Apache/WinNT Security;

Security, security, wherefore art thou, security?

Presentation at ApacheCon/Europe 2000 track W07 on 25 October, 2000

William A. Rowe, Jr. – Apache Group Member and Developer

From Unix to WinNT; the convoluted path…

Prior to 1998, an administrator familiar with the Apache server was familiar with Unix. The Apache http server, and Apache's parent, NCSA httpd, were Unix services from the beginning. It wasn't until Nov 17, 1997, that the first Win32 announcement was broadcast to the world, heralding the arrival of a Win32 API port of the Apache server. [Interestingly, in September of that year, 49% of the sites surveyed by Netcraft were running on Apache and

its derivatives; today, in August 2000, 61% of the sites are running on Apache (http://www.netcraft.com).]

This presentation in October of 1997 would have addressed the Unix administrator, teaching them the fundamentals of the WinNT architecture and its security model. I expect that today the majority of WinNT Apache users are less familiar with Unix. But a thorough review of security would be incomplete without the background of the Unix security model, so we must start there. Many Unix concepts don't translate well to the WinNT architecture, and this is where we must focus our attention, and where Win32 developers in the Apache project focus much energy.

Apache 1.3 was a Unix server. It was 'ported', or hacked to conform within the Win32 environment. That does not mean it does not work, or that it is worse than a similar Unix port of Apache. What this implies was demonstrated this year; a bug was exposed that allowed users to peek at the folder contents of a web directory, even when an index.html file existed. The code under Unix was well tested, but Win32 introduced limits on the file name length, and the assumptions tested in two different parts of the code did not employ the same test. While this bug is corrected in Apache 1.3.13, there is always a risk (even under Unix ports) of one more bug hiding in the code to exploit.

Open Source – Open Security

This brings us to the chief security advantage of Apache, that it is open source. This means that any administrator can review the code for potential holes, and close them. It also means that hackers can review the code for exploits. Since the hacker that discovers such a hole will generally advertise their conquest, these holes can be closed as they are identified by hackers or by administrators. There is the possibility that someone knows of an Apache server exploit that they have never advertised, but any publicized hole is often patched within hours.

For this reason, among many others, always use the most recent build of Apache whenever possible. Don't, of course, delete the old version, since you may discover that the newest build introduces a problem for your specific web server configuration. Search the bugs.apache.org for fixes, and if you have a unique problem, document it as thoroughly as possible, and post the problem so it is addressed as soon as possible. If you discover a security hole, however, always email it to I-found-a-security-problem-in-the-apache-source-code@apache.org. This provides the critical time need to identify and close the security hole and publish a formal announcement.

Coming Soon; Apache 2.0…

The key underpinning of Apache 2.0 is abstraction, and the implementation is called the Apache Portability Runtime (APR) library. The key goal is eliminating all of these obscure problems from the shared code of the Apache server, and localizing them in the APR. Where the APR is well tested, the Apache server will be more predictable. Remember that Apache 2.0 is new, and it will take months before most twists of the server are attempted, and the majority of the security and performance risks are identified.

APR doesn't solve everything, nor can it. The way the Apache server flows from request to request changes based on the system's architecture. That is a separate, new Apache 2.0 feature, the Multi-Process Module (MPM). Just like mod_cgi or mod_perl, an MPM is an Apache module, with a special set of hooks and calls to create a functioning server. While Win32 could perhaps use a generic Unix MPM with the right support, it is simply not practical. The Win32 port uses its own, custom WinNT MPM. At this point, there is simply one Win32 MPM; at some point there may be several to choose between.

Just as APR introduces a new set of potential problems to explore, the WinNT MPM needs thorough testing in real world environments before we thoroughly trust this code. Unix users will not be testing it, since it is unique to the Win32 environments. Windows 95/98 takes a different code path than Windows NT/2000 through the APR and MPM, since many NT

features are not available in the consumer cousins. There is a limited group stressing the server, and that adds time until most flaws are discovered.

Evaluating the Threats

The key to any review or audit of security issues must start by defining the threats, and the mechanisms to handle the issues. The following grid covers 27 core threats posed to any web server or web application.

Threats to the Web Server: Attackers

Threats to the security of the WinNT Apache web server come from three distinct groups:

- The system administrators and internal staff
- The web site administrators and authors
- The web site users

We will not dwell on the first issue, except to point out that it is primarily avoided by using good, reliable offsite backup methodology. This method allows the compromised system to be restored, without impinging on the staff's flexibility to maintain the machine. This forum will focus on the second two categories.

Note that the second category includes 'subscribers' who are maintaining their web site on your host web server. As the owner of that third party server, you must limit the potential harm to the server on behalf of all the hosted web sites.

Threats to the Web Server: Targets

We will divide Targets for assault in the WinNT Apache web server into categories:

- Non-Apache WinNT networking services or physical access to the box
- The core services of the Apache server and loaded server modules
- CGI applications or other module-invoked applications such as ISAPI modules

We won't dwell here, either, on the first aspect. We will briefly cover basic NT security, in order to eliminate or unbind services that don't apply to the Apache web server. We will assume the box is not running basic networking services that may allow the web server to be compromised, and the physical location is secured. This will limit administrative flexibility.

If more flexibility is desired, then the appropriate resources should be consulted for security implications. The most important thing to do when dealing with the WinNT operating system (or any Unix box, for that matter) is to watch for security bulletins and apply service packs as the security holes are identified and closed.

Threats to the Web Server: Risks

The actual risks posed by a security assault fall can be grouped into three categories:

- Incapacitating or crippling the WinNT box or services
- Disclosure of private data or documents
- Modification of web data or documents, including impersonation

There is no aspect here to ignore. All three risks are tangible, and they are all ever present. The first risk can be lowered by never storing internal data and documents on a public web server; unfortunately, that may not be practical in your environment.

Understanding the Security Model

The world of Unix, a simplified security model

Early experience with http protocol was within Unix, so the security is very straightforward. World permission is required for web content, since the server runs with the minimum permissions. WinNT administrators will recognize this privilege as the Everyone group or Guest user account.

All file storage volumes are mounted under the same tree under Unix, therefore every absolute directory begins with a slash. If the webdocs volume is mounted to the root of the file system, then /webdocs is really on it's own volume. The Unix /webdocs isn't a directory name, but the volume mount point's name. If you mounted the same volume as /http, then that will be its name.

Besides obscuring volume mount points, Unix allows symbolic hardlinks. Take the example of the perl directory. Some users expect to find it at /usr/bin/perl, some at /usr/local/perl, and some expect it to be /usr/perl. Supposing it is truly at /usr/local/perl, the administrator can create a symbolic hardlink in the /usr and /usr/local directories named perl, that both refer to the directory /usr/bin/perl. Now the user won't recognize that they are in the 'wrong place', and their scripts will continue to work.

Files and directories in the Unix system have only three levels of permissions and three permissions per level. The access levels are the owner of the file, the group the file belongs to, and the world. The permissions are the right to read, or write, or execute the file. The permissions, owner and group can be changed (with appropriate permissions) using the chmod, chown and chgrp utilities. We will cover permission levels first.

Every user has a default group, but that does not mean the user cannot belong to more than a single group. Once group permissions are combined, for example, the user may belong to the webauthor and webadmin groups, but by default all files the user creates will be assigned to the webauthor group. These groups are invented by the administrator to implement whatever security they deem appropriate. Permissions of Read and Write to a file are self-explanatory to any administrator. Execute permission means the user can run the code. Read and not-execute means it is not a program, while Execute without Read means the user cannot see (or decompile) the code they are executing. There is no true analogy under with Win32 system.

Directory permissions are a bit different. Execute (X) access is required for *every* parent directory in order to have any access to a specific child directory. In order to access /docs/www/thatsite, the user must have Execute (X) access to the /., /docs and /docs/www directories. In order to list the contents, the user must have Read (R) access, and to create or delete a file, the user must have Write (W) access to the directory.

The WinNT model, a complex beast.

We don't suggest that complexity is bad, or inappropriate, in an operating system. But the big differences can cause headaches. Our first warning: do *not* use the FAT or FAT32 file system on any server, of any type. You *must* use NTFS to assure that security can be properly assigned. Permissions have no effect on FAT volumes.

All file storage volumes are top level entities. Under WinNT, you would recognize this as perhaps d:\. But they are actually members of the same namespace; you can access a volume with the syntax \\.\d:\path\file. The dot directory (.) on both Unix and Windows is 'me' (and the dot-dot directory .. is the parent directory.) Since \\ represents a machine's name, \\. means this machine. WinNT interchangeably recognizes both the '/' and '\' separator in almost all situations. Apache uses the '/' semantic throughout the server. Only Windows 2000 recognizes true hardlinks, as described in the Unix model above.

Since the file or directory does not belong to a single group (as in Unix), WinNT Access Control Lists (ACLs) allow specific permissions to multiple users or groups. It still has an

owner, but ownership permissions can be in the form of an ACL User or generic fileowner entry.

Directory permissions behave differently than Unix. Execute (X) access is required for *every* parent directory and volume in order to have any access to a specific child directory. So, in order to access D:\WebDocs\thatsite, the user must have Execute (X) access to the D:\ volume root and the D:\WebDocs directories. Under Windows NT, This can be circumvented, though, by giving the user the "Bypass Traverse Directory Checking" privilege (also referred to as Posix Compatibility). To actually list the files, or change to the directory (cd), the user must have Read (R) access to the directory.

Experimenting with User Accounts

To really understand the effects of permissions, we strongly suggest you use the SU.EXE utility provided with the Microsoft Windows NT Resource Kit. This utility allows you to start an explorer or command prompt session with another user's permissions. We will be using this utility to demonstrate and test the permissions assigned to the Apache service user account.

To invoke the SU.EXE utility, you will need to change the account permissions for your usual administrator or power user account. The SU.EXE utility requires the user that invokes the SU command to possess the "Replace a Process Level Token" and "Create a Process Token" privileges. After using the User Administration applet to add these rights to your account, you can experiment with SU.EXE.

Be warned, we are not sufficiently comfortable with the security underlying SU to support the Apache/Unix suexec model. Don't use SU from CGI scripts! It should only be used for testing and experimenting. If you have additional questions, the SU.EXE documentation is provided in the SU.TXT file installed by the Microsoft Windows NT Resource Kit.

Deploying Apache/WinNT

Creating the Service User Account

The service user account should be a local account. This account is unique for each machine running the Apache Server. It needs the following rights:

- Log on as a Service

For file and directory access, the service user account requires Execute (X) access to the root of the OS volume (where \WinNT resides), the root of the volume where Program Files reside, and to Program Files itself. However, you should not change the subdirectory permissions within the Program Files directory except on a case by case basis (such as Program Files/Apache, or Program Files/Perl.)

This account needs (RX) Read access to the C:\WinNT (or appropriate operating system path) and it's subdirectories. Be careful not to change the permissions on the C:\WinNT\profiles directory! To avoid this, select all files in the C:\WinNT folder, unselect the profiles directory, and change the permissions including subdirectories. Then change the permissions of C:\WinNT itself, without changing subdirectories.

You may wish to change a directory's permission to Execute (X) only. Add the user with 'List' access to the drive or directory, and after the user is added, double click the service user account's entry and remove the Read (R) permission. Alternately, you can assign the "Bypass Traverse Checking" option in the advanced User Rights, which will ignore permissions on the parent volume and directories when determining permissions. This may cause serious problems, however, in trying to determine the .htaccess rules.

Finally, the service user account needs access to the Apache tree itself. Read (RX) access is needed for the entire tree, and Change (RWXD) access is required for the logs and cache directories, as well as any other directories storing resources. This includes the locations of the pid file, the DAV locking file, and any other file that Apache may modify.

Don't apply the permissions to the individual web directories for the Service User Account. It won't hurt if the Apache\htdocs access is set for the service user account, but we will see next why this isn't the best account to set up the remaining web page permissions.

Creating the Apache Service Group

We will create an Apache service group as a domain wide group. Each local service user account should become a member of the Apache service group. All web files and directories are controlled through this account, so that local service user accounts on other machines can access these files and resources. This will make it simpler to experiment with the service user account, different versions of Apache, and multiple web servers sharing content from the local machine.

Running the Apache Service

After installing the Apache Service with the Apache –k install –n "ApacheService" command, you will need to make several modifications. In the Control Panel, Services window, you will need to change the startup options of "ApacheService". By selecting the "Log on as User", and using the apache service user account we created above (and entering its password), you can customize the security allowed to the Apache program.

The rest of this document follows the Apache 1.3.13 release. It prohibits you from installing the service if the httpd.conf file contains blaring errors. You may find, at this moment, that you need to define the ServerName in the httpd.conf file. This should *never* be necessary if you correctly define the machine's public IP domain name. Under Network -> Protocols -> TCP/IP -> DNS you should correctly identify this host name and domain the host name resides in. If this is the server www within joes-domain.com, then fill in www as the host name and joes-domain.com as the domain.

Securing the Server, non-Apache Considerations

WinNT offers several network clients and services. We assume you are running at least one NIC card 'outside' of the private LAN, on the Internet itself.

If you have two network cards, internal and external (or perhaps a VPN), then you will probably choose to leave most Network Services installed. Always remove all network services that are not needed, but once you are left with those that services you require, then review the Bindings and disable WINS access to the Adapter on the public LAN.

If you have a single NIC, it is not advisable to leave the WINS, Server and NetBios running. Consider using DAV, or FrontPage extensions to maintain the web server's content. One alternate strategy, if you must leave them running, is to install a firewall between the public network and the LAN. Allow only port 80 (and other ports you intend to implement, such as HTTPS on 443) inbound through the firewall.

Never trust a port scan from your local machine, it is looking at the machine through the internal redirector. Always scan the server's ports from an external machine.

The User Account

It is time to test our access to the server. Use the suexec utility, and once you have the command window (the default behavior), use the WINFILE command to browse the server.

You should discover that you do not have access to most files. You can now toggle between the WINFILE session and the windows explorer in order to set up the permissions appropriately.

Try deleting important files; you should fail. Try creating a file in the Apache/logs directory. You should succeed. Adjust the permissions as appropriate. Grant extra permissions (beyond Read access) to any web content the users will administer, as well as data

directories for web apps. And be sure you can access any Program Files directory that contain CGI scripting applications (such as perl.)

Private Data Security

This is the most important message.

- Never store private data in web document directories.

A better approach is to create a directory one level above the cgi-bin directory, called private or data. Web applications can use the '../data/' directory to store these private files, collecting commerce files and other form submission data. Any file in an Alias, ScriptAlias, or DocumentRoot directory or its subdirectories is always at risk, regardless of permissions. Don't try to mask a private directory with an Alias (Alias directories are case sensitive, so changing the case of an Alias will get at any underlying file or directory of the same name.)

Configuring the Server

Several important concepts here!

First, consistently use forward slashes, not backslashes, in file specifications! Failing to do so can be hazardous to your mental health. This could allow paths to slip by the security tests unhindered, or conversely, becomes overly protective of files you want to serve. You are also best advised to use long file names at all times. Although they are transformed within the server, using short file names (such as Progra~1) makes the server configuration hard to follow.

Prior to Apache 1.3.13, the <Directory /> section does absolutely nothing. Nada. Since the path specification such as "e:/blah/blah/blah" does not start with "/", but instead starts with "e", it simply fails to ever test <Directory />. In Apache version 1.3.13 this is resolved, so think of <Directory /> as a default permissions section from this version forward (as you probably expected it to be.) Always assure you have a very restrictive default:

```
<Directory />
   AllowOverride None
   Options FollowSymLinks
</Directory>
```

Additional 'protective' restrictions kept the server from testing <Directory />, <Directory //>, or <Directory //machine> sections, *even when they would be valid*. Apache 1.3.13 also incorporates changes to test these sections properly, although it obviously cannot test an .htaccess file for overrides in those non-directory paths.

This brings us to a critical point of confusion. No, notepad won't let you save a file by typing the file name .htaccess. But if you quote it, as ".htaccess", it will save it without argument. Just another mind-numbing aspect of Notepad (right up there with it's appending .txt to most names, again, unless they are typed in quotes).

Launching the Server

The server should be functional at this point. There is now support (in Apache 1.3.13 and later) for early errors (before log files are opened) to send messages to the Application Event Log. Looking in the Event Log viewer will help you determine the reason that the Apache Service fails to start. If the server fails to start, look in the Event log for guidance.

Note that you may be able to launch Apache from the command line, but not from the server. This is expected, since the server may not have the same permissions. You may want to review the SU.EXE utility, and try to start Apache from that console window using its own account.

You should now be successful at browsing the Apache default page, and can begin to customize the httpd.conf file for your environment.

Securing the Server

Creating the Policy – Web Site Standards

Five friends or 5,000 subscribers – regardless of the number of servers, it's important to formulate the structure for your server. We subscribe to the following model:

```
d:/webroot
          /subscriber
                    /docs
                    /logs
                    /data
```

Nothing should ever be stored outside of this tree, with the possible exception of read-only programs or script interpreters. Their virtual DocumentRoot is /webroot/subscriber/docs, so your first problem is most likely, how would they touch data or logs? FTP, with its own issues, is not a great solution. We would suggest you focus on providing shared CGI that can retrieve or receive files, appropriately named, for the respective logs and data paths.

Why protect logs? Other than revealing poor form submission practices (transmitting private data with form method=get rather than post), further compromising confidential data, it creates a large security hole. If a user seeking to get around a CGI script can review the error log for hints, it becomes a much simpler task to find a hole in the script.

One last, important note. Not every site can support this model. With 5,000 sites, the d:/webroot directory becomes a bottleneck in itself. You can start with Overrides none on this folder, but that won't cure the entire problem. You are better off breaking up the list into an additional level of directories, perhaps by the first letter of the subscriber name, or some other equally simple designation. For even larger sites, multi-level divisions such as j/a/james.com/ might be needed.

Securing Webs with Basic or Digest Authentication

Digest is far more useful than Basic Authentication, but we will discuss both. The first step is to create a password file. The same note applies here... *never* place your password or group files within the website directory tree! Always place it above the DocumentRoot and Script targets, to prevent it from being downloaded.

To create a Basic authentication file under WinNT, type the following:

```
htpasswd –c c:\apache\.users username
```

To create a Digest authentication file under WinNT, type the following:

```
htdigest –c c:\apache\.digestusers authname username
```

The system will prompt you for the password, twice. The passwords in these files are encrypted with the MD5 cipher. Be warned… this same password file may not be compatible with some systems such as Unix, so you may be unsuccessful if you transfer the file between servers.

While they look similar, these schemes are very different. Basic authentication transmits the password back to the server. It can be intercepted en route on any network the request passes through! Digest authentication, on the other hand, transmits a hash of the password, which is not so easily decoded. Given the choice, always choose Digest, but beware; many browsers in use today do not support Digest authentication.

To add users to either file, drop the –c (create) option from the commands above. To delete a user, simply edit the file in notepad, and delete that user's line.

And to create the groups authentication file, type the following:

```
notepad c:\apache\.groups
```

The group file is a plain text file; it contains lines formatted as:

```
group: username username username
```

There can be as many groups as necessary, each is followed by the complete user list. Since it is a text file, it is very simple to maintain, so take advantage of it!

Large Volume Sites using mod_auth_dbm

The basic and digest user and group authentication files described above aren't very effective at handling very large (1000+) lists of users. The alternative is mod_auth_dbm. Unfortunately, at the time of this writing, mod_auth_dbm employs Basic authentication. Keep yours eyes and ears open for (near) future dbm support for Digest (and perhaps NTLM) authentication.

The users (and optionally, groups) are stored in a DBM file. In fact, there are many DBM implementations, but Apache (and Perl) bundle the public domain SDBM flavor. Note that these flavors may not be portable from machine to machine, so a GNU GDBM database from your Unix server may not be accessible using SDBM under Windows. If you like, you can use any DBM on Windows, and relink the mod_auth_dbm to that database, rather than to SDBM, but how to do so is way beyond the scope of this discussion.

To load the mod_auth_dbm, assure that the following line is *not* commented out in httpd.conf:

```
LoadModule dbm_auth_module modules/ApacheModuleAuthDBM.dll
```

A very large site might be secured by the following directives at the root, in order to assure only the admin user has access to manage *any* web site:

```
<Location />
    Dav Off
    AuthType Basic
    AuthName Administration
    AuthDBMGroupFile c:/apache/.dbmusers
    AuthDBMDigestFile c:/apache/.dbmusers
    <LimitExcept GET HEAD POST OPTIONS>
        Require user admin
    </LimitExcept>
</Location>
```

You are already asking, why is the AuthDBMGroupFile pointing at the AuthDBMUserFile? The key field in a DBM file is always the User's name. The Value field contains the password, or it may contain

```
password:group[,group2...]:anything
```

The password is obvious, but the group or groups (seperated by commas) may follow a user's password, delimited by a colon, and anything you like may follow a second colon. Even the value password::anything is allowed, if the user belongs to no group whatsoever. You can use that anything field for whatever you need, the user's full name, a billing code, or whatever pleases you. Apache itself simply ignores it.

Management is simple. The dbmmanage.pl script would do everything you need, however Windows won't support crypt() encrypted passwords, groups or comments prior to Apache 1.3.13. The password entries must be in plain text, SHA1, or MD5 format. It's unlikely that you can use your Unix DBM users file for an Apache server under Windows NT.

Securing Webs with NTLM Authentication

One option is to integrate the NTLM Authentication scheme. This is the scheme used by Microsoft's Internet Explorer browser, and is not supported by other browsers. It is especially

useful if the web authors will be using Microsoft tools, such as FrontPage or Explorer's Web Folders (which is, in fact, a DAV client.)

You need to distinguish between NTLM protocol, or authorization scheme, and the NTLM password store. The former is a distinct mechanism, similar to Basic or Digest. The later is more akin to mod_auth_dbm, which is a password source. These modules provide one, or the other, or both.

There are several 3rd party NTLM implementations for Apache.

http://www.ozemail.com.au/~timcostello/mod_ntlm/ provides NTLM authorization against the WinNT user repository, but strictly for Apache/Win32.

http://www.runestig.com/mod_ntlm.html provides NTLM authorization against text user/group files (principally for Unix, I believe.)

To prevent confusion, mod_auth_samba (http://sourceforge.net/projects/modauthsamba/) and mod_auth_smb (http://josefine.ben.tuwien.ac.at/~mfischer/mod_auth_smb/) both provide authentication for Apache/Unix servers against an NT Domain.

DAV Security

DAV, Distributed Authoring and Versioning, is available as an Apache 1.3 Module from http://www.moddav.org/, and will be supported natively under Apache 2.0. You have already seen a DAV client if you use the later Microsoft Explorer environments… the top-level folder "Internet Folders" is a DAV client. Be warned,

If you want to administer your server using DAV, the starting point is here:

Install the DAV 1.0 binary mod_dav.dll in your modules, and assure the following lines are present in your httpd.conf file:

```
#
# Dynamic Shared Object (DSO) Support
#
LoadModule digest_auth_module modules/ApacheModuleAuthDigest.dll
LoadModule dav_module modules/mod_dav.dll

#
# mod_dav configuration
#
<IfModule mod_dav.c>
DAVLockDB "c:/apache/logs/dav_lock.db"

<Location />
    Dav Off
    AuthType Digest
    AuthName DAV
    AuthGroupFile c:/apache/.groups
    AuthDigestFile c:/apache/.digestusers
    AuthDigestDomain / http://www.server.com/
    <LimitExcept GET HEAD POST OPTIONS>
        Require user admin
    </LimitExcept>
</Location>
</IfModule>
```

This starts by assuring no webs are DAV enabled. Where a web site author requests DAV authoring, use this example to quickly enable DAV for this host:

```
<VirtualHost *>
    usual vhost options
    <Location />
```

```
        Dav On
        <LimitExcept GET HEAD POST OPTIONS>
            Require user admin vhostowner
        </LimitExcept>
    </Location>
</VirtualHost>
```

To repeat the warning; don't place the DAVLockDB, AuthDigestFile, or AuthGroupFile underneath any DocumentRoot! Also, if you will have multiple authors or administrators, consider using the Require group instead of Require user option.

Remember that using Basic authentication is a poor choice, since passwords are very easily snooped. But using digest authentication, you must create the user with the same realm as the AuthName directive. You must also modify AuthDigestDomain to include all related sites that the same users can access and administer, so they will not need to log in again.

FrontPage Security

The SERK, Front Page Server Extensions Resource Kit, is available from Microsoft directly at http://msdn.microsoft.com/workshop/languages/fp/default.asp. It consists of a set of ISAPI .dlls (on WinNT) or CGIs (on Unix) which directly manipulate the web site. They are closely tied to the server in order to maintain the web site permissions. Unfortunately, Microsoft's official support is less than ideal, since only Unix based Apache servers are supported. At the time of this writing, it is fruitless to attempt the FrontPage SERK version 2000, SR2, since it is entirely MMC (Microsoft Management Console) based, and has lost its command line interface!

We expect a to install FrontPage SERK 98 at the conference, if time and the installer permit. Note that FPSE is not a growing standard, but diminishing, based on Microsoft's own success in implementing DAV. FrontPage will undoubtedly add DAV support over time. See the "All Finished?" section below to retrieve the final notes.

suexec Behavior

If you are familiar with Apache/Unix, you have noticed suexec. At this time, there is no support for an suexec behavior under Apache/WinNT. Microsoft IIS administrators will have noticed that it's possible to use NTLM authentication to enable access with the permissions of the web site user. This is also impossible today under Apache/WinNT 1.3.13.

The suexec module allows the Apache server to handle page requests under their own owner's permission. This means that a user's CGI script will run with that script's own permissions. Plans are in the works to build on the new Apache 2.0 security model and create suexec-like behavior for virtual hosts under WinNT, so that each virtual host will run with it's own permissions. We nearly guarantee this will not be presented at the conference, unfortunately.

However, you should design the security model of your server to ultimately incorporate this very important feature. Create individual users and groups reflecting the individual hosts, and you will find the transition is very simple. For today, simply grant the server group membership in all of the virtual host groups.

Per-Virtual-Host Security

Apache 2.0 for Unix implements a Per-Virtual-Host schema that allows the administrator to run each Virtual Host in a separate process, each running under their own User and Group ID. This spells the end of the line for Apache 1.3, since such a feature is unlikely to ever be back-ported.

An implementation of the new Apache/WinNT MPM using thread-token security is under investigation, and has been incorporated in this final presentation if the alpha implementation is complete. If not, please stay tuned…

All Finished?

Apache Security is a Work In Progress. Several security deficiencies and been uncovered and corrected since ApacheCon/Europe was announced. New features were identified, and hopefully added to Apache 1.3.13. The presentation may include several topics not covered here (or covered incompletely), since Apache 1.3 and 2.0 are both evolving.

Please refer to dev.apache.org/~wrowe following the convention for updated and revised notes and the entire step-by-step and sample httpd.conf files from this presentation. Follow the apachecon.com site for additional post-conference publications.