NAME

pageant - PuTTY SSH authentication agent

SYNOPSIS

pageant (**−X** | **−T** | **−−permanent** | **−−debug**) [*key−file...*]
pageant [*key−file...*] **−−exec** *command* [*args...*]
pageant −a *key−file...*
pageant (**−d** | **−−public** | **−−public−openssh**) *key−identifier...*
pageant −D
pageant −l
pageant −−askpass *prompt*

DESCRIPTION

pageant is both an SSH authentication agent, and also a tool for communicating with an already-running agent.

When running as an SSH agent, it listens on a Unix-domain socket for connections from client processes running under your user id. Clients can load SSH private keys into the agent, or request signatures on a given message from a key already in the agent. This permits one-touch authentication by SSH client programs, if Pageant is holding a key that the server they are connecting to will accept.

pageant can also act as a client program itself, communicating with an already-running agent to add or remove keys, list the keys, or extract their public half.

The agent protocol used by **pageant** is compatible with the PuTTY tools and also with other implementations such as OpenSSH's SSH client and *ssh-agent(1).*

To run **pageant** as an agent, you must provide an option to tell it what its *lifetime* should be. Typically you would probably want Pageant to last for the duration of a login session, in which case you should use either **-X** or **-T**, depending on whether your login session is GUI or purely terminal-based respectively. For example, in your X session startup script you might write

eval $(pageant −X)

which will cause Pageant to start running, monitor the X server to notice when your session terminates (and then it will terminate too), and print on standard output some shell commands to set environment variables that client processes will need to find the running agent.

In a terminal-based login, you could do almost exactly the same thing but with **-T**:

eval $(pageant −T)

This will cause Pageant to tie its lifetime to that of your controlling terminal: when you log out, and the terminal device ceases to be associated with your session, Pageant will notice that it has no controlling terminal any more, and will terminate automatically.

In either of these modes, you can also add one or more private keys as extra command-line arguments, e.g.

eval $(pageant −T ˜/.ssh/key.ppk)

in which case Pageant will prompt for the keys’ passphrases (if any) and start the agent with those keys already loaded. Passphrase prompts will use the controlling terminal if one is available, or failing that the GUI if one of those is available. (The prompt method can be overridden with the **--gui-prompt** or **--tty-prompt** options.) If neither is available, no passphrase prompting can be done.

To use Pageant to talk to an existing agent, you can add new keys using **-a**, list the current set of keys' fingerprints and comments with **-l**, extract the full public half of any key using **--public** or **--public-openssh**, delete a key using **-d**, or delete all keys using **-D**.

LIFETIME

The following options are called *lifetime modes*. They all request Pageant to operate in agent mode; each one specifies a different method for Pageant to start up and know when to shut down.

-X Pageant will open a connection to your X display, and when that connection is lost, it will terminate. This gives it the same lifetime as your GUI login session, so in this mode it is suitable for running from a startup script such as **.xsession**. The actual agent will be a subprocess; the main Pageant process will terminate immediately, after printing environment-variable setting commands on standard output which should be installed in any process wanting to communicate with the agent.

The usual approach would be to run

eval $(pageant –X)

in an X session startup script. However, other possibilities exist, such as directing the standard output of '**pageant -X**' to a file which is then sourced by any new shell.

-T Pageant will tie its lifetime to that of the login session running on its controlling terminal, by noticing when it ceases to have a controlling terminal (which will automatically happen as a side effect of the session leader process terminating). Like **-X**, Pageant will print environment-variable commands on standard output.

--exec *command*

Pageant will run the provided command as a subprocess, preloaded with the appropriate environment variables to access the agent it starts up. When the subprocess terminates, Pageant will terminate as well.

All arguments on Pageant's command line after **--exec** will be treated as part of the command to run, even if they look like other valid Pageant options or key files.

--permanent

Pageant will fork off a subprocess to be the agent, and print environment-variable commands on standard output, like **-X** and **-T**. However, in this case, it will make no effort to limit its lifetime in any way; it will simply run permanently, unless manually killed. The environment variable **SSH_AGENT_PID**, set by the commands printed by Pageant, permits the agent process to be found for this purpose.

This option is not recommended, because any method of manually killing the agent carries the risk of the session terminating unexpectedly before it manages to happen.

--debug

Pageant will run in the foreground, without forking. It will print its environment variable setup commands on standard output, and then it will log all agent activity to standard output as well. This is useful for debugging what Pageant itself is doing, or what another process is doing to it.

CLIENT OPTIONS

The following options tell Pageant to operate in client mode, contacting an existing agent via environment variables that it should already have set.

-a *key-files*

Load the specified private key file(s), decrypt them if necessary by prompting for their passphrases (with the same choice of user interfaces as in agent mode), and add them to the already-running agent.

The private key files must be in PuTTY's **.ppk** file format.

-l List the keys currently in the running agent. Each key's fingerprint and comment string will be shown.

--public *key-identifiers*

Print the public half of each specified key, in the RFC 4716 standard format (multiple lines, starting with '**---- BEGIN SSH2 PUBLIC KEY ----**').

Each *key-identifier* can be any of the following:

- The name of a file containing the key, either the whole key (again in **.ppk** format) or just its public half.
- The key's comment string, as shown by **pageant -l**.

- Enough hex digits of the key's fingerprint to be unique among keys currently loaded into the agent.

If Pageant can uniquely identify one key by interpreting the *key-identifier* in any of these ways, it will assume that key was the one you meant. If it cannot, you will have to specify more detail.

If you find that your desired *key-identifier* string can be validly interpreted as more than one of the above *kinds* of identification, you can disambiguate by prefixing it with '**file:**', '**comment:**' or '**fp:**' to indicate that it is a filename, comment string or fingerprint prefix respectively.

--public-openssh *key-identifiers*, **-L** *key-identifiers*
Print the public half of each specified key, in the one-line format used by OpenSSH, suitable for putting in **.ssh/authorized_keys** files.

-d *key-identifiers*
Delete each specified key from the agent's memory, so that the agent will no longer serve it to clients unless it is loaded in again using **pageant -a**.

-D Delete all keys from the agent's memory, leaving it completely empty.

SSH-ASKPASS REPLACEMENT

--askpass *prompt*
With this option, **pageant** acts as an *ssh-askpass(1)* replacement, rather than performing any SSH agent functionality. This may be useful if you prefer Pageant's GUI prompt style, which minimises information leakage about your passphrase length in its visual feedback, compared to other *ssh-askpass(1)* implementations.

pageant --askpass implements the standard *ssh-askpass(1)* interface: it can be passed a prompt to display (as a single argument) and, if successful, prints the passphrase on standard output and returns a zero exit status. Typically you would use the environment variable **SSH_ASKPASS** to tell other programs to use **pageant** in this way.

OPTIONS

-v Verbose mode. When Pageant runs in agent mode, this option causes it to log all agent activity to its standard error. For example, you might run

```
eval $(pageant –X –v 2>~/.pageant.log)
```

and expect a list of all signatures requested by agent clients to build up in that log file.

The log information is the same as that produced by the **--debug** lifetime option, but **--debug** sends it to standard output (since that is the main point of debugging mode) whereas **-v** in all other lifetime modes sends the same log data to standard error (being a by-product of the program's main purpose). Using **-v** in **--debug** mode has no effect: the log still goes to standard output.

-s, **-c** Force Pageant to output its environment setup commands in the style of POSIX / Bourne shells (**-s**) or C shells (**-c**) respectively. If neither option is given, Pageant will guess based on whether the environment variable **SHELL** has a value ending in '**csh**'.

--gui-prompt, **--tty-prompt**
Force Pageant to prompt for key passphrases with a particular method (GUI or terminal) rather than trying to guess the most appropriate method as described above. (These options are relevant whenever an encrypted key filename is specified to **pageant**, and in **--askpass** mode.)

--help Print a brief summary of command-line options and terminate.

--version, **-V**
Print the version of Pageant.

-- Cause all subsequent arguments to be treated as key file names, even if they look like options.