplicaID of the author's webadmin.ntf will be the same as everyone elses webadmin.ntf. Therefore by making a request for webadmin.ntf's ReplicaID an anonymous user can can access to the Web Administrator and use its functionality. To get the ReplicaID of the webadmin.ntf file open catalog.nsf holding Control, Shift and H down together whilst you open the catalog. (This key sequence will show all hidden views as well as visible of the database). The ReplicID can be found in the "$ReplicaID" hidden view.

Another method of tricking Domino into opening the Web Administrator template is through the use of buffer truncation. By making the following request

http://server/webadmin.ntf++++++_250_pluses+++++.nsf/

access to webadmin.ntf is granted. This works because Domino attempts to protect itself from buffer overrun attacks and chops a user request down to a safe size. In terms of events here's what happens. Domino receives the request and converts all the pluses to spaces and sees it has a .nsf file extention and therefore loads the database parser. The database parser chops the end off of the request, (thus removing the .nsf) to prevent any buffer overrun and then looks in the lotus\domino\data directory for the file, webadmin.ntf<space><space><space>.... which it finds and then opens. Thus again the attacker can use webadmin.ntf's functionality.

As far as securing against this goes the best solution is to delete the webadmin.ntf file. And probably best to remove the webamin.nsf file too. The buffer truncation issue will be fixed in Domino 5.0.9. Further to this the default permissions set on the Web Administrator template file will be modified.

General Information Leakage

Statrep.nsf Default Permissions
The default permissions on the Statistics Report database (statrep.nsf) are set to Default with Manager level access. This means that an anonymous user can, not only browse the database for information, but also edit, create and delete documents. Many of the entries under the "3. Events" view can contain sensitive information pertinent to the system setup and its failures. This information can be used by an attacker when forming further attacks. Needless to say the permissions set on this database should be modified to prevent anonymous access.

Domino Banners
By default with every page sent by a Domino web server in response to a client request a banner is embedded at the top of the HTML. This banner details the version of Domino and the operating system type it is running on - such as Windows NT on Intel. This is not desirable, as this information can aid an attacker in building attacks against the server. To prevent this information from being sent, edit the notes.ini file adding a line "DominoNoBanner=1". Note that even if you set this value the server version will still be sent in the "Server:" HTTP response - however, the

operating system is omitted. With Domino 5.0.9 due to be released in November 2001 if the DominoNoBanner is set to 1 then the "Server:" HTTP header will no longer be sent as well.

Physical Paths
In older versions of Domino if a request was made to a nonexistent executable in the /CGI-BIN the physical path was revealed. More recent versions don't exhibit this behaviour.

Common Misconfigurations

Often, too many people allow anonymous access to sensitive databases such as the Domino Directory, names.nsf. This database controls the server's configuration and also stores the user database. It would not be a good thing to allow anonymous users to download valid Notes users' userid files - with access to these an attacker will be able to masquerade as that user. Other databases that can commonly and wrongly be accessed include but is not limited to, catalog.nsf, domlog.nsf, log.nsf, statrep.nsf, bookmark.nsf and domcfg.nsf.

User Authentication
Domino supports two methods for authenticating users over the web. The first, and default method, is to send a 401 Unauthorized HTTP response back to a client if they attempt to access a resource they are not authorized to do so. This response will cause the user's web browser to pop up a window prompting for a user ID and password. With the initial 401 server response also comes a realm – the realm describes the protected resource and upon successful authentication every time the user requests anything in that realm (i.e. anything to do with that resource) their browser will automatically send with the request the user's credentials. This type of web authentication is known as Basic Authentication and the scheme used to obfuscate the user id and password is base 64 encoding. This is not encryption. A base 64 encoded character string can be easily converted into its original. Anyone with a network sniffer on the wire between the user and the Domino server will be able capture this information.

The second type of authentication used if forms based authentication. Here, when a user makes a request for a resource they are not allowed to access the server returns a 200 OK HTTP response but rather than returning the page, it returns a login form. The user would enter their ID and password in this form and submit it. If authentication is successful the server generates a cookie that is used in future requests made by the user. The server maintains a list in memory of all active sessions and uses this cookie for authorization purposes. If no activity has taken place in a given amount of time, half and hour by default then the server times out the session and the user will have to authenticate again. Like with Basic authentication the user id and password are sent over the network wire in clear text and anyone with a sniffer will be able to access this information.

With both methods it is suggested that if you are going to allow users to authenticate over the web it should be done over secure socket layer (SSL) as this will prevent anyone from with a sniffer from capturing credentials as the traffic is encrypted. As a side note, with Domino 5.x, SSL version 2 is disabled and only SSL version 3 is. SSL2 should be left disabled as SSL 2 is much more easy to break than SSL3 and if an attack is successful the attacker will be able to derive the server's private key.

A Note on Notes Access Control Lists

Many times in this document we have drawn attention to setting ACLs correctly. Before we delve into ACLs it's important to understand the difference between authentication and authorization. Authentication describes when a user passes over some credentials such as a user ID and password to be allowed access. Authorization describes what that user can do after they have been authenticated. Just because a user has be authenticated does not mean they have the

authorization to perform a certain task or access a resource.

As far as the web server is concerned the Domino server performs 4 authorization checks: firstly on the database itself, then the view, then the document and finally fields within the document. Even if a user is listed as having access to a view or document if they're not included in the database ACLs then they will not be given access and will be prompted to authenticate. Successful authorization needs to be successful with the first three of the four checks. If the user is not authorized to access a field in a document, but they are authorized on the database, view and document, then the document is still displayed, though the field they're not authorized to access will be omitted. If a user has permission to access a database and a document, but not the view the document exists in, then an illegal argument exception error is generated by the server when the user attempts to access this document. As has already been noted, though, it is possible to by-pass this view permission issue by requesting the document through another view so this is one thing to be wary of.

Conclusion

As can be seen there are many issues to tackle before a Domino web server should be exposed to the Internet and what is key to the security of the Domino server and its applications is the correct access controls being set on databases. Whilst this document contains up to date information as of October the 21st 2001 it may be out of date in a week if a new Domino vulnerability is discovered. It is suggested that Domino and Notes administrators keep abreast of current Domino security issues. The Notes website (www.notes.com) contains a wealth of security information and likewise www.dominosecurity.com. Subscribing to security mailing lists such as bugtraq (www.securityfocus.com) can help keep those who need to know in the know. NGSSoftware will also maintain a vigil and we will endeavour to keep our website (www.nextgenss.com) up to date and we have written and provide a security management tool, DominoScan, that is continually updated to help protect Domino web servers. More information about this audit tool is available from the NGSSoftware website.

A NGSSoftware Insight Security Research Publication
www.nextgenss.com