Anleitung zu CCcam

Stand 10.10.2006

CCcam installieren

Anfangs mal was für Anfänger:
Zum basteln an der Box brauchen wir einige kleine Programme. Die sollte man sich vor dem anfangen mal heranschaffen:
1. Einen linux konformen Editor, wie z.B. ultraedit32 oder Crimson Editor (freeware)
2. Ein ftp Programm wie FlashFXP
3. Einen syslog tool für windows wie 3csyslog (freeware)
4. Ein Telnet-Tool, wie es in Windows dabei ist.

Installation der CCcam
Im wesentlichen besteht die CCcam aus 2 files, das Binary (auführbar) und die .config. In der originalen .rar Datei das die Macher anbieten heißt die Binary für die Dreambox CCcam.ppc, die .config ist für alle Versionen die selbe.
Ich empfehle die Binary in CCcam umzubenennen, das vereinfacht den Umgang später im Telnet, die Datei gehört dann nach /var/bin kopiert. Die Rechte müssen auf 755 gestellt werden, damit die ausführbar wird, das kann man im Flashfxp unter *Attributes* machen.
Die CCcam.cfg wird nach /var/etc/ kopiert.

Erster Start des Emus
Nun wäre das Emu eigentlich startbereit, doch ohne Script geht das nur von der telnet-Konsole aus. Das ist recht praktisch wenn man sehen will was das Emu macht.
Start -> ausführen -> telnet 192.168.1.10 (=IP der box)
Eingabe von user (=root) und password (=dreambox, wenn's nicht verändert wurde)
Jetzt hangeln und wir mal bis nach /var/bin (eingabe *cd /var/bin*) und sehen nach ob unsere bin auch wirklich da ist (*ls* = dir in linux) an den Farben erkennt man welche Attribute ein file hat, unser CCcam sollte grün sein.
Zum Start *CCcam -dv* eingeben (Achtung Gross- Kleinschreibung!) der Parameter ist für die Ausgabe des logs in der telnet-Konsole. So sieht man was die CC gerade macht, das sollte ungefähr so aussehen:

> **Zitat:**
> 16:09:26.209 CCcam: ==========
> 16:09:26.212 CCcam: starting CCcam 1.2.0 compiled on Jul 5 2006@21:12:46
> 16:09:26.212 CCcam: ==========
> 16:09:26.276 CCcam: online using nodeId 578103ff60952939
> 16:09:26.305 CCcam: DM70x0 detected
> 16:09:26.307 CCcam: create 2 cam device(s)
> 16:09:26.920 CCcam: provider num: fff830
> 16:09:27.120 CCcam: provider num: 021c00
> 16:09:27.215 CCcam: card added to broker with caid 500
> 16:09:28.018 CCcam: card added to broker with caid 4a70
> 16:09:28.072 CCcam: added 389 keys from /var/keys/SoftCam.Key
> 16:09:28.132 CCcam: added 541 keys from /var/keys/AutoRoll.Key
> 16:09:28.133 CCcam: static cw not found or bad
> 16:09:28.134 CCcam: read_ignorefile: cannot open /var/keys/CCcam.ignore or not found
> 16:09:28.135 CCcam: server started on port 12000

sollte hier die Eingabeaufforderung kommen ist was schiefgelaufen und CCcam ist abgeschmiert, im Normalbetrieb erscheinen hier beim zappen die Meldungen über die ecms (Keyanfragen) des Emus.
Will man genau wissen ob das Emu noch rennt, einfach ein zweites telnet Fenster öffnen und dort *ps* eingeben, da werden dann allen laufenden Prozesse angezeigt, das sieht dann ungefähr so aus:

Zitat:
root@dm7020:~# ps
PID Uid VmSize Stat Command
1 root 608 S init [2]
2 root SWN [ksoftirqd/0]
3 root SW< [events/0]
4 root SW< [khelper]
--- da hab ich einige irrilevanten Zeilen rausgeschnitten ---
599 root 2100 S /var/bin/CCcam_1.2.1
600 root 2100 S /var/bin/CCcam_1.2.1

Die Einträge im Log:
online using nodeId 578103ff60952939 bei jedem Start bekommt jeder CCcam Server/Client eine Identifikationszahl die einzigartig ist und nur diesen Node auszeichnet.

create 2 cam device(s) die 7020 hat 2 aktive Cardslots die erkannt wurden, in den nächsten Zeilen folgen die Daten der Karten

provider num: 021c00 das ist die Nummer des Providers in diesem Fall Redlight/FullX

card added to broker with caid 500 das ist der caid (Verschlüsselungsstandard) der von der aktuellen Karte unterstützt wird, in diesem falle Viaccess. Die Karte wird durch die Kombination caid/provider identifiziert, in diesem Beispiel 0500:021c00
Manche Provider vertreiben Karten mit verschiedenen caids, so ist die Redlight/FullX auch als Irdeto Version erhälltlich das ist dann die 0600:021c00.
Das caid/provider System wird später noch sehr wichtig, also gut merken.

added 389 keys es wurden erfolgreich 389 Schlüssel aus dem Keyfile eingelesen

static cw not found or bad es ist kein static.cw File vorhanden, ist nicht wieter tragisch da optional

read_ignorefile: cannot open /var/keys/CCcam.ignore or not found auch dieses file ist optional, also nicht so tragisch wenns fehlt

server started on port 12000 der Server ist läuft auf Port 12000 TCP (das ist wichtig für die Freigabe im Router/Firewall)

Keyfiles
Die Keyfiles kommen in das /var/keys Verzeichnis (wenn man das in der config nicht ändert). Die CCcam benutzt das weit verbreitete SoftCam.key/Autoroll.key Format, das auch von anderen Emus benutzt wird, somit ist man nicht an ein bestimmtes Keyformat gebunden. Keyfiles sind optional, das Emu funktioniert auch ohne.

CCcam und Cardsharing

Nun verbinden wir die CCcams. Dazu muss die /etc/CCcam.cfg editiert werden (nicht Windows Notepad benutzen, siehe oben). Die Standard Config dient gleichzeitig als Anleitung. Alles was mit # beginnt wird con dem emu ignoriert und ist nur zum lesen da.
Die Reihenfolge der einzelnen Lines ist irrelevant, beim Start lädt des Emu die Config und interpretiert die Lines nach ihrem Anfangsbuchstaben.

F: Friends

> **Zitat:**
> F: user1 pass1 1

Das ist eine F (friends) Line, die kommt auf die Serverbox, da definiert man Benutzer die auf die eigene Box zugreifen dürfen, die Zahl nach user und pass definiert wie weit der user1 über meine Serverbox hinaus auf die Boxen meiner freunde die auch an mit hängen sehen kann. Im Beispiel oben kann der user1 nur meine karten sehen. Mit einer 2 dahinter, meine und die aller Freunde die direkt an meiner box hängen, mit 3 die Karten der Freunde meiner Freunde usw.... das nennt man Cascading, da kann schnell mal eine ganze Menge Karten zusammenkommen.

C: Connect

> **Zitat:**
> C: server.dyndns.org 12000 user1 pass1

Das ist eine C (connect) Line, damit wird die CCcam auf der Clientbox an eine Serverbox connected. Die URL oder IP gleich nach dem C: identifiziert die Serverbox im Netz, 12000 ist der Port auf dem der Server arbeitet und user1 pass1 identifizieren den User.
Wichtig: Ein User wird auf dem Server nur einmal akzeptiert, er kann sich gleichzeitig nicht mehrmals verbinden!

Will man den Client nur an den Server hängen genügt es F: am Server zu machen und C: am Client. Möchte man die Karten aus dem Clientbox auch auf der Serverbox sehen geht das zusätzlich nochmals umgekehrt. Also hat dann jetde Box eine C: und eine F: Line.

Diagnose im Log
Hat man das alles gemacht kann man die beiden CCcams aus dem Telnet mit CCcam -dv starten, und sehen ob eine Verbindung zustande kommt. Anfangs sollte man das im Telnet versuchen, das Skript für BP kommt dann wenn alles rennt wie es soll.
So sieht das Log im Telnet zwischen den beiden Testboxen im Haus aus:

> **Zitat:**
> 11:55:06.977 CCcam: found betacrypt caid: 0x1702 ecmpid: 0x100a id: 0x0
> 11:55:06.978 CCcam: found betacrypt caid: 0x1722 ecmpid: 0x100a id: 0x0
> 11:55:06.978 CCcam: found nagra caid: 0x1801 ecmpid: 0x1642 id: 0x0
> 11:55:06.978 CCcam: cam[0] set PMT for sid=a
> 11:55:06.979 CCcam: start EMM
> 11:55:06.996 CCcam: cam[0] ecm even **nok** caid:0x1702 id:0x0 pid:0x100a Premiere Sat (19E) tunneled Nagra (took 0.0010 seconds)
> 11:55:07.061 CCcam: cam[0] ecm even **nok** caid:0x1722 id:0x0 pid:0x100a Premiere Kabel (19E) tunneled Nagra (took 0.0003 seconds)

In den ersten drei Zeilen wird angezeigt wie der Kanal verschlüsselt ist. Pre***e kommt gleich mit 3 möglichen caids an:
1801 reines Nagra (wird zur Zeit nicht oder nur mit s04 Karten benutzt) = irrelevant
1702 das ist tunneled Nagra = betacrypt für Satellit
1722 das ist tunneled Nagra = betacrypt für Kabel

die Provider id ist bei Pre***e, id:0x0, in gewohnter Schreibweise mit 6 Ziffern (einfach 0x vorne dran weglassen und von links mit 0 füllen) 000000

die pid ist der Programm identifier ist in diesem Fall 100a (ohne die 0x) das ist Pre***e 1
mit start EMM wird die Schlüsselsuche gestartet.

die nächsten beiden Zeilen beinhalten die Antworten von lokal. Die Antworten sind negativ (!) ... ecm even nok caid:0x1722 ... da in dieser Box keine Karte steckt. Die Anfrage hat 0.0003s gedauert.

Dann folgen die Anfragen an die Serverbox:

> **Zitat:**
> 11:55:07.194 CCcam: remote ecm -> 192.168.1.3:12000 0x1702(0x000)
> 11:55:07.331 CCcam: remote ecm <- 192.168.1.3:12000 **ok** (took 0.1364 econds)
> 11:55:07.333 CCcam: cam[0] ecm even **ok** caid:0x1702 id:0x0 pid:0x100a Premiere Sat (19E) tunneled Nagra (took 0.1383 seconds)

diese Anfrage war erfolgreich, ...ecm even ok caid:0x1702... und hat im lokalen LAN 0.1383s gebraucht

Anhalten des Servers
Solange man sich in der Telnet Konsole befindet in der das Log läuft genügt Ctrl+c .
Sonst muss von telnet aus mit *killall CCcam* terminiert werden.

Status des Emus

Der integrierte Webserver
CCcam bietet einen Webserver aus den man den aktuellen Status es Emus und dessen Verbindungen abfragen kann. Mit Standardeinstellungen läuft der Webserver auf Port 16001. Beim Zugriff mit dem Browser nicht vergessen das Protokoll *http://ip_der_box:16001* hinzu zuschreiben.
Im Webserver könne Active Clients, Clients, der Status der Server usw. angezeigt werden.
In unserem Beispiel müsste ein Eintrag bei Clients und einer bei Servers sein. Ist der Eintrag vorhanden, steht die Verbindung und das Bild muss hell werden.

So sollte die Active Clients Seite aussehen, im ersten Teil werden die User aufgelistet die in den letzten 20 Sekunden aktiv waren und wie viele ECM sie gezogen haben.
Im zweiten Teil werden alle User und deren Traffic aufgelistet, es wird sogar unterschieden welcher Traffic lokal aufgelöst werden konnte und welchen remote abgefragt werden musst

code:
```
   Connected clients: 4
   2 ACTIVE CLIENTS IN LAST 20 SECONDS
   +---------+--------------+-----------+-----------+---------+-----+-----
 1:---+
 2:| Username| Host         | Connected  | Idle time  | ECM     | EMM |
 3:Version|
 4:+---------+--------------+-----------+-----------+---------+-----+-----
 5:---+
 6:|User1    |XX.XX.10.1     |00d 01:19:32|00d 00:00:01|518 (509)|0 (0)|1.2.1
 7:|
 8:|User2    |XX.XX.10.2     |00d 00:46:25|00d 00:00:10|174 (167)|0 (0)|1.2.0
 9:|
10:+---------+--------------+-----------+-----------+---------+-----+-----
11:---+
12:
13:+---------+-------------------------------------+
14:| Username| Shareinfo                           |
15:+---------+-------------------------------------+
16:|User3    |local 500:021500 1(0)                |
17:|User4    |remote 919:000000 280(280)           |
   |         |remote 1801:000501 2(0)              |
   +---------+-------------------------------------+
```

So sollte die Server Seite aussehen, ganz rechts findet man die CAIDs der server mit der Anzahl der angefragten ECMs. Auch hier werden lokale und remote Karten unterschieden

code:

```
1 Server connections: 1
 :+-----------------+------------+---------+--------+---------------+-------
2 ------+----------------------------+
 :| Host            | Connected  | Type    | Version| NodeID         | Nr Of
3 Cards | CAID/Idents                |
 :+-----------------+------------+---------+--------+---------------+-------
4 ------+----------------------------+
 :|192.168.1.4:12000|00d 15:15:58|CCcam-s2s|1.2.1   |5d341729df599d5a|2
5 |remote 919:000000 60(46)     |
 :|                 |            |         |        |                |
6 |local  4a70:000000 67(0)     |
 :+-----------------+------------+---------+--------+---------------+-------
7 ------+----------------------------+
 :
```

Loggen mit Syslog

Wer schon etwas mit dem Log in der Telnet Konsole gearbeitet hat, hat sicher erkannt wie hilfreich das ist, wenn man dem Emu beim arbeiten zusehen kann um Fehler zu suchen.
Um nicht jedes mal das Emu im -dv Modus in der Konsole starten zu müssen, um zu loggen sollte man sich das 3cSyslog (siehe Link in der Einleitung des ersten Teils) installieren.
In der Config diese beiden Parameter bei Bedarf auf yes stellen:

> **Zitat:**
> # if timing should be shown in OSD and debug output
> # default is no (turned off)
> #
> SHOW TIMING : yes
>
> # turns debugging on and off
> # default is no (turned off)
> #
> DEBUG : yes

Startet man jetzt mit CCcam mit dem parameter -V schreibt das Emu das Log in den Syslog Daemon der Box.
Um das Syslog der Box in der Software am PC zu sehen muss dessen Ausgabe aktiviert werden, in Gemini findet man den Punkt Sys/Kernel Log im Blue Panel unter *(3) Extra/Einstallungen -> Sys/Kernel Log*.
Hier muss der Syslog-Daemon aktiviert werden, dann ganz unten *remotes logging aktivieren* und den Host = IP des PCs mit 3cSyslog angeben, der Port bleibt auf 514 (UDP). Dienst Restarten.
Nun sollte man am PC das Log live verfolgen können.

Das 3cSyslog funktioniert auch für andere Emus, wie Camd3 oder Newcs, da wird in die Config einfach die IP und der Port (514) eingetragen. Der Daemon auf der Box ist für die beiden nicht nötig.

<u>Verbindung mit fremden Emus</u>

Camd3

CCcam kann camd3 Shares lesen, allerdings nicht als Server dafür dienen. Es funktioniert allerdings nur für UDP shares (die erkennt man in der camd3.servers an der 357 vorne dran).
Die L: Line verbindet CCcam mit einem camd3 Server, dazu sind folgende Daten nötig: IP oder Url des Servers, Server Port, User und Pass und caid/provider

Zitat:
L: server.dyndns.org 567 user pass 0100 000080

Hier kommen die caid/provider Zahlen wieder ins Spiel (siehe erster Teil), die eingegeben werden müssen, da camd3 von sich aus nicht verrät welche Karten es anbietet.
Hat der camd3 Server mehrere Karten im Angebot muss für jede eine eigene L: Line geschrieben werden.

Radegast

Die R: Line verbindet CCcam mit einem radegast server, dazu sind folgende Daten nötig: IP oder Url des Servers, Server Port, User und Pass und caid/provider

Zitat:
R: 127.0.0.1 678 0100 000080

Auch hier muss wie bei camd3 pro Karte eine eigene R: Line geschrieben werden

Newcamd (Newcs)
Die Verbindung zu einem Newcamd Server ist da schon etwas einfacher, da Newcamd beim Login dem Client mitteilt was er anbietet.
Die N: Line verbindet CCcam mit einem Newcamd Server (z.B. NewCS oder Newcamd), dazu sind folgende Daten nötig: IP oder Url des Servers, Server Port, User und Pass und der DesKey der für die Verschlüsselung der Kommunikation benutzt wird.

Zitat:
N: 127.0.0.1 10000 dummy dummy 01 02 03 04 05 06 07 08 09 10 11 12 13 14

Da CCcam momentan (V 1.2.1) noch nicht alle Karten lesen kann, bedient man sich für "besondere" fälle eines Cardreaders. Einer der besten Cardreader ist sicherlich NewCS der als Server im Newcamd Protokoll arbeitet.
Eine Besonderheit des Newcamd Protokolls ist das Login, bei dem sich Client und Server bidirektional Daten austauschen, so wird dem Server z.B. mitgeteilt welches Emu der Client benutzt, möchte man sich am Server nicht als CCcam ausgeben, so kann man den Stealth Modus benutzen:

Zitat:
NEWCAMD STEALTH : yes

und sich als Mgcamd ausgeben.
Hat man bereits ein aufwendiges Newcamdsetup auf der Box am laufen, so kann man das einfach mit

Zitat:
NEWCAMD CONF : yes

auf Servereinträge parsen lassen ohne alles nochmals einzugeben.
Bei der aktuellen Version funktioniert (zumindest bei mir) die Unterstützung der CCcam für externe Cardreader nicht, mit NewCS gehts anstandslos.

Konfiguration von NewCS
Eigentlich hat NewCS gar nichts mit CCcam zu tun, ist aber Voraussetzung um Karten zu lesen die das Emu (noch) nicht alleine schafft, z.B. Nagra 2. Aus diesem Grund möchte ich hier mal etwas näher darauf eingehen.
Newcs besteht wie CCcam aus 2 Files, die ausführbare newcs (binary) die in /var/bin/newcs steht und rechte 755 braucht und das Konfigurationsfile das in /var/tuxbox/config/newcs.xml steht und im Xml Format aufgebaut ist.
Xml ist wie Html aufgebaut, es besteht aus Tags, die geöffnet und auch wieder geschlossen werden, die Tags sind verschachtelt und beinhalten die Infos für den Cardserver.

Hier ein kleiner Ausschnitt aus der newcs.xml

Zitat:
<device>
<name>lower</name>
<type>Sci</type>
<node>/dev/sci0</node>
<parity>even</parity>
<export>yes</export>
<enabled>yes</enabled>
<blocksa>no</blocksa>
<blockua>no</blockua>
<blockga>no</blockga>
<boxid></boxid>
<PTShandshake>no</PTShandshake>
<Seca-PPV>no</Seca-PPV>
<crypto-special>no</crypto-special>
<carddetect>no</carddetect>
<newcamd_port>34000</newcamd_port>
<autosid>yes</autosid>
<priority>round</priority>
</device>

In diesem Teil steht die Konfiguration des unteren Slots (/dev/sci0) der Dreambox. Ich möchte hier jetzt nicht näher auf die einzelnen Tags eingehen, wichtig für die Kommunikation mit der CCcam ist der unterstrichene Teil, hier wird der Port dieses Ports festgelegt, jeder Port oder Kartenleser braucht einen eigenen Port. Das ist der Port der in der CCcam.cfg in der N: Line eingetragen werden muss.

Zitat:
<newcamdserver>
<enabled>yes</enabled>
<deskey>01 02 03 04 05 06 07 08 09 10 11 12 13 14</deskey>
<name>newcs</name>
<user>
<name>dummy</name>
<password>dummy</password>
<au>on</au>
<allow>lower</allow>
<allow>upper</allow>
</user>
</newcamdserver>

In diesem Abschnitt wird der Newcamdserver erstmals aktiviert und der Deskey (zur Verschlüsselung der Kommunikation) festgelegt, auch dieser kommt in der CCcam.cfg vor.
Das letzte was für die N: Line noch fehlt ist user/pass, das wird im nächsten Subtag festgelegt (dummy/dummy), weiters wird hier noch eingestellt ob der User AU (Autoupdate) machen kann und welche oben definierte devices er benutzen darf.
Unsere N: Line sollte dann so aussehen, 127.0.0.1 ist der lokalhost, das der newcs lokal ist:

> **Zitat:**
> N: 127.0.0.1 34000 dummy dummy 01 02 03 04 05 06 07 08 09 10 11 12 13 14

Wenn man Newcs als Cardserver benutzt muss dieser natürlich vor CCcam gestartet werden. Ich empfehle dazu 2 getrennte Telnetfernster zu öffnen, eines für Newcs und das andere für CCcam.

Fortgeschrittene Einstellungen

Friends (Clients) und Kaskadierung

Wie schon kurz im 1. Teil angeschnitten kann CCcam hervorragend kaskadieren.
Wir haben in unserem Testaufbau 4 Boxen, die 1er ist mit der 2er verbunden, die 2er mit der 3er und die 3er mit der 4er. das sind die einzigen Verbindungen der Boxen, die 1er ist also nicht direkt mit der 4er verbunden.
Durch die Kaskadierung kann die box1 auch auf die box4 zugreifen.

box1 <----> box2 <----> box3 <----> box4
^----1 hop-----^
^----------2 hops----------^
^--------------------3 hops--------------------^

Holt sich die box1 die Keys von box4 (diesen Vorgang nennt man ecm) muss die Anfrage und die Anwort über 3 boxen springen, das nennt man hops.

> **Zitat:**
> F: user1 pass1 **2**

Dieser User kann alle lokalen Karten des Servers sehen und noch 2 hops weiter.
Täte man also diesen User auf der box2 anlegen, könnte box1 die karten von box2, box3 und box4 sehen.

> **Zitat:**
> F: user1 pass1 **1**

So könnte box1 nur box2 und einen hop weiter also box3 sehen.

Lesen von lokalen Keyfiles und remote EMM

Das ist die einfachste Einstellung für einen User, es geht auch etwas aufwendiger

> **Zitat:**
> F: user2 pass2 0 1 0

die 3 Zahlen hinter dem user/pass haben folgenden Sinn:
Die erste ist die uphops, wie oben beschrieben
Die zweite gibt an ob der User Zugriff auf die lokalen Keyfiles haben sollte

Das ist eine sehr praktische Option, wenn man mehrere Boxen Zuhause hat und nicht auf allen die Sofcam.keys pflegen möchte. Man aktualisiert nur die Serverbox und greift dann vom Client aus mit folgender C: line zu, wichtig ist das *yes* am Ende

> **Zitat:**
> C: 192.168.1.2 12000 user3 pass3 **yes**

Default ist die zweite Zahl auf 1, d.h. wenn man nur einen einfachen User definiert hat (mit nur einer Zahl dahinter) nimmt die CCcam an dass der User auf die lokalen Keyfiles Zugreifen darf. Möchte man das verhindern muss da eine 0 rein.

Die dritte Zahl ist für remote emm's zuständig. Steht hier eine 1 darf der User emm auf der Karte machen. Beispiel: Es gibt einen Keywechsel, die neuen Keys werden mit dem TV Signal ausgestrahlt. Damit die Karte aktualisiert wird, müsste die ServerBox angemacht und auf einen gewissen Kanal gestellt werden. Wenn der User emm zugelassen ist, genügt es dass dieser die Clientbox auf den gewissen Kanal stellt und so die Karte updatet, er spielt gewissermaßen das Update remote auf, das nennt man Remm (remote emm)
Default ist auch hier 1, wird also ein einfacher User mit nur einer Zahl hinten dran definiert darf der remm machen.

Nochmals kurz Zusammengefasst:

> **Zitat:**
> F: user1 pass1 3 0 0

Dieser User sieht die Karten vom Server und 3 hops dahinter, hat keinen Zugriff auf die Keyfiles am Server und darf auch keine emm machen (remm)

Rechte für einzelne Karten UP- und DOWNshare
So, nun wird noch etwas komplizierter, es ist möglich für jede freigegebene Karte einzeln folgende Einstellungen zu definieren:

> **Zitat:**
> F: user2 F: user2 pass2 0 1 0 { 0100:000080, 0622:000000:1, 0500:000000:2 }

dieser User bekommt (wegen der 0 gleich hinter dem pass (...pass2 **0** 1 0 ...) nur die lokalen Karten des Servers zu sehen (im Beispiel sind es 3 Karten)
er darf auf die lokalen Keyfiles zugreifen (...pass2 0 **1** 0 ...)
er darf keine emms machen (...pass2 0 1 **0** ...)
er hat keinen Zugriff auf die Karte 0100:000080, weil dahinter nichts steht (0100:000080**,**)
er hat Zugriff auf 0622:000000, aber nur für sich selbst = 1 downhop (0622:000000**:1**)
er hat Zugriff auf 0500:000000 und darf diese einen hop weitergeben = isng. 2 downhops (0500:000000**:2**)

Fazit, man kann nicht nur einstellen wie weit der Client hinter die Serverbox (uphops) sehen kann, sonder auch wie weit der Client diese keys dann selbst weitersharen kann (downhops).

Noch ein Beispiel:

> **Zitat:**
> F: user3 pass3 5 0 1 { 0:0:3, 0100:000080:1 }

Dieser User bekommt (wegen der 5 gleich hinter dem pass (...pass2 **5** 0 1 ...) die lokalen Karten der Serverbox und alle weitern Karten zu sehen die bis zu 5 uphops hinter der Serverbox stehen.
der user bekommt keine lokalen Keyfiles geshart und darf remm auf den Karten am Server machen.
In den Klammern stehen hier 0 für Caid und 0 für services, das sind Jokers, es bedeutet alle caids und alls services, die 3 danach ist die downshare Tiefe.
Also kann dieser User alle Karten die er von der Serverbox bekommt 3 hops downsharen, ausser die 0100:000080, die explizit für einen hop zugelassen ist (...0000:**1**...), diese kann er somit nur selbst sehen.

Das hier ist zusammenfassend nochmals der mögliche Inhalt der C: Line

> **Zitat:**
> F: <username> <password> <uphops> <shareemus> <allowemm> ({ caid:id(:downhops), caid:id(:downhops),... })

Start- und Stopskripte

Nachdem unser Emu bis jetzt immer im Telnet gestartet wurde damit wir das Log beobachten können, sehen wir uns nun mal die Skripte an, die es uns erlauben von der Fernbedienung aus ein Emu zu starten bzw. stoppen

Der Pfad zu den Skripts ist in jedem Image verschieden.

Gemini
Im Gemini Image sind alle Skripts in */var/script* untergebracht. Skripts die ein Emu betreffen und im BluePanel angezeigt werden sollen, enden mit *_cam* (z.B. CCcam_cam.sh).
ACHTUNG: da Skripts ausführbar sind müssen die Attribute auf 755 stehen.

Die originalen Gemini Skripts beginnen mit einer Liste von CAMID's. Jedem Emu wird ein Bereich zugewiesen. Diese Auflistung ist nur zu Orientierung gedacht und kann vernachlässigt (gelöscht) werden.

Die aktuelle CAMID wird weiter unten im Skript festgelegt:

> **Zitat:**
> CAMID=6000

Die CAMID legt die angezeigte Reihenfolge der Emus im BluePanel fest. Jede CAMID darf nur einmal vorkommen, da Gemini sich das aktuelle Emu durch diese Zahl merkt.
Wird die Box neu gestartet sucht sich Gemini das Skript mit der beim Runterfahren gespeicherten Nummer raus und startet es. Findet es kein Skript mit dieser Nummer gibt's eine Fehlermeldung: "cam id nicht gefunden". Das ist nicht weiter schlimm, bedeutet einfach dass kein Skript mit der gesuchten CAMID gefunden worden ist.
Das Einlesen der vorhandenen Skripts (mit *_cam.sh* Endung) mit dazugehörigen Nummern wird beim Enigmastart gemacht oder wenn der User im Menu *CamEinstellungen -> alle Cams -> Cam Linste aktualisieren* klickt.
Also nicht Wundern wenn ein geänderter Emu Name nicht gleich im Blue Panel erscheint.

> **Zitat:**
> CAMNAME="CCcam-1.2.0"

Der CAMNAME ist die Zeichenfolge die im Bluepanel angezeigt wird. Benutzt man Skins die das akive Emu anzeigen wird auch diese Zeichenfolge angezeigt, deshalb sollte man auf die Länge achten.

Zitat:
ZAPTIME=6

Ist die Zeit die das Emu braucht bevor am Bildschirm nach dem Emuwechsel etwas hell wird.

Zitat:
INFOFILE="ecm.info"

Das File wo die aktuellen Ecm Infos abgelegt werden (im Gemini können die auch in der Skin angezeigt werden)

nun beginnt das eigentliche Skript

Zitat:
case "$1" in

als erstes wird der Parameter (start oder stop) ausgelesen mit dem das Skript gestartet wurde

Zitat:
start)
echo "[SCRIPT] $1: $CAMNAME"
/var/bin/CCcam -v
;;

start) - ist der Parameter start dann passiert folgendes
echo "[SCRIPT] $1: $CAMNAME" - macht einfach Ausgabe com Skriptnamen, dem Parameter und dem Namen das Cams
/var/bin/CCcam -v - nun wird die CCcam gestartet, im Beispiel ist dann noch der Parameter -v (verbose) für die Log Ausgabe im Syslog dabei, das ist allerdings spezifisch für dieses Emu
;; - Ende des Start Skripts

Zitat:
stop)
echo "[SCRIPT] $1: $CAMNAME"
killall CCcam
;;

stop) - ist der Parameter stop dann passiert folgendes
echo "[SCRIPT] $1: $CAMNAME" - Ausgabe wie bei Start
killall CCcam - das Emu wird beendet
;; - Ende des Stop Skripts

Zitat:
*)
$0 stop
exit 1
;;
esac

Hier wird noch festgelegt was passieren soll wenn kein Parameter angegeben wird, in desen Fall -> Emu Stop und die Abfrage beendet.

Möchte man dass neben CCcam auch noch der Cardserver NewCS gestartet wird, muss ganz einfach noch der Aufruf dazu, am besten man gibt dem Newcs etwas Zeit um durchzustarten (sleep 3) das sind 3 Sekunden (damit ihm CCcam nicht die Slots wegschnappt), danach wird CCcam gestartet

> **Zitat:**
> ..
> echo "[SCRIPT] $1: $CAMNAME"
> /var/bin/newcs
> sleep 3
> /var/bin/CCcam -v
> ..

Beim Stoppen sieht das dann ähnlich aus

> **Zitat:**
> ..
> echo "[SCRIPT] $1: $CAMNAME"
> killall newcs
> killall CCcam
> ..

Die Filenamen der ausführbaren Dateien sind genau zu übertragen (auch Gross- und Kleinschreibung)

Zum Testen des Skripts einfach im Telnet

> **Zitat:**
> /var/skripts/emu_cam.sh start

wenn alles passt sollte das Emu nun starten, die Anzeige im Bluepanel wird nicht aktualisiert!

Diese Anleitung wurde von der Autorin Tissa, vom Dream Pirates Board erstellt.
Ihr gilt der Dank für diese tolle Anleitung und die freundliche Genehmigung, zur Veröffentlichung auf dem DFB Board.