NAME

cntlm - authenticating HTTP(S) proxy with TCP/IP tunneling and acceleration

SYNOPSIS

cntlm [**-AaBcDdFfgHhILlMPprSsTUuvw**] [*host1 port1* | *host1*:*port1*] ... *hostN portN*

DESCRIPTION

Cntlm is an NTLM/NTLM SR/NTLMv2 authenticating HTTP proxy. It stands between your applications and the corporate proxy, adding NTLM authentication on-the-fly. You can specify several "parent" proxies and Cntlm will try one after another until one works. All auth'd connections are cached and reused to achieve high efficiency. Just point your apps proxy settings at Cntlm, fill in cntlm.conf (cntlm.ini) and you're ready to do. This is useful on Windows, but essential for non-Microsoft OS's. Proxy IP addresses can be specified via CLI (*host1:port1* to *hostN:portN*) or the configuration file.

Another option is to have **cntlm** authenticate your local web connections without any parent proxies. It can work in a stand-alone mode, just like Squid or ISA. By default, all requests are forwarded to parent proxies, but the user can set a "NoProxy" list, a list of URL matching wild-card patterns, that route between direct and forward modes. **Cntlm** can also recognize when all your corporate proxies are unavailable and switch to stand-alone mode automatically (and then back again). Aside from *WWW* and *PROXY* authentication, **cntlm** provides a useful feature enabling users migrate their laptops between work and home without changing proxy settings in their applications (using **cntlm** all the time). **Cntlm** also integrates transparent TCP/IP port forwarding (tunneling). Each tunnel opens a new listening socket on local machine and and forwards all connections to the target host behind the parent proxy. Instead of these SSH-like tunnels, user can also choose a limited SOCKS5 interface.

Core **cntlm** function had been similar to the late NTLMAPS, but today, **cntlm** has evolved way beyond anything any other application of this type can offer. The feature list below speaks for itself. **Cntlm** has many security/privacy features like **NTLMv2** support and password protection - it is possible to substitute password hashes (which can be obtained using **-H**) in place of the actual password or to enter the password interactively (on start-up or via "basic" HTTP auth translation). If plaintext password is used, it is automatically hashed during the startup and all traces of it are removed from the process memory.

In addition to minimal use of system resources, **cntlm** achieves higher throughput on a given link. By caching authenticated connections, it acts as an HTTP accelerator; This way, the 5-way auth handshake for each connection is transparently eliminated, providing immediate access most of the time. **Cntlm** never caches a request/reply body in memory, in fact, no traffic is generated except for the exchange of auth headers until the client <-> server connection is fully negotiated. Only then real data transfer takes place. **Cntlm** is written in optimized C and easily achieves fifteen times faster responses than others.

An example of **cntlm** compared to NTLMAPS: **cntlm** gave avg 76 kB/s with peak CPU usage of 0.3% whereas with NTLMAPS it was avg 48 kB/s with peak CPU at 98% (Pentium M 1.8 GHz). The extreme difference in resource usage is one of many important benefits for laptop use. Peak memory consumption (several complex sites, 50 paralell connections/threads; values are in KiB):

```
   VSZ   RSS CMD
  3204  1436 ./cntlm -f -c ./cntlm.conf -P pid
411604  6264 /usr/share/ntlmaps/main.py -c /etc/ntlmaps/server.cfg
```

Inherent part of the development is profiling and memory management screening using Valgrind. The source distribution contains a file called *valgrind.txt*, where you can see the report confirming zero leaks, no access to unallocated memory, no usage of uninitialized data - all traced down to each instruction emulated in Valgrind's virtual CPU during a typical production lifetime of the proxy.

OPTIONS

Most options can be pre-set in a configuration file. Specifying an option more than once is not an error, but **cntlm** ignores all occurences except the last one. This does not apply to options like **-L**, each of which creates a new instance of some feature. **Cntlm** can be built with a hardcoded configuration file (e.g. /etc/cntlm.conf), which is always loaded, if possible. See **-c** option on how to override some or all of its settings.

Use **-h** to see available options with short description.

-A IP/mask (Allow)

Allow ACL rule. Together with **-D** (Deny) they are the two rules allowed in ACL policy. It is more usual to have this in a configuration file, but **Cntlm** follows the premise that you can do the same on the command-line as you can using the config file. When **Cntlm** receives a connection request, it decides whether to allow or deny it. All ACL rules are stored in a list in the same order as specified. **Cntlm** then walks the list and the first *IP/mask* rule that matches the request source address is applied. The *mask* can be any number from 0 to 32, where 32 is the default (that is exact IP match). This notation is also known as CIDR. If you want to match everything, use **0/0** or an asterix. ACLs on the command-line take precedence over those in the config file. In such case, you will see info about that in the log (among the list of unused options). There you can also see warnings about possibly incorrect subnet spec, that's when the *IP* part has more bits than you declare by *mask* (e.g. 10.20.30.40/24 should be 10.20.30.0/24).

-a NTLMv2 | NTLM2SR | NT | NTLM | LM (Auth)

Authentication type. NTLM(v2) comprises of one or two hashed responses, NT and LM or NTLM2SR or NTv2 and LMv2, which are computed from the password hash. Each response uses a different hashing algorithm; as new response types were invented, stronger algorithms were used. When you first install **cntlm**, find the strongest one which works for you (preferably using **-M**). Above they are listed from strongest to weakest. Very old servers or dedicated HW proxies might be unable to process anything but LM. If none of those work, see compatibility flags option **-F** or submit a Support Request.

IMPORTANT: Although NTLMv2 is not widely adopted (i.e. enforced), it is supported on all Windows since NT 4.0 SP4. That's for **a very long time**! I strongly suggest you use it to protect your credentials on-line. You should also replace plaintext **Password** options with hashed **Pass[NTLMv2|NT|LM]** equivalents. NTLMv2 is the most and possibly the only secure authentication of the NTLM family.

-B (NTLMToBasic)

This option enables "NTLM-to-basic", which allows you to use one **cntlm** for multiple users. Please note that all security of NTLM is lost this way. Basic auth uses just a simple encoding algorithm to "hide" your credentials and it is moderately easy to sniff them.

IMPORTANT: HTTP protocol obviously has means to negotiate authorization before letting you through, but TCP/IP doesn't (i.e. open port is open port). If you use NTLM-to-basic and DON'T specify some username/password in the configuration file, you are bound to loose tunneling features, because **cntlm** alone won't know your credentials.

Because NTLM identification has at least three parts (username, password, domain) and the basic authentication provides fields for only two (username, password), you have to smuggle the domain part somewhere. You can set the **Domain** config/cmd-line parameter, which will then be used for all users, who don't specify their domain as a part of the username. To do that and override the global domain setting, use this instead of plain username in the password dialog: "domain\username".

-c <filename>
Configuration file. Command-line options, if used, override its single options or are added at the top of the list for multi options (tunnels, parent proxies, etc) with the exception of ACLs, which are completely overriden. Use */dev/null* to disable any config file.

-D IP/mask (Deny)
Deny ACL rule. See option **-A** above.

-d <domain> (Domain)
The domain or workgroup of the proxy account. This value can also be specified as a part of the username with **-u**.

-F <flags> (Flags)
NTLM authentication flags. This option is rater delicate and I do not recommend to change the default built-in values unless you had no success with parent proxy auth and tried magic autodetection (**-M**) and all possible values for the **Auth** option (**-a**). Remember that each NT/LM hash combination requires different flags. This option is sort of a complete "manual override" and you'll have to deal with it yourself.

-f Run in console as a foreground job, do not fork into background. In this mode, all syslog messages will be echoed to the console (on platforms which support syslog LOG_PERROR option). Though **cntlm** is primarily designed as a classic UNIX daemon with syslogd logging, it provides detailed verbose mode without detaching from the controlling terminal; see **-v**. In any case, all error and diagnostic messages are always sent to the system logger.

-G <pattern> (ISAScannerAgent)
User-Agent matching (case insensitive) for trans-isa-scan plugin (see **-S** for explanation). Positive match identifies requests (applications) for which the plugin should be enabled without considering the size of the download (see **-S**). You can use shell wildcard characters, namely "*", "?" and "[]". If used without **-S** or **ISAScannerSize**, the *max_size_in_kb* is internally set to infinity, so the plugin will be active ONLY for selected User-Agents, regardless of download size.

-g (Gateway)
Gateway mode, **cntlm** listens on all network interfaces. Default is to bind just loopback. That way, only local processes can connect to **cntlm**. In the gateway mode though, **cntlm** listens on all interfaces and is accessible to other machines on the network. Please note that with this option the command-line order matters when specifying proxy or tunnel local (listening) ports. Those positioned before it will bind only loopback; those after will be public.
IMPORTANT: All of the above applies only to local ports for which you didn't specify any source address. If you did, **cntlm** tries to bind the given port only on the specified interface (or rather IP

address).

-H Use this option to get hashes for password-less configuration. In this mode, **cntlm** prints the results and exits. You can just copy & paste right into the config file. You ought to use this option with explicit **-u** and **-d**, because some hashes include the username and domain name in the calculation. Do see **-a** for security recommendations.

-h Display help (available options with a short description) and exit.

-I Interactive password prompt. Any password settings from the command line or config file is ignored and a password prompt is issued. Use this option only from shell.

-L [<saddr>:]<lport>:<rhost>:<rport> (Tunnel)

Tunnel definition. The syntax is the same as in OpenSSH's local forwarding (**-L**), with a new optional prefix, *saddr* - the source IP address to bind the *lport* to. **Cntlm** will listen for incomming connections on the local port *lport*, forwarding every new connection through the parent proxy to the *rhost*:*rport* (authenticating on the go). This option can be used multiple times for unlimited number of tunnels, with or without the *saddr* option. See **-g** for the details concerning local port binding when *saddr* is not used.

Please note that many corporate proxies do not allow connections to ports other than 443 (https), but if you run your target service on this port, you should be safe. Connect to HTTPS is "always" allowed, otherwise nobody would be able to browse https:// sites. In any case, first try if you can establish a connection through the tunnel, before you rely on it. This feature does the same job as tools like **corkscrew(1)**, but instead of communicating over a terminal, **cntlm** keeps it TCP/IP.

-l [<saddr>:]<lport> (Listen)

Local port for the **cntlm** proxy service. Use the number you have chosen here and the hostname of the machine running **cntlm** (possibly localhost) as proxy settings in your browser and/or the environment. Most applications (including console) support the notion of proxy to connect to other hosts. On POSIX, set the following variables to use e.g. **wget(1)** without any trouble (fill in the actual address of **cntlm**):

```
$ export ftp_proxy=http://localhost:3128
$ export http_proxy=$ftp_proxy
$ export https_proxy=$ftp_proxy
```

You can choose to run the proxy service on more than one port, in such case just use this option as many times as necessary. But unlike tunnel definition, **cntlm** fails to start if it cannot bind all of the proxy service ports. Proxy service port can also be bound selectively. Use *saddr* to pick source IP address to bind the *lport* to. This allows you, for example, to run the service on different ports for subnet A and B and make it invisible for subnet C. See **-g** for the details concerning local port binding when *saddr* is not used.

-M <testurl>

Run magic NTLM dialect detection. In this mode, **cntlm** tries some known working presets against your proxy. Probe requests are made for the specified *testurl*, with the strongest hashes going first. When finished, settings for the most secure setup are printed. Although the detection will tell you which and how to use **Auth**, **Flags** and password-hash options, you have to configure at least your credentials and proxy address first. You can use **-I** to enter your password interactively.

-N <pattern1>[,<patternN] (NoProxy)

Avoid parent proxy for these host names. All matching URL's will be proxied *directly* by **cntlm** as a stand-alone proxy. **Cntlm** supports WWW authentication in this mode, thus allowing you to access local intranet sites with corporate NTLM authentication. Hopefully, you won't need that virtualized MSIE any more. :)

-O [<saddr>:]<port_number> (SOCKS5Proxy)

Enable SOCKS5 proxy and make it listen on local port *port_number* (source IP spec is also possible, as with all options). By default, there will be no restrictions as to who can use this service. Some clients don't even support SOCKS5 authentication (e.g. almost all browsers). If you wish to enforce authentication, use **-R** or its equivalent option, **SOCKS5User**. As with port tunneling, it is up to the parent proxy whether it will allow connection to any requested host:port. This feature can be used with **tsocks(1)** to make most TCP/IP applications go thru the proxy rather than directly (only outgoing connections will work, obviously). To make apps work without DNS server, it is important that they don't resolve themselves, but using SOCKS. E.g. Firefox has this option available through URI "about:config", key name **network.proxy.socks_remote_dns**, which must be set to **true**. Proxy-unaware **tsocks**ified apps, will have to be configured using IP addresses to prevent them from DNS resolving.

-P <pidfile>

Create a PID file *pidfile* upon startup. If the specified file exists, it is truncated and overwritten. This option is intended for use with **start-stop-daemon(8)** and other servicing mechanisms. Please note that the PID file is created AFTER the process drops its privileges and forks. When the daemon finishes cleanly, the file is removed.

-p <password> (Password, PassNT, ...)

Proxy account password. **Cntlm** deletes the password from the memory, to make it invisible in /proc or with inspection tools like **ps(1)**, but the preferable way of setting password is the configuration file. To that end, you can use **Password** option (for plaintext, human readable format), or "encrypt" your password via **-H** and then use **PassNTLMv2**, **PassNT** and/or **PassLM**.

-R <username>:<password> (SOCKS5User)

If SOCKS5 proxy is enabled, this option can make it accessible only to those who have been authorized. It can be used several times, to create a whole list of accounts (allowed user:pass combinations).

-S <max_size_in_kb> (ISAScannerSize)

Enables the plugin for transparent handling of the dreaded ISA AV scanner, which returns an interactive HTTP page (displaying the scanning progress) instead of the file/data you've requested, every time it feels like scanning the contents. This presumptuous behavior breaks every automated downloader, updater and basically EVERY application relying on downloads (e.g. wget, apt-get).

The parameter *max_size_in_kb* allows you to choose maximum download size you wish to handle by the plugin (see below why you might want that). If the file size is bigger than this, **cntlm** forwards you the interactive page, effectively disabling the plugin for that download. Zero means no limit. Use **-G**/**ISAScannerAgent** to identify applications for which *max_size_in_kb* should be ignored (forcing the plugin). It works by matching User-Agent header and is necessary for e.g. wget, apt-get and yum, which would fail if the response is some HTTP page instead of requested data.

How it works: the client asks for a file, **cntlm** detects ISA's bullshit response and waits for the secret link to ISA's cache, which comes no sooner than the file is downloaded and scanned by ISA. Only then can **cntlm** make the second request for the real file and forward it along with correct headers to the client. The client doesn't timeout while waiting for it, b/c **cntlm** is periodically sending an extra "keepalive" header, but the user might get nervous not seeing the progress bar move. It's of course **purely psychological** matter, there's no difference if **cntlm** or your browser requests the scanned file - you must wait for ISA to do it's job and download then. You just expect to see some progress indicator move, which is all what the ISA's page does: it shows HTML countdown.

If the plugin cannot parse the interactive page for some reason (unknown formatting, etc.), it quits and the page is forwarded to you - it's never "lost".

The keepalive header is called `ISA-Scanner` and shows ISA's progress, e.g.:

```
HTTP/1.1 200 OK
ISA-Scanner: 1000 of 10000
ISA-Scanner: 2000 of 10000
...
```

-r "<name>: <value>" (Header)
Header substitution. Every client's request will be processed and any headers defined using **-r** or in the configuration file will be added to it. In case the header is already present, its value will be replaced.

-s Serializes all requests by not using concurrent threads for proxy (tunneling still works in parallel). This has a horrible impact on performance and is available only for debugging purposes. When used with **-v**, it yields nice sequential debug log, where requests take turns.

-T <filename>
Used in combination with **-v** to save the debug output into a trace file. It should be placed as the first parameter on the command line. To prevent data loss, it never overwrites an existing file. You have to pick a unique name or manually delete the old file.

-U <uid>
When executed as root, do the stuff that needs such permissions (read config, bind ports, etc.) and then immediately drop privileges and change to *uid*. This parameter can be either number or system username. If you use a number, both uid and gid of the process will be set to this value; if you specify a username, uid and gid will be set according to that user's uid and primary gid as defined in */etc/passwd*. You should use the latter, possibly using a dedicated **cntlm** account. As with any daemon, you are **strongly** advised to run **cntlm** under a non-privileged account.

-u <user>[@<domain>] (Username)
Proxy account/user name. Domain can be be entered as well.

-v Print debugging information. Automatically enables (**-f**).

-w <workstation> (Workstation)
Workstation NetBIOS name. Do not use full qualified domain name (FQDN) here. Just the first part. If not specified, **cntlm** tries to get the system hostname and if that fails, uses "cntlm" - it's because some proxies require this field non-empty.

CONFIGURATION

Configuration file is basically an INI file, except there are no "=" between keys and values. It comprises of whitespace delimited keyword and value pairs. Apart from that, there are sections as well, they have the usual "[section_name]" syntax. Comment begins with a hash "#" or a semicolon ";" and can be anywhere in the file. Everything after the mark up until EOL is a comment. Values can contain any characters, including whitespace. You *can* use double quotes around the value to set a string containing special characters like spaces, pound signs, etc. No escape sequences are allowed in quoted strings.

There are two types of keywords, *local* and *global*. Local options specify authentication details per domain (or location). Global keywords apply to all sections and proxies. They should be placed before all sections, but it's not necessary. They are: `Allow, Deny, Gateway, Listen, SOCKS5Proxy, SOCKS5User, NTLMToBasic, Tunnel`.

All available keywords are listed here, full descriptions are in the OPTIONS section:

Allow <IP>[/<mask>]
ACL allow rule, see **-A**.

Auth NTLMv2 | NTLM2SR | NT | NTLM | LM
Select any possible combination of NTLM hashes using a single parameter.

Deny <IP>[/<mask>]
ACL deny rule, see **-A**.

Domain <domain_name>
Proxy account domain/workgroup name.

Flags <flags>
NTLM authentication flags. See **-F** for details.

Gateway yes|no
Gateway mode. In the configuration file, order doesn't matter. Gateway mode applies the same to all tunnels.

Header <headername: value>
Header substitution. See **-r** for details and remember, no quoting.

ISAScannerAgent <pattern>
Wildcard-enabled (*, ?, []) case insensitive User-Agent string matching for the trans-isa-plugin. If you don't define **ISAScannerSize**, it is internally set to infinity, i.e. disabling the plugin for all downloads except those agent-matched ones. See **-G**.

ISAScannerSize <max_size_in_kb>
Enable trans-isa-scan plugin. See **-S** for more.

Listen [<saddr>:]<port_number>
Local port number for the **cntlm**'s proxy service. See **-l** for more.

Password <password>

Proxy account password. As with any other option, the value (password) can be enclosed in double quotes (") in case it contains special characters like spaces, pound signs, etc.

PassNTLMv2, PassNT, PassLM <password>

Hashes of the proxy account password (see **-H** and **-a**). When you want to use hashes in the config (instead of plaintext password), each **Auth** settings requires different options:

```
Settings     |  Requires
-------------+-----------------
Auth NTLMv2  |  PassNTLMv2
Auth NTLM2SR |  PassNT
Auth NT      |  PassNT
Auth NTLM    |  PassNT + PassLM
Auth LM      |  PassLM
```

Proxy <host:port>

Parent proxy, which requires authentication. The same as proxy on the command-line, can be used more than once to specify an arbitrary number of proxies. Should one proxy fail, **cntlm** automatically moves on to the next one. The connect request fails only if the whole list of proxies is scanned and (for each request) and found to be invalid. Command-line takes precedence over the configuration file.

NoProxy <pattern1>, <pattern2>, ...

Avoid parent proxy for these host names. All matching URL's will be proxied *directly* by **cntlm** as a stand-alone proxy. **Cntlm** supports WWW authentication in this mode, thus allowing you to access local intranet sites with corporate NTLM authentication. Hopefully, you won't need that virtualized MSIE any more. :) See **-N** for more.

SOCKS5Proxy [<saddr>:]<lport>

Enable SOCKS5 proxy. See **-O** for more.

SOCKS5User <username>:<password>

Create a new SOCKS5 proxy account. See **-R** for more.

NTLMToBasic yes|no

Enable/disable NTLM-to-basic authenticatoin. See **-B** for more.

Tunnel [<saddr>:]<lport>:<rhost>:<rport>

Tunnel definition. See **-L** for more.

Username

Proxy account name, without the possibility to include domain name ('at' sign is interpreted literally).

Workstation <hostname>

The hostname of your workstation.

FILES

The optional location of the configuration file is defined in the Makefile, with the default for 1) deb/rpm package, 2) traditional "make; make install" and 3) Windows installer, respectively, being:

```
1) /etc/cntlm.conf
2) /usr/local/etc/cntlm.conf
3) %PROGRAMFILES%\Cntlm\cntlm.ini
```

PORTING

Cntlm is being used on many platforms, little and big endian machines, so users should not have any problems with compilation. Nowadays, **cntlm** is a standard tool in most Linux distributions and there are various repositories for other UNIX-like systems. Personally, I release Debian Linux (deb), RedHat Linux (rpm) and Windows (exe) binaries, but most people get **cntlm** from their OS distributor.

For compilation details, see README in the source distribution. Porting to any POSIX conforming OS shouldn't be more than a matter of a Makefile rearrangement. **Cntlm** uses strictly POSIX.1-2001 interfaces with ISO C99 libc and is also compliant with SUSv3. Since version 0.33, **cntlm** supports Windows using a POSIX emulation layer called **Cygwin**.

BUGS

To report a bug, enable the debug output, save it to a file and submit on-line along with a detailed description of the problem and how to reproduce it. Visit the home page for more.

```
cntlm -T cntlmtrace.log -v -s ... the rest ...
```

AUTHOR

Written by David Kubicek <dave (o) awk.cz>
Homepage: http://cntlm.sourceforge.net/

COPYRIGHT

Cntlm uses DES, MD4, MD5 and HMAC-MD5 routines from **gnulib** and Base64 routines from **mutt(1)**.