



Anwenden des Befehls »cpau«

Für wen ist diese Anleitung gedacht bzw. für wen ist der Befehl »cpau« nützlich?

Diese Anleitung ist für diejenigen gedacht, die Befehle unter Windows in einem anderen Benutzerkontext ausführen lassen wollen. Im Klartext: Der aktuell angemeldete Benutzer soll einen Befehl unter der Flagge eines anderen Benutzers ausführen. Meistens geht es darum, dass ein Benutzer für das Ausführen eines bestimmten Befehls mehr Berechtigungen haben muss, als er normalerweise hat.

Die erfahrenen Anwender unter uns werden natürlich gleich an den Befehl "runas" denken, der ja bekanntlich zu den Bordmitteln von Windows gehört. Richtig gedacht: Mit diesem Befehl kann man genau solche "Jobs" erledigen. Doch wie steht es mit der Sicherheit, wenn ich das Passwort eines privilegierten Kontos im Klartext in meine Batchdatei schreiben muss?

Der Befehl "runas" beherrscht mittlerweile auch das Speichern eines Passwortes für das Ausführen von Befehlen (Option "/savecred"). Doch das eingegebene Passwort wird nicht in einer separaten Datei, sondern auf den jeweiligen Rechner lokal gespeichert, auf dem runas aufgerufen wurde. Wenn man nur einen Rechner zu bedienen hat, dass ist diese Option absolut ausreichend. Man gibt nur beim ersten Aufruf das Passwort ein und das System verlangt bei den Folgeaufrufen nicht mehr danach.

Allerdings wartet "runas" nicht, bis der auszuführende Befehl abgeschlossen ist, weshalb Skripte nicht mehr synchron ablaufen.

Mit dem Kommandozeilenprogramm "cpau" ist es tatsächlich möglich, den auszuführenden Job verschlüsselt in einer Jobdatei abzuspeichern. Somit bleibt das Passwort den Neugierigen verborgen. Zudem kann man über eine Option festlegen, dass "cpau" warten soll, bis der aufzurufende Befehl mit seiner Arbeit fertig ist.

Grund genug, einen Blick auf die Fähigkeiten dieses kostenlosen Werkzeugs zu werfen, oder?

Die folgenden Seiten beschreiben, wie man die Kommandozeilenparameter dieses genialen Programms so anwendet, dass auch das herauskommt, was man von ihm erwartet. Die Ideen zu den Einsatzmöglichkeiten überlassen wir den Leserinnen und Lesern dieses Artikels und wünschen viel Spaß bei der Lektüre.

Ihr TDWsoft-Team

1 Allgemeines

Je nachdem, was man mit den Berechtigungen eines anderen Benutzers ausführen möchte, sollte man sich eventuell ein paar Gedanken zum Konzept machen. Folgende Fragen könnten beim Festlegen der Vorgehensweise hilfreich sein:

- Handelt es sich immer um das gleiche Programm?
- Handelt es sich um eine Applikation oder um eine Batchdatei bzw. ein Skript?
- Erfolgt ein Remote-Zugriff oder wird nur lokal gearbeitet
- Soll der Befehl "cpau" von einem Netzlaufwerk gestartet werden, oder erfolgt der Aufruf lokal?
- Muss der Befehl "cpau" warten, bis der Befehl oder das Skript ausgeführt wurden? (Synchrone Ausführung)

Je nach Einsatzgebiet müssen nämlich die entsprechenden Kommandozeilenparameter von "cpau" gesetzt werden. Doch dazu mehr in den folgenden Abschnitten.

2 Grundregeln

Vorweg ein paar Grundregeln, die der Vermeidung einiger klassischer Fehler helfen sollen:

- × Pfade, welche Leerzeichen enthalten, sollten immer mit Hochkommata maskiert werden.
- × Falls das auszuführende Programm oder das auszuführende Skript irgendetwas lokal an der Maschine tun sollen (Dateien erzeugen, Programm ausführen etc.), dann sollte der Parameter "-profile" verwendet werden. Er sorgt dafür, dass die "lokalen Arbeiten" auch mit den Berechtigungen des angegebenen Kontos ausgeführt werden. Wird der Parameter "-profile" nicht gesetzt, erfolgen die lokalen Arbeiten mit den Berechtigungen desjenigen Benutzers, der CPAU aufgerufen hat.
- × Wenn einzelne Shell-Befehle (dir, copy usw.) ausgeführt werden sollen, muss zuerst die Shell als Präfix angegeben werden:
Also an Stelle von " dir c:\tmp " muss man "cmd /c dir c:\tmp" als Befehl angeben.
Alternativ hierzu kann man die Optionen "-c" bzw. "-k" verwenden (siehe auch 3.3)
- × Man sollte sich angewöhnen, bei Domänenkonten immer die Domäne als Präfix voranzustellen. So erreicht man eine Eindeutigkeit der Zuordnung der Konten. So könnte es z.B. ein lokales Benutzerkonto namens "Administrator" und ein domänenweit gültiges Administratorkonto geben. Die Eindeutigkeit erreicht man über den Präfix der "Domäne" (Die "Domäne" kann auch der lokale Rechner sein).
Ein lokales Konto würde demnach mit
"<Computername>\<Name des lokalen Kontos>" angesprochen werden, während mit
"<Domänenname>\< Name des Domänen-Kontos>" ein Domänenkonto gemeint ist.

3 Ausführen von CPAU

3.1 Unverschlüsseltes Ausführen

Für die ersten Experimente sollte man CPAU nur mit den notwendigsten Parametern aufrufen:

```
cpau -profile -u <Domänenname*\Benutzername> -p <Passwort> -ex "<Auszuführender Befehl>"
```

*Bezüglich der Angabe des Domänennamens siehe auch "2 Grundregeln"

Wenn man das Passwort des lokalen Administrators seines Rechners kennt, könnte man z.B. folgenden Befehl absetzen, um einen ersten (hoffentlich erfolgreichen) Aufruf von CPAU zu starten:

```
cpau -profile -u %computername%\Administrator -p <Passwort> -ex "cmd /c dir %temp% & pause"
```

Es sollte sich ein DOS-Fenster öffnen, das Sie am Ende der Befehlsausführung (von "dir") zum Drücken einer beliebigen Taste auffordert. Gibt man kein Passwort an, verlangt CPAU vor der Ausführung des angegebenen Befehls danach.

Die folgende Tabelle bringt etwas Licht in das Dunkel der eben verwendeten Optionen bzw. Befehle:

-profile	Sorgt dafür, dass auch alle lokalen Aktivitäten unter der Flagge des mit "-u" angegebenen Benutzerkontos ablaufen. Der Autor des Programms empfiehlt grundsätzlich die Anwendung dieser Option.
-u	Angabe des Benutzernamens. TIPP: Wenn man immer ein lokales Benutzerkonto angeben möchte und der Befehl auf mehreren Maschinen ausgeführt werden soll, dann empfiehlt sich die Verwendung der Umgebungsvariablen "%computername%". [Bezüglich der Angabe des Domänennamens siehe auch "2 Grundregeln"]
-p	Passwort des mit "-u" angegebenen Benutzerkontos
-ex	Auszuführender Befehl
cmd /c	Starten einer DOS-Box (Shell) zur Ausführung der nach "/c" angegebenen Shell-Befehle. An dieser Stelle kann natürlich auch der Start eines beliebigen Programmes stehen: z.B. -ex "notepad". Man sollte sich angewöhnen, den aufzurufenden Befehl grundsätzlich mit Hochkommata zu maskieren. So vermeidet man Probleme mit Leerzeichen in Pfaden bzw. mit Optionen, die zum aufgerufenen Programm gehören. TIPP: Durch Verwendung der Option "-c" kann man sich die Angabe von "cmd /c" ersparen, falls Shell-Befehle zum Einsatz kommen.
dir %temp%	Zeigt den Inhalt des Verzeichnisses an, welches in der Variablen "%temp%" hinterlegt ist.
& pause	Sorgt dafür, dass nach der Ausführung des Befehls "dir" die aufgerufenen DOS-Box geöffnet bleibt (pause). Das "&" dienen lediglich der Verkettung zweier Shell-Befehle, damit diese innerhalb einer Zeile verarbeitet werden können. TIPP: Beim Verwenden von "&&" wird der Folgebefehl nur dann ausgeführt, wenn die Ausführung des ersten Befehls erfolgreich war (also z.B. "&& pause").

3.2 Verschlüsseltes Ausführen

Nun möchte man ja nicht unbedingt das Passwort eines privilegierten Benutzerkontos im Klartext in irgendwelchen Batchdateien gewissermaßen auf dem Silbertablett präsentieren. Daher bietet CPAU die Möglichkeit, einen "Job" in einer so genannten "Jobdatei" zu verschlüsseln (siehe hierzu auch die Hinweise des Autors unter 4). Das heißt, man erstellt sich zuerst eine verschlüsselte Jobdatei, um diese dann als Option an CPAU zur Ausführung zu übergeben. Hierfür bietet CPAU das Optionspaar "-enc" [steht für "Encode"] bzw. "-file":

```
cpau -u %computername%\Administrator -p <Passwort> -ex "notepad" -enc -file myjob.job
```

Man muss nicht alle Ausführungs-Optionen wie z.B. "-profile" zur Verschlüsselung angeben. Denn in der Jobdatei werden ohnehin nur die folgenden Informationen abgespeichert (was CPAU abspeichert gibt der Befehl nach der erfolgreichen Verschlüsselung aus):

- × Benutzername
- × Passwort
- × Auszuführender Befehl

Das macht die Sache insofern flexibler, als dass man eine verschlüsselte Jobdatei mit unterschiedlichen Ausführungsoptionen aufrufen kann. Doch bevor wir uns mit weiteren Ausführungsoptionen befassen, sollten wir einen Blick auf die Ausführung einer verschlüsselten Jobdatei werfen:

```
cpau -profile -dec -file myjob.job
```

Die Kombination aus "-dec" und "-file" sorgt also für die Entschlüsselung ("Decode") der Jobdatei. Der Aufruf für das Verschlüsseln lässt sich also allgemein folgendermaßen beschreiben:

```
cpau -u <Domänenname\Benutzername> -p <Passwort> -ex "<Auszuführender Befehl>" -dec -file <Jobdatei>
```

Und die Verwendung der erzeugten Jobdatei (Dateiendung spielt keine Rolle) erfolgt dann folgendermaßen:

```
cpau -profile -dec -file <Jobdatei>
```

TIPP

Möchte man eine Befehlszeile mit Umgebungsvariablen verschlüsseln, sollte man auf die Schreibweise achten. Angenommen, man möchte die Variable "%computername%" bei der Angabe des Kontos verwenden, dann würde man den Aufruf zur Verschlüsselung folgendermaßen gestalten:

```
cpau -u %computername%\Administrator -p <Passwort> -ex "<Auszuführender Befehl>" -dec -file <Jobdatei>
```

So lange man die erzeugte Jobdatei nur an der Maschine verwendet, mit welcher man die Jobdatei erzeugt hat, gibt es keine Probleme. Versucht man jedoch die erstellte Jobdatei an einer anderen Maschine auszuführen, wird es sicherlich eine Fehlermeldung geben. Denn die Angabe von "%computername%" sorgt für die Auflösung der Variablen **vor** der Verschlüsselung. Möchte man die Auflösung der Umgebungsvariable während der Entschlüsselung erreichen, muss man folgende Notation verwenden: "%{computername}%" (das gilt natürlich für alle anderen Umgebungsvariablen auch).

3.3 Feinheiten

Neben der bereits erklärten Funktionalität bietet CPAU noch ein paar nette Optionen. So sorgt die Option "-wait" dafür, dass CPAU wartet, bis das aufgerufene Programm fertig ist. Eine ganz wichtige Sache, wenn das aufgerufene Programm z.B. irgendwelche Daten erzeugt, die nach dem Aufruf von CPAU verarbeitet werden sollen (man spricht in diesem Fall von einem "synchronen Ablauf"):

```
cpau -profile -wait -dec -file <Jobdatei>
```

Wenn man das aufgerufene Programm gänzlich vor den Augen des Anwenders verstecken möchte, kann man die Option "-hide" verwenden. Doch Vorsicht! Ruft man unter Verwendung dieser Funktion ein Windows-Programm auf, dann sieht man nicht dessen Oberfläche. Doch der Prozess existiert und er kann nur über die Kommandozeile oder mit dem TaskManager über das "Abschießen" ("kill") beendet werden.

Weitere "Feinheiten" bieten die folgenden Optionen:

-lwop	Es wird kein [Windows-]Benutzerprofil geladen, weshalb der angegebene Befehl schneller ausgeführt wird. Das funktioniert meist dann, wenn Skriptsprachen bzw. Batchbefehle zum Einsatz kommen. Manche Windows-Programme erwarten jedoch eine ordentliche Windows-Umgebung, weshalb sie mit dieser Option eventuell nicht mehr sauber funktionieren (siehe auch 4)
-k / -c	Diese Optionen ersparen Tipparbeit, indem sie den angegebenen [Shell-]Befehle den Präfix "cmd /c" (Option "-c") bzw. "cmd /k" (Option "-k") voranstellen. D.h. zur Ausführung der nach "-ex" angegebenen Befehle wird eine DOS-Box geöffnet und in der DOS-Box wird der Befehl ausgeführt.
-outprocexit	In Verbindung mit "-wait" wird die Variable "ERRORLEVEL" mit dem Ergebnis des aufgerufenen Befehls gefüllt und nicht mit dem Ausführungsergebnis von CPAU.

4 Hinweise des Autors

4.1 Allgemeine Hinweise

CPAU is an extremely popular tool. My original point in creating this tool was two fold, the ability to pass the password on the command line and to start up with network credentials versus local credentials. The first couldn't be done with RUNAS, the second required me to add an additional parameter and I was running RUNAS many times every day and was sick of typing /netonly.

The use of this tool has gone through the roof. Unfortunately most people use it to enhance permissions on the local PC for logon scripts and other things like that so they are all stuck using the -profile switch which forces a local interactive logon. Note I said, stuck. That's right, I am not changing that functionality. When in doubt, type in the -profile. If you are trying to connect to a foreign machine with the enhanced permissions and the ID you are using can't be authenticated through the trust channels you will get an authentication error and you will know you need to get network credentials. If you are working in workgroup mode, you almost ALWAYS need network credentials instead of local credentials.

Now there is one fun thing that people don't seem to get the hang of with network credentials. When you establish them, the password is NOT verified until you try to connect to something. So you can type in any password you want and it will fire up the process for you. When you go to touch the remote resource is when you will catch the error if you typed the password wrong, keep that in mind, it is important. Note that the program isn't broken, that is how it HAS to work.

Another thing that confused people is security of network drives. When you spawn a process in another security context, you lose access to your current network drives. This is a security function Microsoft has been implementing. It wasn't the case in Windows NT and I know of no way to help you get it re-enabled because you can't. You should use UNC's as much as possible for connecting to remote file shares.

One function that people kept asking for that I eventually added was the ability to encode the userid, password, and command line to be executed in an encrypted file. I have done that but instead of dealing with the massive issues in making encryption work well for everyone I have set up a proprietary encoding algorithm that seriously obfuscates the information in the encoded file. Again, this is NOT strong encryption. This tool is too cheap for me to go through the hassle of dealing with people having encryption issues. I will say that the encoding is pretty decent but I have no doubt that someone who was seriously interesting in cracking it certainly could given enough time. On the positive side there is a large use of random numbers in the encoded file and the same command with the same ID and password will not generate the same encoded file two times. This makes it much tougher to crack the file.

Along with the encoding option there is a crc option that will allow you to generate CRCs for the files and store those in the encoded file as well. If the CRC check fails, the job file will not execute.

The format of encoded file is a simple text format so if you want to copy and paste it or email it to someone, you will be able to do so without hassle.

My number one email with CPAU is about people trying to use it to run logon scripts and it not supposedly switching the context of the user to the admin context. The answer is always, use the -profile switch. My next most common email is people doing things with spaces in the paths and not properly using quotes. If something doesn't work right, use -profile and use quotes before trying to contact me. Note that if you are using job files, you specify the -profile on the command line when you decode the job file, not when you encode it.

4.2 Versionshinweise

I had to add some protection to this app. It seems people were running this with a networked drive for the current working directory. Microsoft prevents cross-security context access of network drives on purpose, this causes CPAU to not be able to fire the process up. To correct for that, if CPAU realizes your current working directory is a network drive it will change the CWD to the default local path (usually c:\windows\system32). To override this functionality you must specify the CWD option, note that if you set it to a network drive you most likely will not function properly. Also note that this is not a bug in CPAU, this is purposeful functionality from MS. You can see this out of anything that changes your local security context.

If you are using this for a logon script or something else where you need the permissions to take affect locally, you need to specify the -lwp (or -profile) switch. By default the process spawned has the current user's security context locally and the new security context remotely. Also keep in mind the note above concerning network drives, logon scripts run from network drives, you will need to set the CWD to a local machine (c:\temp maybe) and copy whatever files are necessary locally and then run cpau.

As of Version 1.08.00 I have added the ability to insert environment variables into the job file. Normally env vars get converted into their values on the machine encoding the job file, I have made it so you can escape these so they will get decoded on the machine that runs the job file. To do this, on the command line when building the job file specify the environment variable like `{%{env-var}%}` instead of like `%env-var%`. So for instance if you wanted SystemRoot you would specify `{%{SystemRoot}%}`. This only works for items that are part of the -EX parameter.

As of Version 1.08.00 I have also added additional protection around the CRC option. When you add CRC files to the job file, cpau will mark the file in such a way that no version prior to 1.08.00 will be able to use the job file. This is to prevent someone from taking a 1.08.00 or better job file with CRCs and use an older version of CPAU to avoid the CRCs.

As of Version 1.08.00 I have also added the feature to display the encoded information when creating the job file. This should help reduce the questions I am getting on why a certain job file doesn't work. Often what people specify isn't encoded in the way they think, especially around env vars. As of Version 1.11.00 the -lwp option was added which allows a local logon without loading the user's profile. This may cause odd responses in some programs. If you experience issues, use -lwp to load the user's profile to see if that works.

As of Version 1.11.00 I am specifically disallowing use from LocalSystem. This is something that works on older OS versions but doesn't work on XP SP2 and K3 and the inconsistency was causing a lot of support issues. The primary intent of this application is to allow interactive logons to switch security context for specific processes, not crutch unattended applications working for web apps and from the task scheduler.

This software is Freeware. Use it as you wish at your own risk. If you have improvement ideas, bugs, or just wish to say Hi, I receive email 24x7 and read it in a semi-regular timeframe.

You can usually find me at joe@joeware.net

4.3 Alle Parameter

Der Aufruf von CPAU ohne Parameter gibt Informationen zu den Optionen aus:

Usage:

```
CPAU -u user [-p password] -ex "WhatToRun" [switches]
```

```
user      User to log on as. Ex: user or domain\user
password  User's password
WhatToRun What to execute
```

Switches: (designated by - or /)

```
-profile   Do local logon with profile instead of net logon
-localwithprofile Alias for -profile
-lwp       Alias for -profile
-localwithoutprofile Local logon but do not load profile.
-lwop      Alias for -localwithoutprofile
-k         Prefix command with cmd /k to leave window open
-c         Prefix command with cmd /c to close window after exec.
-pipepwd   Special method allows you to pipe password in
-enc       Encrypt a job file for later user
-dec       Use an encrypted job file
-file      Specify job file to execute or create
-wait      Wait for process completion before returning.
-outprocexit Used with -wait, the errorlevel variable has the
           exit code of the spawned process instead of cpau.
-cwd x     Start at working directory x.
-hide      Start the new process in a hidden state.
-title x   Allow you specify title of command prompt windows.
-crc file[,file,file] This option allows you to encode
           CRC info for files in the job file. When decoded
           the CRC have to match or the program bombs. Note that
           it will not chase paths looking for the file, you must
           specify the exact path.
-nowarn    Don't output warning about network logon.
```

4.4 Beispiele

Ex1:

```
cpau -u joehome\joe -p logon -ex "perl cleanup.pl" -lwp
Runs perl script cleanup.pl as joehomejoe
```

Ex2:

```
cpau -u joehome\joe -p logon -ex "perl cleanup.pl" -enc -file cleanup.job
Creates job file called cleanup.job to run perl script cleanup.pl as joehomejoe
```

Ex3:

```
cpau -dec -file cleanup.job -lwp
Execute job file cleanup.job
```

Ex4:

```
cpau -u joehome\joe -p logon -ex "perl cleanup.pl" -wait -lwp
Runs perl script cleanup.pl as joehomejoe and waits for process to end
```

Ex5:

```
cpau -u joe -p logon -ex notepad.exe -lwp
Runs notepad as user joe
```

Ex6:

```
cpau -u joehome\joe -p logon -ex logonscript.cmd -lwp
Runs logon script in current directory as user joe (see note below)
```

Ex7:

```
cpau -u joehome\joe -p logon -ex logonscript.cmd -lwp -cwd c:\temp  
Runs logon script in/from c:\temp as user joe (see note below)
```

Ex8:

```
cpau -u joe -p logon -ex logonscript.cmd -enc -file logon.job -crc logonscript.cmd  
Encodes logon.job file and CRC protects the batch file
```

www.tdwssoft.com