die datenschleuder.

das wissenschaftliche fachblatt für datenreisende
ein organ des chaos computer club

Byebyecicle.

ISSN 0930-1054 • 2004
Zwo Euro fuffzich, bitte sehr.
Postvertriebsstück C11301F

#85

Erfa-Kreise / Chaostreffs

Bielefeld im AJZ, Heeper Str. 132 >> mittwochs ab 20 Uhr http://bielefeld.ccc.de/ info@bielefeld.ccc.de

Berlin, CCCB e.V. (Club Discordia) Marienstr. 11, (Briefe: CCCB, Postfach 640236, D-10048 Berlin) >> donnerstags ab 17 Uhr http://berlin.ccc.de/

Düsseldorf, CCCD/Chaosdorf e.V. Fürstenwall 232 >> dienstags ab 19 Uhr http://duesseldorf.ccc.de mail@duesseldorf.ccc.de

Erlangen/Nürnberg/Fürth, BitsnBugs e.V. "E-Werk", Fuchsenwiese 1, Gruppenraum 5 >> dienstags ab 19 Uhr http://erlangen.ccc.de/ mail@erlangen.ccc.de

Hamburg (die Dezentrale) Lokstedter Weg 72 >> 2. bis 5. Dienstag im Monat ab etwa 20 Uhr http://hamburg.ccc.de/ mail@hamburg.ccc.de

Hannover, Leitstelle511 Kulturcafé, Schaufelder Str. 30, Hannover >> 2. Mittwoch im Monat ab 20 Uhr https://hannover.ccc.de/

Karlsruhe, Entropia e.V. Gewerbehof, Steinstr. 23 >> sonntags ab 19:30 Uhr http://www.entropia.de/ info@entropia.de

Kassel Uni Kassel, Wilhelmshöher Allee 71-73 (Ing.-Schule) >> 1. Mittwoch im Monat ab 18 Uhr http://kassel.ccc.de/

Köln, Chaos Computer Club Cologne (C4) e.V. Chaoslabor, Vogelsanger Str. 286 >> Letzter Donnerstag im Monat ab 19:30 Uhr http://koeln.ccc.de/ mail@koeln.ccc.de

München, muCCC e.V. Kellerräume in der Blutenburgstr. 17 >> 2. Dienstag im Monat ab 19:30 Uhr http://www.muc.ccc.de/

Ulm Café Einstein an der Uni Ulm >> montags ab 19:30 Uhr http://ulm.ccc.de/ mail@ulm.ccc.de

Wien, chaosnahe gruppe wien Kaeuzchen, 1070 Wien, Gardegasse (Ecke Neustiftgasse) >> Alle zwei Wochen, Termine auf Webseite http://www.cngw.org/

Aus Platzgründen können wir die Details aller Chaostreffs hier nicht abdrucken. Es gibt aber in den folgenden Städten Chaostreffs mit Detailinformationen unter http://www.ccc.de/regional/ : Aachen, Bad Waldsee, Basel, Bochum, Darmstadt, Dortmund, Dresden, Frankfurt am Main, Freiburg im Breisgau, Gießen/Marburg, Hanau, Heidelberg, Ilmenau, Mainz, Mülheim an der Ruhr, Münster/Osnabrück, Offenbach am Main, Paderborn, Regensburg, Stuttgart, Trier, Weimar, Wuppertal.

Friends & Family

Zur näheren Chaosfamilie zählen wir (und sie sich) die Häcksen (*http://www.haecksen.org/*), den/der "Verein zur Förderung des öffentlichen bewegten und unbewegten Datenverkehrs e.V." - FoeBuD (*http://www.foebud.de/*), den Netzladen e.V. in Bonn (*http://www.netzladen.org/*) und die c-base Berlin (*http://www.c-base.org/*).

Die Datenschleuder Nr. 85

Herausgeber *(Abos, Adressen, Verwaltungstechnisches etc.)*
Chaos Computer Club e.V., Lokstedter Weg 72, D-20251 Hamburg, Fon: +49.40.401801-0, Fax: +49.40.801401-41, <office@ccc.de> Key fingerprint: 0891 587D 8936 CB96 0EFE F4B0 B156 0654 617C AB8E

Redaktion *(Artikel, Leserbriefe, Inhaltliches, etc.)*
Redaktion Datenschleuder, Postfach 640236, D-10048 Berlin, Fon: +49.30.28097470, <ds@ccc.de> Key fingerprint: 03C9 70E9 AE5C 8BA7 42DD C66F 1B1E 296C CA45 BA04

Druck *Pinguindruck, Berlin; http://pinguindruck.de/*

ViSdP und Produktion
Tom Lazar, <tom@tomster.org>

Layout *Dirk Engling, Tom Lazar,*

Redakteure dieser Ausgabe
Dirk Engling <erdgeist> und Tom Lazar <tomster>

Autoren dieser Ausgabe
Alexander Bernauer, Volker Birk, Matthias "wetter" Mehldau, Alien8, Mirko Swillus, Lars Weiler, elektra

Dieses Jahr war ein entmutigendes Jahr. Biometrie ist in die Reisepässe gekommen, das Urheberrecht wurde verschärft, das Bankgeheimnis ist uns durch die Finger geglitten und die gute alte Briefmarke ist quasi abgeschafft.

Ja, richtig! Abgeschafft. Ausgetauscht gegen einen herzlosen, volldigitalen 2D-Barcode. Unlesbar für uns Menschen und jeder Ästhetik beraubt. Und bei dem Versuch "darf ich dir mal meine Briefmarkensammlung zeigen..?" wird es in Zukunft skeptische Blicke geben.

Nachdem wir in den letzten Wochen mit einer ungewöhnlichen Fülle von anonym zugesandten und oftmals hochspannenden Dokumenten bedacht wurden, möchte die Redaktion die Gelegenheit benutzen allen Einsendern auf diesem Wege zu danken. Wir mussten uns in dieser Ausgabe aus Platzgründen auf den Hack-a-Bike Report und die nähere Betrachtung des Gesundheitskarten-Backends beschränken, aber werden die anderen uns zugespielten Leckerbissen auf jeden Fall in der nächsten Ausgabe vorstellen.

Für den Fall das dem einen oder anderen Leser Papierstücke oder Bits in die Hände fallen die im öffentlichen Interesse der Publikation in der Datenschleuder bedürfen, bitten wir um Zusendung per Post (bitte den bewährten unmarkierten braunen Umschlag für anonym zugespielte Dokumente verwenden) oder e-mail (an ds@ccc.de, bei hohem Risikofaktor wird ein Anonymizer sowie die geschickte Nutzung eines öffentlichen Netzes empfohlen).

Entmutigende Tendenzen hin oder her: sie unterstreichen nur die Dringlichkeit, uns zu organisieren und tätig zu werden. Vor diesem Hintergrund betrachtet, lehnen wir uns bestimmt nicht zu sehr aus dem sprichwörtlichen Fenster, wenn wir Euch einen heissen diesjährigen Congress versprechen.

Die Datenschleuder-Redaktion

P.S. Einige von Euch werden es vielleicht bemerkt haben: ohne grosse Ankündigung (Neudeutsch: Launch) hat die Datenschleuder mittlerweile nach und nach eine Online-Heimat gefunden. Unter `http://ds.ccc.de` *können nun Artikel der (meisten) vergangenen Ausgaben, sowie die Titelgeschichte der aktuellen Ausgabe nachgelesen werden und finden so Ihre suchmaschinenerfassbare Heimat.*

Inhalt

Ein HackABike in freier Wildbahn - der Ausschnitt zeigt den im Artikel besprochenen Kasten.

Der Redaktion Datenschleuder wurde eine Technologieanalyse anonym zugespielt, die unten stehend zu Bildungszwecken dokumentiert ist. Der Text wurde redaktionell gekürzt.

HACK A BIKE

von anonymous, Leserbriefe bitte an <ds@ccc.de>

Schon immer mal nachts ohne Transportmöglichkeit in einem fremden Bezirk aufgewacht? "Mal schnell" ein Fahrrad benötigt? In Berlin und anderen Großstädten Deutschlands bietet Die Bahn mit dem Call-A-Bike Service Abhilfe.

Im November 2003 wurde uns ein CallABike 'zugetragen', das nicht richtig abgeschlossen wurde. Dieses musste erstmal als Testobjekt herhalten. Die meisten dachten, dass in dem Schlosskasten GPS oder sonstiger Funk enthalten sei, nach dem Öffnen war hiervon jedoch nix zu sehen. Um die Schrauben der Schlosskästen zu öffnen, benötigt man nur ein Torx TR (Temper Resistant). In dem Kasten ohne Display ist die Stromversorgung durch Batterien sichergestellt (3x 1.5V Mono).

Die beiden Kästen sind durch eine Art Bügel miteinander verbunden. In diesem Bügel befindet sich ein sechspoliges Kabel für den Strom und zwei Spulen. Damit kann geprüft werden, ob das Schloss wirklich geschlossen ist, oder einfach kein oder nur irgendein anderer Bolzen zum Verschließen genommen wurde.

Der Kasten mit dem Display enthält den Exzentermotor zum Öffnen des Schlosses, zwei Taster (Mikroschalter) und ein kapazitives 5x2-Touchpad. Die Hauptlogik sitzt unter dem Display. Sie ist nochmal durch eine Metallplatte abgesichert, welche nur die Kabel zum Display/Touchpad durchlässt. Damit wird der Motor und der Verschlussmechanismus vor einer Attacke durchs Display geschützt.

Die gesamte Platine ist mit schwarzem Silikon übergossen, das man erstmal runterkratzen muss. Das geht prima mit einer Mess-Spitze. Außer einer streichholzschachtelgroßen Platine (Rückseite Datenschleuder 82), auf der ein Atmel AT90S8535 (8-Bit-Risc-Prozessor, 4x8-IO-Pins, 8KB Flash, 512-Bytes-EEProm und 512-Bytes-RAM) [1], ein paar LEDs (rot, grün und IR) und IR-Receiver aufgebracht sind, enthält der Kasten noch ein paar andere elektrische Bauteile (Motor, Schalter und ein Beeper). Es ist auch ein Neigungssensor vorhanden, aber im Code wird er nicht benutzt. Daher bestand keine Gefahr, uns zu orten.

Es wurden ein paar Bilder von der Aktion gemacht, aber dann lag die Technik erstmal zwei Monate einsam in einer Kiste, weil wir es nicht geschafft haben, das CallABike zu booten. Es dauerte eine Weile, bis wir merkten, dass das System nach dem Booten durch ein Infrarot-Signal akti-

Kurzeinführung in das CallABike System

Als Kunde ruft man die CallABike-Zentrale und gibt per DTMF-Wahl die vierstellige Radnummer durch. Von der Zentrale erhält man dann den vierstelligen Code, mit dem man das CallABike-Fahrad öffnen kann. Zur Sicherheit wird man nach dem Anruf von der Zentrale zurückgerufen. Die letzten vier Ziffern des Rückrufes enthalten nochmal den Code - annehmen muss man den Anruf deswegen nicht. Jetzt läuft die Uhr und der Kunde muss pro Minute 6 Cent bezahlen (mit Bahncard nur 4 Cent). Wenn der Kunde das CallABike einfach mal schnell abstellen will (z.B. für einen kurzen Einkauf), kann man das CallABike abschließen und im Display auf 'Nicht Abgeben' tippen. Es ist sozusagen kurz geparkt. Danach kann man das CallABike mit dem gleichen Code wie beim ersten Mal öffnen. Das kann man so oft wiederholen, wie man möchte. Die Zeit, die man bezahlen muss, läuft natürlich weiter.

Wenn der Kunde das CallABike dann endgültig abgeben will, muss er beim Schließen auf 'Abgeben' tippen; das CallABike gibt einem dann den Rückgabecode. Mit diesem Code kann man gegenüber der Zentrale 'beweisen', dass man das CallABike wirklich wieder abgeschlossen hat. Man ruft jetzt einfach wieder die Zentrale an und gibt den Rückgabecode durch. Danach muss man noch die Straßenecke auf Band sprechen, an der man das CallABike abgestellt hat. Die Mietzeit wird damit dann auch beendet.

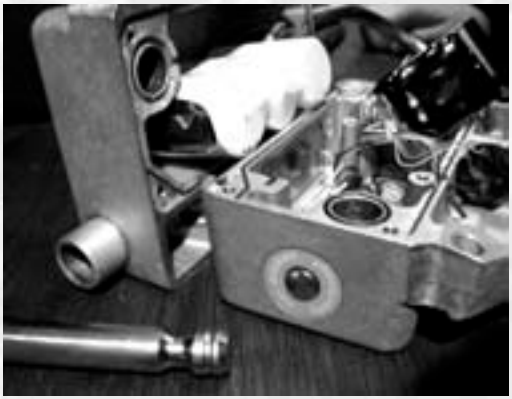
Es ist auch möglich, zwei Räder mit einem Anruf auszuleihen oder abzugeben. Wenn der Kunde in seiner Nähe kein CallABike-Rad findet, kann er auch die Zentrale anrufen und fragen, wo das nächste CallABike-Rad steht. Ein Servicemitarbeiter der Bahn schaut dann in der Datenbank nach und gibt den Standort des nächstgelegenen CallABike-Rads durch.

viert werden muss. Das war mehr oder weniger Zufall. Wenn man eine normale Glühlampe benutzt, um besser sehen zu koennen, piepte die Elektronik gelegentlich scheinbar unmotiviert. Wie sich später herausstellte, reichte der durch die Glühlampe emittierte IR-Anteil aus, um den IR-Receiver zu triggern und den Bootvorgang fortzusetzen. Beim Booten testet sich das System selbst, und der Empfang eines Infrarot Signals gehört eben dazu. Im Zuge fortschreitender Professionalisierung™ wurde in der Folgezeit die Glühlampe durch ein Infrarot-Photon-Micro-Light ersetzt.

Bei unserer weiteren Analyse des Systems begannen wir damit, alle Anschlüsse des Atmel durchzumessen, um uns einen ungefähren Schaltplan zu erstellen **(siehe Bild)**. Die Datenblätter für den Atmel und das verwendete Display haben wir uns aus dem Web besorgt.

Im Januar hatte einer der Beteiligten dann endlich eine Idee, wie weiter vorzugehen sei. Auf der Platine war uns eine unbenutzte 6-polige Steckerleiste aufgefallen, und wie sich herausstellte, handelt es sich dabei um den ISP-Connector (In System Programming) des Atmel. Daran schlossen wir dann ein Atmel-Developer-Board (STK500) an. Zum Auslesen wurde hauptsächlich das freie UISP (*"Uisp is a tool for AVR microcontrollers which can interface to many hardware in-system programmers"*[2]) benutzt. Die auf dem Atmel vorhandenen „Intellectual-Property"-Bits waren in einem undefinierten Zustand, deswegen konnten wir das Flash des Atmels mit der 8KB großen Firmware auslesen.

> … Es gibt natürlich auch andere Zeitgenossen, die haben, schon aus sportiven Gründen, allerlei versucht, um die Standfestigkeit der Hardware oder das elektronische Prinzip der eingebauten Mikrochips und Prozessoren zu ergründen.
>
> Sie rückten dem Schloss mit Schraubenziehern und gängigen Imbusschlüsseln zu Leibe.
>
> Sie versuchten ihr Glück mit Brechstange, Vorschlaghammer, sogar mit der Motorflex. Oder, ganz Smart, mit Laptop, mit Dechiffrierprogrammen, auch mit Fangfragen an das Wartungspersonal. Doch vergebens! Wieder lächelt Reth, der einst erste Ausflüge auf einem grünen Puky-Rad unternahm, sich heutzutage aber als "postmoderner Urbaniker", denn als "Fahrradfreak" versteht. Er lächelt und sagt: "Erst diese Technik macht uns zum weltweit einzigen stationsunabhängigen Stadtradsystem. Der Code ist nicht zu knacken und darauf sind wir richtig stolz."
>
> **… aus dem Kundenmagazin DB mobil**



In den nächsten Wochen waren mehrere Hacker damit beschäftigt, den ausgelesenen Assemblercode zu verstehen und zu dokumentieren. Dazu verwendeten wir AVR-Studio [3] und Ida Pro [4]. Den Scramble Code (zum berechnen der Ausleih- und Abgabecodes) fanden wir relativ schnell, da sich dort eine Menge rotate-und-shift-Befehle befanden. Den Initialisierungscode erkannten wir wieder, da wir wussten, dass der Motor sich beim Einschalten zweimal herumdreht. So konnten wir das gelernte immer wieder an unserem Prototyp auf Richtigkeit überprüfen.

Die Ausleih- und Abgabecodes werden durch einen Scrambler generiert, der mit einem 16Bit-Counter des CallABikes und einem Zustandswert aufgerufen wird. Ein gerader Counterwert erzeugt Ausleihcodes und ein ungerader erzeugt die Abgabecodes. Der Scrambler nutzt den Counter und das Zustandsbyte, um ein Offset auf ein 1024 Bit großes Feld zu errechnen. Dieses Feld ist ein für jedes CallABike eindeutiger binärer String, der als der (wahrscheinlich) eindeutige Key des CallABikes bezeichnet werden könnte. Von diesem Offset aus werden dann 4x4 Bit genutzt, die die vier Ziffern für die Ausleih- und Abgabecodes repräsentieren. Die 16 Bit des Counters werden aber schlecht genutzt, denn schon nach 1024 Iterationen wiederholen sich die Ausleih- und Abgabecodes. Das bedeutet auch, dass es nur 512 Ausleihcodes je CallABike gibt, da es nur 512 gerade Offsets gibt die auf den Key (1024 Bit) zeigen können. CallABikes, die wir geöffnet haben und die wir wegen der Lockbits nicht auslesen konnten, haben wir mit einem Script 511 mal resetten lassen (bei einem Reset erhöht sich der Counter immer um zwei). Damit haben wir den ursprünglichen Zustand

EEPROM Inhalt:

```
0x0000 - 0x0001 unused
0x0002          lock_sensor_calibration
0x0003 - 0x0019 unused
0x001A - 0x001B 16bit counter (scrambler)
0x001C          unused
0x001D - 0x001F CallABike Radnummer
0x0020 - 0x009F 128 Byte Random (Key)
0x00A0 - 0x00A2 Die ersten drei Bytes des Key nocheinmal
0x00A3 - 0x00AF unused
0x00B0 - 0x01FF Text für Display
```

wiederhergestellt, und das CallABike war wieder 'in sync' mit der Zentrale.

Wer sich das Display mal genauer angeschaut hat, wird festgestellt haben, dass der Zeichensatz ein proportionaler ist. Dazu gibt es im Code eine Tabelle, in der die Länge des Zeichens und die Position im Flash gespeichert sind. Ein 'i' und ein '!' belegen nur ein Byte, wogegen z.B. ein 'w' sieben Bytes belegt. Die großen Logos und das Zahleneingabefeld liegen als 400 Byte große Bitmaps vor. Die lange schwarze Linie im CallABike-Logo zeigt die Stärke der Spule im Schloss an. Das haben wir nur durch das Code-Auditing herausgefunden.

Unser erstes Ziel war es, den aus unserem Disassembler erhaltenen Sourcecode so anzupassen, dass nach dem Assemblieren mit dem Commandline Tool Avra ("Assembler for the Atmel AVR microcontrollers" [5]) ein **EXAKT** identisches Binary herauskam. Auf der Grundlage dieses Referenzcodes konnten wir dann endlich Änderungen vornehmen.

Nachdem wir uns diese Grundlage geschaffen hatten, konnten wir das CallABike mit unserem eigenen Code flashen.

Da wir keine Vulnerabilities oder Backdoors fanden (jedes CallABike hat einen eigenen Key, der im EEPROM gespeichert ist), mit denen man ein CallABike exploiten könnte, ohne es aufzuschrauben, kamen wir auf die Idee, uns eine eigene Backdoor in den Code zu programmieren. Hört sich eigentlich ganz leicht an, ist es aber nicht. Erstmal mussten wir den Code der BAHN optimieren, um uns den entsprechenden Platz zu schaffen. Schließlich sollte ja auch noch ein Logo mit 400 Bytes von uns in das 8KB große Flash. Den Datenmüll, den man über unserem HackABike Logo sehen kann, ist der Backdoor Code. Das sparte nochmal ca. 150 Bytes. Außerdem wollten wir nicht, dass man mit dem Backdoor Code normalen Kunden, die das Rad nur geparkt (verschlossen, aber nicht abgegeben) haben, das ausgeliehene HackABike wegschnappen konnte. Das erforderte noch ein paar Zeilen mehr im Code. Auch kann man mit unserem Backdoor Code das HackABike nicht 'parken', da ja der Öffner nichts bezahlen muss und deswegen nicht motiviert ist, sich weiter um das Rad zu kümmern, und es aber auch niemand anders aufmachen könnte. Um das HackABike auch auf größere Entfernung noch von seinen unbehandelten Verwandten unterscheiden zu können, haben wir ihm eine leicht veränderte Blink-Sequenz beigebracht.

Code zum Generieren der Abgabe/Ausleihcodes bei gegebenem Key aus dem eeprom

```
unsigned char g_key[4];

void scrambler(uchar param, long counter) {
    long bitoffset;
    uchar r21 = param, r28 = 1;
    short r27_26 = counter, short r31_30;
    r28 <<= r27_26 & 7;
    r27_26 += r21;
    r27_26 &= 0x3ff;
    r31_30 = r27_26;
    r27_26 <<= 5;
    r27_26 -= r31_30;
    r27_26 &= 0x3ff;
    r27_26 += r28;
    r27_26 &= 0x3ff;
    bitoffset = r27_26 & 7;
    r27_26 >>= 3;
    r27_26 += 0x20;
    r27_26 &= 0xff;
    fillkey(r27_26,bitoffset);
}

void fillkey(long address, long bitoffset) {
    uchar r16;
    long fullkey;
    fullkey  = eeprom[address++] << 16;
    fullkey += eeprom[address++] << 8;
    fullkey += eeprom[address++];
    fullkey >>= bitoffset;
    r16 = fullkey         & 0xf;
    if(r16 >= 10) r16 -= 10;
    g_key[3] = r16;
    r16 = (fullkey >> 4 ) & 0xf;
    if(r16 >= 10) r16 -= 6;
    g_key[2] = r16;
    r16 = (fullkey >> 8 ) & 0xf;
    if(r16 >= 10) r16 -= 10;
    g_key[1] = r16;
    r16 = (fullkey >> 12) & 0xf;
    if(r16 >= 10) r16 -= 6;
    g_key[0] = r16;
}
```

Im Verlauf der weiteren Analyse des Codes ist uns aufgefallen, dass das CallABike im Abgabecode integriert Statusinformationen an die Zentrale durchgeben kann. Je nachdem, in welchem technischen Zustand sich das CallABike befindet, kann es unterschiedliche Rückgabecodes - abhängig vom bereits erwähnten Zustandsbyte - angeben. Das CallABike kann z.B. melden, dass die Batterie nicht mehr lange hält, oder der Motor für das Schloss nicht mehr in der richtigen Stellung ist. Wenn man z.B. den Schließknopf sieben mal ohne eingeführten Bolzen drückt, liefert der Scrambler einen entsprechenden Rückgabecode, der gültig ist, diesen Zustand aber für die Zentrale erkennbar anzeigt. Von diesen Codes gibt es 52 (eine Matrix aus 4x13).

Die Backdoor erlaubt, das HackABike mit einem von uns festgelegten Ausleihcode einfach zu öffnen. Wenn man das HackABike dann wieder abgibt, ist es ganz normal wieder ausleihbar. Es steht auch wieder der Ausleihcode des vorherigen Kunden im Display. Die Zentrale merkt von diesem eingeschobenen Ausleihvorgang nichts - außer, dass es an einem anderem Ort steht, als es in der Datenbank vermerkt ist. Wenn es dann aber wieder normal ausgeliehen und abgestellt wird, ist auch in der Zentrale alles wieder in Ordnung.

Fürs CallABike mit der Nummer 2883 z.B.:

```
unsigned char eeprom[ ] = {
0x5A,0xD5,0xAD,0x6B,0xFD,0xD7,0x34,0x78,
0xB3,0x03,0x22,0x13,0x61,0x23,0xAD,0xFE,
0x51,0x6E,0xAA,0xA2,0xD4,0xB7,0xBA,0xC0,
0x78,0x9A,0x84,0x55,0x2A,0xB9,0x6E,0xBC,
0x33,0x15,0x2C,0x97,0x33,0x98,0x4B,0x78,
0x43,0xE5,0x20,0xD5,0x1C,0x1C,0x75,0x12,
0x2A,0x91,0x17,0xFC,0x0C,0x61,0x31,0x31,
0x50,0x6D,0xFD,0x5C,0xC5,0x60,0x8D,0xE0,
0x0A,0xF2,0x85,0xF1,0x3B,0xA3,0xBD,0x74,
0xF3,0xD4,0x9E,0xBB,0x45,0x95,0x69,0x24,
0x79,0x36,0x9A,0xA6,0x66,0x96,0xFB,0xE8,
0x5D,0x38,0x34,0x28,0xC0,0x51,0x3B,0x18,
0x46,0xCA,0xD9,0xE3,0xD7,0xC8,0x86,0x01,
0x11,0x60,0xF2,0xF0,0xA4,0xA4,0xEF,0x16,
0x3E,0xBE,0xB9,0x1F,0xA8,0xF9,0x61,0x0B,
0xD6,0x7F,0x75,0xE7,0xF4,0x31,0x3F,0x6B
};
```

Um ein CallABike in ein HackABike zu verwandeln, mussten wir sechs Schrauben auf der Innenseite des Schlosskastens mit dem Display öffnen und das Kabel des STK500 an den ISP-Anschluss der Platine stecken. Danach haben wir ein Script gestartet, dass das Flash und den EEPROM Bereich auslies. Das EEPROM wird danach mit zurückgesetztem Counter und dem Flash mit unserer Backdoor wieder zurückgeschrieben. Damit niemand unsere Backdoor auslesen kann, haben wir natürlich noch die Lockbits gesetzt. Ein geschulter Hacker brauchte ca. 12 Minuten, um zwei CallABikes parallel in HackABikes zu verwandeln.

Da UISP das Setzen der Lockbits nicht korrekt unterstützte, mussten wir das erstmal einbauen. Dazu haben wir den Output von AVR-Studio mit einem serial sniffer mitgelesen und uns die entsprechenden Kommandos für das STK500 rausgesucht und in UISP eingebaut.

```
/usr/local/bin/uisp -dserial=\
/dev/ttyU0 -dprog=stk500 \
-dpart=AT90S8535 --erase --upload \
--verify if=flash_backdoor.hex
```

Abschließend ist festzustellen, dass das technische Design des CallABike in unseren Augen sehr gut ist. Jedes CallABike hat vermutlich einen eigenen 1024 Bit Key, der benötigt wird, um die Abgabe- und Ausleihcodes berechnen zu können. Dazu muss vermutlich das CallABike geöffnet und ausgelesen werden. Es wurde nur versäumt, die Lockbits zu setzen, um die Firmware vor dem Auslesen zu schützen. Unsere Attacke ist von den verbrauchten Mannstunden wohl mehr Wert als ein paar Dutzend CallABikes.

Die UART-Loop erlaubt folgende Kommandos:

0x5B Fahrradnummer auslesen

0xCE Spule Kalibrieren

0xC5 RAM ab 0x00AD lesen

Nach Eingabe der Ersten 32 Randombytes (Key):

0xCA Watchdog enablen (rebooten)

0xC8 Randombereich im EEPROM schreiben und wieder lesen

0xCD Andere Bereich im EEPROM schreiben und wieder lesen

[1] *http://www.atmel.com/dyn/resources/prod_documents/1041S.PDF*
[2] *http://savannah.nongnu.org/projects/uisp/*
[3] *http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725*
[4] *http://www.datarescue.com/*
[5] *http://avra.sourceforge.net/*

Auszug aus dem eeprom des CallABike 2882, pro Counter jeweils 3 Ausleihcodes gefolgt von den 52 möglichen Abgabecodes

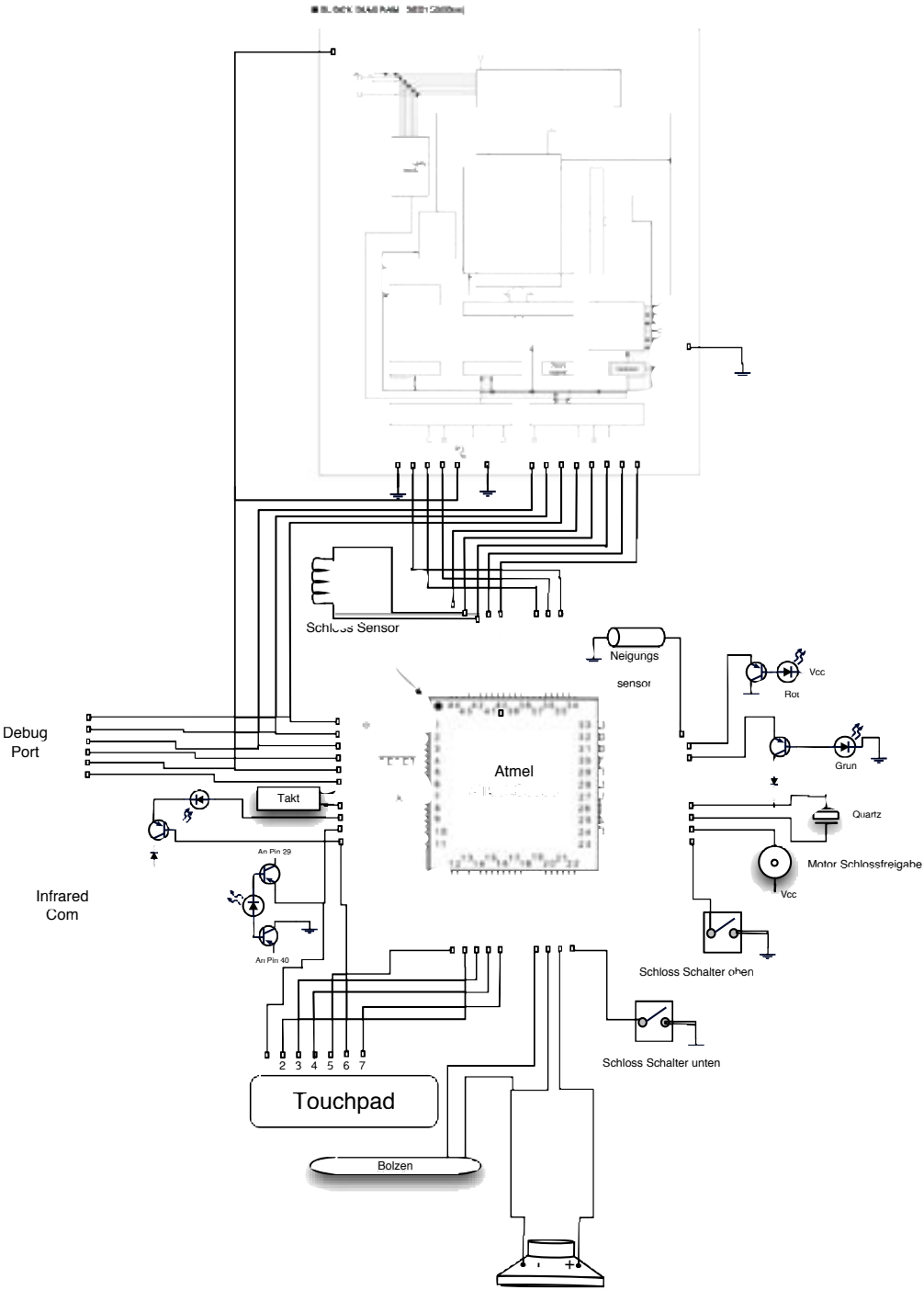
```
Counter 0x01E4: 3053 2671 1775
    -00- -01- -02- -03- -04- -05- -06- -07- -08- -09- -10- -11- -12- -13-
00: 8234 7161 4355 4892 9290 9998 8304 7365 9562 2095 3043 6551 7590 7270
01: 8589 6501 7447 1493 6180 3012 1741 8518 9843 8709 9172 1151 9723 9368
10: 9544 7869 5662 4655 8255 2595 9391 6608 8674 3599 4120 6087 5181 8181
11: 3273 7512 6017 4884 8282 9760 7439 2556 6031 9146 0413 3699 9160 2446

Counter 0x01E6: 7145 5444 7084
    -00- -01- -02- -03- -04- -05- -06- -07- -08- -09- -10- -11- -12- -13-
00: 1382 8104 4463 7399 2412 4608 5856 1205 4872 7241 7327 7391 8241 5088
01: 9445 6228 3243 9514 7081 1365 0360 6858 1434 3263 7921 5394 7103 9678
10: 5372 3104 5861 9271 8793 4825 4210 7162 2400 9084 4227 4645 5182 7942
11: 0325 2894 4580 7946 3428 8162 2675 0674 7051 2872 6088 5256 4789 9610
```

Hier sieht man das CallABike 2882 mit dem Counterstand 0x01E4. Ein Kunde erhält als Öffnungscode die 3053. Wenn das CallABike normal abgegeben wird und alles in Ordnung ist, bekommt er als Abgabecode die 8234. Falls z.B. die Batterie schwach ist (-01-) bekommt er als Code die 7161.

Bit 0-5 (x-Dimension) dezimal:
-00- Alles OK
-01- Batteriespannung niedrig
-02- Schloss ist nicht richtig zu (Spule)
-03- gelb-blaues Kabel defekt (Spule im Bolzen)
-04- grün-graues Kabel defekt (Spule im Schlosskasten)
-05- Motor ist nicht in richtiger Position / Knopf nicht gedrückt
-06- Motor ist nicht komplett herum
-07- Knopf nicht gedrückt
-08- Rückgabe default (unklar)
-09- unklar
-10- mind. drei misslungene Abgabeversuche
-11- sieben misslungene Abgabeversuche. Schloss ist nicht zu!
-12- unklar
-13- unklar

Bit 6-7 (y-Dimension) binär
-00- Alles OK
-01- Batteriewechsel nötig
-10- unklar
-11- unklar


Schloss Sensor
Neigungs
sensor
Vcc
Rot
Grun
Debug
Port
Atmel
Takt
Quartz
An Pin 29
Motor Schlossfreigabe
Vcc
Infrared
Com
An Pin 40
Schloss Schalter oben
Schloss Schalter unten
2 3 4 5 6 7
Touchpad
Bolzen

Das Gesundheitskarten-Backend

von Verband forschender Patienten im CCC <ds@ccc.de>

IT-Projekte der öffentlichen Hand kommen nur zu 10% zum erfolgreichen Abschluß. Aber selbst wenn sie abgeschlossen werden, wie es bei der Gesundheitskarte im Moment aussieht, kann die eigentliche Katastrophe noch ausstehen.

Vorwort

Die Planungen zur Gesundheitskarte sehen vor, die Patientendaten zentral zu speichern und Ärzten über ein Datennetz zugänglich zu machen. Auch soll die Gesundheitskarte des Patienten die Rezepte speichern. Der Patient geht damit zur Apotheke, es gibt keine Notwendigkeit für Zettel mit unleserlichen Verschreibungen und leicht fälschbaren Unterschriften mehr.

An sich kein Problem, aber Patientendaten (das beinhaltet neben den üblichen personenbezogenen Daten die Krankengeschichte des Patienten, wogegen er womit behandelt wurde, wogegen er allergisch ist, etc.) sind sozusagen der Inbegriff der vom Datenschutz gemeinten schutzbedürftigen Daten. Daher kommen hier diverse Bedrohungsszenarien ins Spiel:

• Wenn jemand ins zentrale Rechenzentrum einbricht, darf er die Daten nicht lesen können

• Wenn jemand beim Arzt einbricht, darf er die Daten von dessen Patienten nicht lesen können

• Darf der Apotheker Zugriff auf die Krankengeschichte des Patienten haben?

• Vielleicht will man noch, daß der Gynäkologe keinen Zugriff auf Daten von Männern hat? Oder daß der HNO-Arzt nicht zu sehen kriegt, ob der Patient an Geschlechtskrankheiten leidet?

Implementation

Als Fundament der Infrastruktur hinter der Gesundheitskarte ist das PaDok-Konzept der Fraunhofer-Gesellschaft gewählt worden, an dem diese seit 1993 arbeiten. Der gute Name der Fraunhofer-Gesellschaft gekoppelt mit der langen Projektlaufzeit deuten auf eine besonders saubere Planung gekoppelt mit makelloser Umsetzung durch international namhafte Experten und Tests durch unabhängige Audit-Institute hin.

Beim Blick in die PaDok-FAQ [1] kommen uns allerdings erste Zweifel:

Q: Inwieweit enthält die Lösung proprietäre bzw. offene / standardisierte Komponenten?

A: D2D / PaDok setzt durchgängig auf standardisierte Formate und Schnittstellen. Einzige Ausnahme hiervon ist das eigentliche Transport-Protokoll zwischen dem (als datentechnisch "unsicher" einzustufenden) Kommunikations-Client und dem jeweils zugeordneten Server. Hier wird ein spezielles und hoch-ritualisiertes RPC-Protokoll verwendet, dass sich jedem Zugangs-Versuch mit Hilfe allgemeingültiger Internet-Werkzeuge verschließt.

Ein "hoch-ritualisiertes RPC-Protokoll" klingt dann doch eher nach Quacksalberei als nach Expertentum. Aber gut, das kann ja ein Übersetzungs- oder Dokumentationsfehler sein. Weiter unten in dem Dokument erfahren wir, daß die PaDok-Software auf einem existierenden Windows-Rechner im lokalen Netz der teilnehmenden Praxis laufen soll. Einzige Anforderungen: hat ISDN und kann auf einen File-Server schreiben, wo die Dokumente dann hin sollen. Bei den Worten "existierender PC" stellt sich aber die Frage, ob das tatsächlich ein separater Rechner sein wird oder die Software einfach mit auf den PC der Praxis-Buchhaltung installiert wird.

Das PaDok-Konzept ist der Backend-Teil der Gesundheitskarte, d.h. zwischen Arzt und zentraler Datenhaltung. Zur Sicherheit der Gesundheitskarte selbst (die der Patient dann in den Händen hält) können wir bislang noch nichts sagen. Trotzdem ergeben sich interessante Fragestellungen. Z.B. stellt sich die Frage, ob sich ein Abhängiger über einen Trojaner auf dem PC des Arztes E-Rezepte ausstellen kann.

Wir wollten es genauer wissen und haben uns auf der CeBIT bis zum PaDok-Stand durchgeschlagen, uns umfassend beraten lassen, und Unterlagen mitgenommen. Die Unterlagen sinn voll Buzzword-kompatibel und sprechen von signaturgesetzkonformen digitalen Signaturen, starker Verschlüsselung und Einhaltung Privatsphäre.

Um so schockierender war, was uns dann von einem Fraunhofer-Mitarbeiter bei einem Bierchen von einer Teststellung erzählt wurde, die er mal kurz gesehen hatte:

1. Das Kartenterminal hatte keine PIN-Eingabe. Die PIN gibt man auf dem PC ein. Ein Trojaner kann das also abfangen.

2. Der Kartenleser wird einfach seriell angeschlossen. Mit Windows-Hausmitteln kann ein Trojaner die Kommunikation abfangen, mitlesen und sogar manipulieren.

3. Aus Kostengründen findet die Kryptographie nicht auf den Karten statt, sondern im PC. Die Karte ist lediglich ein Lagerraum für den geheimen Schlüssel, den die Karte mit der (mitgelesenen) PIN rausrückt.

Wenn also ein Angreifer einen Trojaner auf dem Arztrechner installiert, kann er den geheimen Schlüssel des Arztes extrahieren, sich ins Internet einwählen (der Arztrechner hat ISDN-Zugang für das PaDok-System, darüber kann der Trojaner sich also prima irgendwo einwählen), und den Schlüssel irgendwo ablegen. In den Unterlagen wird vorgeschlagen, daß die Ärzte sich nur über einen ISDN-Router und nicht über ISDN-Karte einwählen sollen. Man darf sich aber keiner Illusion hingeben, daß das tatsächlich hilft. Falls der Trojaner nicht nach Hause telefonieren kann bricht der Angreifer eben nachts ein und holt sich den Schlüssel von der Festplatte.

Diese Aufbewahrung des Schlüssels ist ein kapitaler Designfehler. Insbesondere stellt sich das ganze Kartenleser- und PIN-Brimborium als lächerliche Sicherheitssimulation ohne Steigerung der tatsächlichen Sicherheit heraus, wenn man sich diese Möglichkeiten vor Augen führt.

Die E-Rezepte liegen nicht komplett auf der Gesundheitskarte des Patienten, sondern auf dem D2D-Server, und auf der Karte liegt lediglich ein Freischaltcode. Mit diesem holt sich der Apotheker dann das Rezept vom Server und entschlüsselt es. So ist immerhin verhindert, daß sich ein Angreifer ohne weiteren Zugriff auf einen kompromittierten Arzt-PC E-Rezepte ausstellen kann. Aber über den Trojaner auf dem Arzt-PC Dokumente auf dem zentralen Server abzulegen erscheint nicht wirklich schwierig, zumal der Angreifer ja den Schlüssel des Arztes hat.

Eine weitere Frage ist, in welcher Form die Dokumente eigentlich verteilt werden. Die Internet-Dokumente, die wir gesehen haben, schlagen XML vor, aber das scheint nicht das einzige Format zu sein. Auf Nachfrage meinte unser Kontaktmann, daß man da auch Word-Dokumente (und damit Makroviren) transportieren kann. Ein Angreifer kann so also Arztpraxen angreifen, indem er mit Makroviren infizierte Dokumente zu seinen Patientendaten tut, z.B. indem er sie bei einem anderen Arzt abgibt als seine Fallhistorie "von einem anderen Arzt" auf Diskette, und der sie dann ins PaDok-System einpflegt.

Ein interessantes Detail ist, daß die Dokumente nur beim Transport verschlüsselt sind. Über den Microsoft-Plattformen inhärenten Makrovirenangriff kommt man so auch an Daten anderer Patienten heran, die noch auf dem System des angegriffenen Arztes herumliegen. Und da können vielleicht auch noch sensible Patientendaten liegen, die gar nicht im PaDok-System eingepflegt wurden.

Eskalation

Nachdem klar ist, daß das System auf Arztseite kompromittierbar ist, stellt sich natürlich die Frage, wieviel ein Angreifer von einem übernommenen Arztsystem aus an der zentralen Infrastruktur machen kann. Der erste Teil dieser Fragestellung ist, wie weit ein Angreifer mit einem ausgespähten geheimen Arztschlüssel kommt.

Laut *http://www.kvno.de/importiert/habu_300.pdf* authentisiert sich der Arzt-PC bei der Zentrale über seine ISDN-Rufnummer. Die Zentrale ruft dann zurück. Auch hier ist möglicherweise ein Angriff auf die umliegende Infrastruktur möglich, über umprogrammierte ISDN-Anlagen oder man würde eben über den Trojaner den Arzt-PC fernsteuern.

Die zentrale Innovation bei PaDok ist, daß man Dokumente "ungerichtet verschlüsselt". Die Fraunhofer-Gesellschaft läßt kaum eine Gelegenheit aus, auf dieses "hochwertige" Softwarepatent hinzuweisen. Die Idee

ist, daß die Dokumente auf dem Server liegen, aber der Dateiname aus der (geheimen) Vorgangskennung besteht. Wenn der Patient jetzt zu einer neuen Praxis geht, hat er von der alten einen Schrieb mit der Vorgangskennung dabei, gibt diese der neuen Praxis, und die kann dann diese Datei abrufen.

Das klingt ja erst mal wie eine gute Idee, bis man mal genauer hinschaut, wie sich so eine Vorgangskennung zusammensetzt. Der Dateiname besteht aus der Vorgangs-ID und einem "Krypto-Teil". Die Vorgangs-ID ist einfach "aa", die Arzt-ID (die man von früheren Überweisungen von/zu dem Arzt her kennt), ein fortlaufender Vorgangszähler, das Datum der Akte sowie ein Ordnerzähler (einzelner Buchstabe, beginnend mit "a"). Das ist alles trivial ratbar.

Diese Vorgangs-ID ist auf dem Server der Name eines Verzeichnisses. In dem Verzeichnis liegen die eigentlichen Dateien, und zwar mit dem Zeitstempel des Clients beim Anlegen des Dokumentes in Millisekunden.

Die Vorgangskennung besteht nun aus der Vorgangs-ID und dem "Krypto-Teil". Insgesamt hat man 21 Zeichen Platz, von denen die Vorgangs-ID nun leider schon 16 belegt. Das hinterläßt bei der gewählten Kodierung noch etwa 26 Bit für den eigentlich Schutz gegen Raten. Das alleine ist schon viel zu wenig, aber wenn man ein paar tausend IDs generiert, stellt man fest, daß es sich um die Ausgabe eines Zufallszahlengenerators mit einem 16-bit Seed handelt. Damit ist es ohne Probleme möglich, mögliche Vorgangs-IDs zu generieren, wobei man im Durchschnitt lediglich ca. 32000 generieren muß, bis man ein Dokument trifft.

Diese Dokumente, zu denen man sich so Zugang beschaffen kann, sind dann allerdings verschlüsselt. Die Frage ist, ob sie anständig verschlüsselt sind, und ob sie z.B. gegen Replay-Angriffe gesichert sind, ob ein Angreifer sich sein Metadon-E-Rezept duplizieren könnte.

Andere Richtung

Aber kann man vielleicht auch einen solchen Server direkt angreifen? Eine ISDN-Verbindung klingt ja wie eine sichere Sache. Von einem Beobachter einer Teststellung in einem Krankenhaus haben wir erfahren, daß über das ISDN eine normale TCP/IP Einwahl durchgeführt wird, und daß der Server ein einfaches Windows NT 4.0 ohne Firewall oder Dienste-Zumachen war. Wir können also davon ausgehen, daß diese Systeme von einem dedizierten Angreifer in Minuten zu öffnen sind, nachdem er zuende gestaunt und fertig gelacht hat.

Nachdem ein Angreifer das Server-System übernommen hat, ergeben sich natürlich noch ganz andere Möglichkeiten. So kann der Angreifer z.B. "strings" über die Anwendung laufen lassen und dabei das Datenbank-Paßwort im Klartext auslesen. In der Datenbank stehen dann die ISDN-Rufnummern, die das System annimmt und zurück ruft. Ein leichtes also, sich hier als Angreifer einzutragen und dann die kompromittierte Arztpraxis nicht mehr zu benötigen. Der Vollständigkeit halber sei noch gesagt, daß die sich auf dem Server einwählenden Clients üblicherweise Windows 98 Systeme sind, die noch leichter zu öffnen sind als die Server. Viele Ärzte werden da vermutlich auch direkt ihre File Shares drauf haben und die Patientendaten direkt per SMB veröffentlichen, zum einfacheren Zugriff im LAN. Der Einsatz von Firewalls ist jedenfalls nicht vorgesehen im Konzept, haben wir uns versichern lassen. So etwas bräuchte man nicht, das sei ja alles sicher und überhaupt sei das Konzept mit Datenschutzbeauftragten zusammen erstellt und von denen abgesegnet worden.

Zusammenfassung

Wir hatten bisher noch keinen Erfolg beim Versuch, uns ein solches Testsystem zur eigenen Begutachtung zu beschaffen. Das ist allerdings geplant. Wir sind gespannt, was eine eigenhändige Analyse des Systems noch so alles ergeben wird.

Was wir bisher von dem System gehört haben, erinnert an eine Schlucht aus einem Indiana Jones Film, über die eine kleine wackelige Hängebrücke montiert wurde, und unser einheimischer Führer teilt uns mit, daß alles sicher sei, man solle einfach nicht nach unten gucken.

Das PaDok-System ist lediglich das Backend für das Gesundheitskarte-Gesamtsystem. Das ist zweifelsohne der wichtigste Teil, aber wenn hier schon derartig gepfuscht wird, was haben wir dann von dem Frontend-Teil zu erwarten? Wie sicher ist die Gesundheitskarte selber? Kann unser marodes Gesundheitssystem diese zusätzliche Belastung durch selbst erhackte E-Rezepte verkraften?

Auch wenn man das Risiko für die Durchführung der skizzierten Angriffe für eher gering hält, ergibt sich doch die beunruhigende Tatsache, daß korrupte Ärzte angesichts solcher Infrastruktur-Mißstände leichtes Spiel haben, weil sie sich immer auf Hacker im System berufen können.

Angesichts dieser katastrophalen Umstände empfiehlt der CCC ausdrücklich, die Gesundheitskarte noch ein paar Jahre nach hinten zu verschieben, und die bereits bestehenden Komponenten und zukünftigen Pläne einer anständigen Analyse von neutraler Seite zu unterziehen. Immerhin wird hier mit Steuergeldern hantiert. Gerade in Zeiten von Hartz IV sind solche Verschwendungen (in PaDok und D2D sollen dreistellige Millionenbeträge geflossen sein) nicht zu rechtfertigen. Dafür hätte man hunderttausenden von sozial Schwachen helfen können, anstatt ihnen jetzt ihre Bezüge zu streichen.

[1] *http://www.kvno.de/mitglieder/d2d/faq_tech.html*

Die Gefahren der Softwarepatente

von Alexander Bernauer <alexander.bernauer@ulm.ccc.de>

Der CCC Ulm hat zwei Chaosseminare zum Thema Softwarepatente veranstaltet. Am 13. Oktober 2003 hat Christian Cremer, Patentanwalt aus Neu-Ulm, das Patentsystem allgemein und den Übergang zu Softwarepatenten aus seiner Sicht als Patentanwalt dargestellt.

Am 4. November 2004 hat Richard M. Stallman [1] einen Vortrag über die Gefahren der Softwarepatente gehalten [2]. Stallman ist als Gründer der Free Software Foundation [3] und der GNU Bewegung [4] besonders angagiert im Kampf gegen Softwarepatente. Jedoch wird nicht nur freie Software von Softwarepatenten bedroht, sondern fast alle, die im Bereich Informationstechnik tätig sind, und im Grunde auch jeder Endanwender. Dieser Artikel greift die Inhalte und die Argumentationen aus beiden Vorträgen auf und gibt einen Überblick über Softwarepatente und ihre Konsequenzen.

Ein Patente ist ein Monopol auf die Verwendung einer Idee. Softwarepatente sind damit Monopole auf die Verwendung von Ideen, die man in Software einsetzen könnte. Wichtig dabei ist, dass sich ein Softwarepatent sich nicht auf ein komplettes Programm bezieht, sondern auf eine Idee, die in vielen verschiedenen Programmen eingesetzt werden könnte.

Die ursprüngliche Idee des deutschen Patentsystems war, Erfindern einen Investitionsschutz zu gewähren und dafür im Gegenzug das neue Wissen für die Allgemeinheit zu erhalten. Jemand forscht an einer neuen Sache und veröffentlicht die Ergebnisse. Im Gegenzug bekommt er ein Monopol zur wirtschaftlichen Ausbeutung seiner Ergebnisse, um seine Investitionen für die Forschung wieder rein zu holen und Gewinn mit seiner Arbeit erzielen zu können. Nach einer gewissen Zeit läuft das Patent aus und das Wissen wird zum Allgemeingut. Der gewünschte Effekt des Patentsystems ist also die Förderung von Investitionen in die Forschung und damit die Förderung von Innovationen.

Damit eine Idee in Deutschland patentierbar ist, muss sie vier Kriterien erfüllen.

• Neuheit: Nach dem absoluten Neuheitsbegriff muss unter Berücksichtigung des allgemeinen Fachwissens zum Prioritätszeitpunkt die Erfindung neu sein. Das bedeutet auch, dass es mehrere Patente auf die selbe Idee geben kann.

• Schöpfungshöhe: Unter Berücksichtigung des Standes der Technik und den Fähigkeiten des Durschnittsfachmanns muss es sich um eine erfinderische Tätigkeit handeln. Das bedeutet insbesondere, dass die aufgezeigte Lösung nicht naheliegend sein darf. Trivialpatente sind damit ausgeschlossen.

• gewerbliche Anwendung: das Erfundene muss seiner Art nach geeignet sein, um in einem technischen Gewerbebetrieb hergestellt oder angewendet zu werden.

• Technizität: die Erfindung muss technisch sein. Die gültige Definition von Technizität ist: planmäßiges Handeln unter Einsatz von beherrschbaren Naturkräften zur Erzielung eines kausal übersehbaren Erfolges ohne menschliche Verstandestätigkeit zwischenzuschalten, wobei der kausal übersehbare Erfolg die unmittelbare Folge des Einsatzes beherrschbarer Naturkräfte ist.

Patente auf Mathematische Methoden, Pläne und Verfahren für geschäftliche Tätigkeiten und Programme für Datenverarbeitungsanlagen sind zusätzlich explizit ausgeschlossen.

Um für eine Idee einen Patentschutz in Deutschland zu erlangen muss man eine Anmeldung beim zuständigen Patentamt abgeben und eine Gebühr bezahlen. Nach einer formalen Überprüfung der Anmeldung folgt die Recherche zum Thema und die Offenlegung der originalen Anmeldeunterlagen. Anschließen kommt es zur materiellen Prüfung in der sich ein technischer Patentprüfer mit den Ergebnissen der Recherche beschäftigt und die Patentierbarkeit der Idee überprüft. Er entscheidet dann, ob das Patent erteilt wird, oder nicht. Wird es erteilt, so kann jeder dritte ein Einspruchsverfahren starten, in dem er eigenes Material zur Erneuten materiellen Prüfung vorlegt. Das kann dazu führen, dass die Erteilung des Patentes zurückgezogen wird.

Nach einer gewissen Frist erlischt die Möglichkeit, ein Einspruchsverfahren anzustreben. Es bleibt jedoch die Möglichkeit, ein Nichtigkeitsverfahren anzustreben, bei dem sich ein Gericht mit der Gültigkeit eines Patentes beschäftigt.

Patente werden oft mit Copyright oder Urheberrecht in einen Topf geworfen [5]. Das ist aber gänzlich falsch. Das Urheberrecht schützt die Arbeit des Erfinders, also eine konkrete Umsetzung einer Idee. Ein Patent bezieht sich aber auf die Idee selber und somit auf alle möglichen Arbeiten, die diese Idee umsetzen. Wenn jemand ein Programm schreibt, so kann er sicher sein, kein Urheberrecht zu verletzen, wenn er nichts kopiert hat. Nicht zu kopieren heißt aber nicht, dass man sicher kein Patent verletzt. Ein weiterer wichtiger Unterschied ist, dass der Schutz des Urheberrechts auf eine Arbeit automatisch wirksam ist, während man für eine Patentanmeldung drei Dinge braucht: Geld, Zeit und das Glück, es als Erster angemeldet zu haben.

Im Lauf der Jahre hat sich die Praxis des Patentwesens von der anfänglichen Idee wegbewegt und widerspricht mittlerweile den Grundsätzen von damals. Klar ist, dass man als Unternehmen zwar den Patentschutz für eine Idee haben, der Konkurrenz aber trotzdem so wenig Wissen wie möglich darüber vermitteln möchte. Das führt dazu, dass die Patentschriften mit Absicht nur so viel über die Idee aussagen, um im Zweifelsfall mit ein wenig Interpretationshilfe ein Gericht überzeugen zu können und sonst keine weiteren Details oder gar Hilfestellungen für einen Leser beinhalten. Die Kunst beim Formulieren von Patentschriften, diesen schmalen Grat nicht zu verlassen, ist das Handwerk der Patentanwälte. Damit wird der Vorteil eines Patentes für das Allgemeinwohl, die Erhaltung von Wissen über Neues, geschwächt.

Die schwammige Formulierungen in Patentschriften führen dazu, dass sich ein Patenprüfer schwer tut, einen Antrag zu bearbeiten. Verständlich, dass sich ein Einzelner schwer tut, innerhalb kurzer Zeit zu entscheiden, ob alle Kriterien für die Patententierbarkeit erfüllt sind. Da es für einen Patentprüfer mehr Arbeit bedeutet, ein Patent abzulehnen, wird er es im Zweifelsfall eher akzeptiert. Die Konsequenz sind Trivialpatente, mehrere Patente auf die selbe Idee und Patente auf bereits bekannte Ideen, wofür es jeweils genügend Beispiele gibt, wie z.B. ein Patent auf das Rad, das 2001 in Australien erteilt wurde [6].

Das Kriterium "Technizität" verhinderte bislang die Patentierbarkeit von Software. Natürlich ist eine exakte Trennung zwischen Softwarepatent und technischem Patent nicht einfach. In der Grauzone bewegt sich z.b. Software, die als zusätzlichen Seiteneffekt irgendwo ein Lämpchen blinken lässt. Da ein technisches Merkmal genügt, ist damit die Hürde der Technizität genommen. So gibt es mittlerweile eine Reihe von Patenten dieser Bauart, die sich eigentlich auf Software beziehen. Dazu kommt jetzt, dass man das Technizitätskriterium so erweitern möchte, dass es offiziell möglich ist, Software zu patentieren. ...

Falls Softwarepatente eingeführt werden, wird sich ein Programmierer überlegen müssen, was zu tun ist, um sicher kein Patent zu verletzen. Man müsste alle Ideen, die in der Software Verwendung finden, auflisten und einzeln auf die Existenz von Patenten abprüfen. Das ist aber im Allgemeinen unmöglich. Zunächst einmal ist es nicht leicht, wirklich alle Strukturen und Ideen einer Software zu finden. Das liegt daran, dass man ein Programm auf viele verschiedene Arten strukturieren kann. Zum Beispiel könnte man die Granularität bei der Analyse verändern und bekäme damit ein ganz anderes Bild von Einzelkomponenten und ihrem Zusammenspiel. Das Erstellen einer erschöpfenden Liste aller verwendeter Ideen erfordert eine viel höhere mentale Fähigkeit als das Schreiben des Programms.


Richard Stallman

Angenommen man hätte aber tatsächlich die vollständige Liste der verwendeten Ideen: dann wäre es immer noch unmöglich, für jede Idee herauszufinden, ob sie patentiert ist. Das liegt zum einen an den sogenannten schwebenden Patenten. Das sind Anmeldungen auf Patente, die zwar schon eingereicht aber noch nicht veröffentlich wurden. Es kann also sein, dass sich erst später herausstellt, dass man ein Patent verletzt, ohne dass man eine Chance gehabt hätte, das im Vorfeld in Erfahrung zu bringen. Beispiel dafür ist die LZW Komprimierung, die im GNU Programm compress ein-

gesetzt werden sollte. Kurze Zeit vor dem Release der Software wurde ein US Patent auf diesen Algorithmus erteilt. Selbst wenn man ein eigenes Patent für die Idee beantragt, bringt das in diesem Fall nichts, weil bei einem Konflikt immer derjenige den Zuschlag vom Patentamt bzw. vor Gericht erhält, der seinen Antrag als erster eingereicht hat. Ein weiteres Problem ist die Masse der existierenden Patente. Es sind so viele, dass das Durchforsten aller möglicherweise relevanten Patente bei weitem länger dauert, als das Schreiben der Software selber. Hinzu kommen die schwammigen und unklaren Formulierungen in Patenttexten, die die TTU (Time-To-Understanding) massiv verlängern. Stallman erwähnte zum Beispiel ein US Patent auf topologische Sortierung, in dessen Patenttext der Begriff "topologische Sortierung" kein einziges Mal auftaucht.

Es bleibt einem als Programmierer also nichts weiter übrig, als zu hoffen, kein Patent zu verletzen. Alternativ bleibt die Hoffnung, dass es niemandem auffällt, wenn man doch eines verletzt. Richard Stallman beschreib das bildhaft als "mit verbundenen Augen durch ein Minenfeld laufen (mit dem Unterschied, dass Minen nur einmal explodieren)".

Falls man trotz aller Vorsicht ein Patent mit seiner Software verletzt, stellt sich die Frage, was man tun kann. Erstens kann man versuchen, die Verwendung der patentierten Idee zu vermeiden. Das kann einfach sein, es kann aber auch beliebig schwer werden. So gibt es zum Beispiel ein US Patent auf die Fast Fourier Transformation (FFT). Wenn man eine Software geschrieben hat, die die FFT verwendet, so kann man vielleicht auf die normale Fourier Transformation zurück greifen, da diese nicht patentiert ist. Wenn die Software aber ohnehin schon selbst auf den größten Maschinen am Resourcenlimit kratzt, so ist sie unbrauchbar, wenn man keine FFT einsetzen kann. Ein anderer Fall, in dem man nicht auf die Verwendung einer Idee verzichten kann, sind Spezifikationen, die die Verwendung bestimmter patentierter Algorithmen vorschreiben. Beispiele dafür sind PostScript und GIF. Die Implementierung gemäß der Spezifikation kann man vielleicht nicht umgehen, weil es sich um einen de-facto oder gar öffentlichen Standard handelt. Der Sinn von Software ist ja, dass sie benutzt wird. Wenn einer Software die Unterstützung vieler Standards fehlt, so wird sie keine Akzeptanz bei Benutzern finden und ihre Entwicklung damit sinnlos werden. Das gleiche gilt für Elemente der grafischen Benutzerschnittstelle, an die sich die Benutzer so gewöhnt haben, dass es schwer fällt, einen Ersatz zu finden, der das selbe vermittelt und vom Benutzer akzeptiert wird. Ein Beispiel dafür ist der Fortschrittsbalken. Ein Problem hat man auch, wenn man es mit einem Patent zu tun hat, das eine ganze Familie von Ideen abdeckt. Ein Beispiel dafür sind US Patente auf Public-Key-Kryptographie. Hier kann man nicht auf die Verwendung der Idee verzichten, weil die Wissenschaft keine Alternativen kennt.

Eine zweite Möglichkeit im Falle der Verletzung eines Patents ist es, eine Lizenz für die Nutzung zu erlangen. Diese Lizenz stellt grundsätzlich der Inhaber des Patents aus. Er ist dazu aber nicht verpflichtet. Auch für die Nutzungsbedingungen gibt es keinen Rahmen, so dass man der Willkür des Inhabers ausgeliefert ist. Zusätzlich ist es so, dass man vielleicht mehr als eine Lizenz kaufen muss, weil mehr als ein Patent verletzt oder mehrere Patente auf die selbe Idee existieren.

Letzteres darf zwar gemäß Patentrecht nicht sein, kommt in der Realität aufgrund der schwammigen Formulierungen in den Patenttexten aber durchaus vor. Die Kosten für Lizenzen sind damit ein unkalkulierbares Risiko für Unternehmen. Erfahrungen aus den USA zeigen, dass wenige Lizenzen genügen, um den Business Plan eines Unternehmens zu sprengen und damit seinen Tod zu bedeuten.

Als letzte Möglichkeit bleibt ein Nichtigkeitsverfahren anzustreben. Man müsste dafür zeigen, dass eines der Kriterien für dieses Patent zum Vergabezeitpunkt nicht erfüllt war. Außer dem Neuheitskriterium sind alle anderen subjektiver Natur, so dass ein Gericht im Allgemeinen wahrscheinlich die damalige Entscheidung des Beamten respektieren wird. Es bleibt also nur die Möglichkeit, Beweise dafür zu finden, dass die Idee damals nicht neu war. Das kann in Einzelfällen gelingen. Im Allgemeinen gibt es jedoch selten beglaubigte Dokumente, die den Einsatz einer Idee zu einem bestimmten Zeitpunkt bestätigen. Selbst wenn man den Prozess gewinnt, so hat man viel Geld und Zeit verloren, was wieder die meisten Business Pläne sprengen dürfte.

Alle drei Möglichkeiten können gangbare Wege sein. Im Allgemeinen wird man jedoch selten das Glück haben, für jedes Patent, das man verletzt, einen der Wege gehen zu können und dabei insgesamt zu überleben. Scheitert man ein Mal, ist das Projekt tot. Und dabei ist im Vorfeld nicht abzusehen, mit wie vielen Forderungen von Patentinhabern man konfrontiert werden wird. In der Konsequenz werden sich weniger Unternehmer auf dieses unkalkulierbare Spiel einlassen, so dass die Einführung von Softwarepatenten für die Softwarebranche weniger Investition und damit weniger Innovation bedeuten würde. Beides Dinge, die das Patentwesen eigentlich fördern wollte.

Softwarepatente haben jedoch nicht nur Nachteile, sonst würde es niemanden geben, der ihre Einführung fordern würde. Der Vorteil von Patenten ist im Allgemeinen, dass man Geld durch die Vergabe von Lizenzen verdienen kann. Außerdem hat man ein Instrument zur Kontrolle des Marktes, weil man Konkurrenten beliebig ausschalten kann, wenn man ein Patent auf eine Schlüsseltechnologie besitzt. Dafür braucht man aber Patentanwälte, die bezahlt werden wollen. Außerdem kann es ebensogut sein, dass ein Konkurrent ein Patent auf eine Schlüsseltechnologie besitzt, was für das eigene Unternehmen negative Konsequenzen haben kann.

Der Nachteil von Softwarepatenten überwiegt also den Vorteil deutlich, so dass sich die berechtige Frage stellt, wie sich das überhaupt für irgend ein Unternehmen rechnen kann. Der Antwort darauf lautet: Lizenzaustausch. Die Idee ist folgende: man hat stapelweise Patente, so dass man einen Patentinhaber im Falle einer Verletzung seines Patentes seinerseits wegen Patentverletzung verklagen kann. Das funktioniert, weil sich in der Menge der schwammig formulierten Patente im Normalfall immer mindestens eines finden lässt, gegen das der andere mit etwas Interpretationshilfe verstößt. Diese Situation der gegenseitigen Forderung lässt sich dann einfach auflößen, in dem man sich gegenseitig Lizenzen zur Verwendung des Patentes gibt.

Die Ameldungen und die Verwaltung der Patente, das Finden von günstigen Patenten, die sich für Forderungen gegenüber andere eignen sowie das Aushandeln und Verwalten der Lizenzen ist ein sehr großer Aufwand, für den ein Unternehmen im Normalfall eine eigene Abteilung mit Patentanwälten braucht. Es ist klar, dass sich das nicht jedes Unternehmen leisten kann. Daher sind es meistens Megakonzerne, die diese Strategie verfolgen. Es folgt, dass man nur Zugang zu gewissen Technologien bekommt, wenn man zum Kreis der Megakonzerne gehört und sich damit den Zugang erzwingen kann. Der Nachteil der Softwarepatente, das unvorhersagbare Belangen durch Patentinhaber, ist somit kompensiert und es bleiben die Einnahmen durch Lizenzen und die Kontrolle des Marktes.

Vor allem in der Informatikbranche ist die Kontrolle des Marktes ein wichtiger Punkt. Zum einen gibt es fast monatlich neue Technologien und Ideen, die in ein Produkt implementiert Erfolg am Markt bringen können. Diese Schnellebigkeit der Branche macht es aber für marktkontrollierende Unternehmen schwer, den Überblick zu behalten. Außerdem braucht es im Gegensatz zum Ingenieurswesen viel weniger Startkaptial, um ein Unternehmen zu gründen und mit Hilfe einer guten Idee Erfolg am Markt zu erlangen. Im Zweifelsfall tun es ein paar PCs aus dem nächsten Supermarkt und eine Hand voll Entwickler. In der Konsequenz gibt es ständig neue Unternehmen, die aus dem Nichts hervortreten und den Etablierten den Markt streitig machen. Mit Hilfe von Softwarepatenten wäre die Gefahr für Megakonzerne gebannt.

Ironischerweise lässt sich das am besten an Hand einer Argumentation von Befürwortern zeigen, und zwar am Mythos des 'Starving Genius' (verhungerndes Genie): Gegeben ein Genie, das eine tolle Erfindung hat und sich jahrelang in seiner Garage einschließt um die Idee zu implementieren. Wenn er damit fertig ist, gründet er ein Unternehmen, um die Früchte seine Arbeit zu ernten. Jetzt kommen aber die etablierten Konzerne der Branche, klauen ihm die Idee und drücken mit ihren Kapitalreserven die Marktpreise. Das arme Genie kann diesem Konkurrenzdruck nicht Stand halten, weil ihm das Kapital fehlt. Er geht zu Grunde. Das Patentsystem soll ihn jetzt vor den Großen schützen, da diese seine patentierte Idee nicht klauen dürfen und er somit in Ruhe sein geniales Produkt kommerziell verwerten kann. Hört sich gut an.

Doch die Realität ist eine andere. Das Genie wird zwar - mit etwas Geld und Geduld - ein Patent auf seine Idee erhalten, und ein Konkurrent wird dann seine Idee nicht klauen dürfen, da er sonst das Patent verletzt. Im Gegenzug jedoch verletzt das Genie mit seiner Software sehr wahrscheinlich gleich mehrere Patente von Megakonzernen. Das liegt ganz einfach daran, dass die Konzerne so viele schwammige Patente besitzen, dass sich mit hoher Wahrscheinlichkeit einige finden lassen, die sich auf irgend einen Teil der Software des Genies beziehen. Das Genie sieht sich somit mit einer Gegenforderung konfrontiert. Als einzigen Ausweg bietet die Konkurrenz einen Lizenzaustausch an. Wenn das Genie es nicht schafft, die Forderungen abzuwehren ohne dabei finanziell zu Grunde zu gehen, ist sein Unternehmen tot. Geht er auf den Lizenzaustausch ein, gibt er sein Monopol auf seine Idee auf, so dass der Megakonzern nun doch mit ihm auf dem Markt konkurrieren kann. Beide Fälle sollte das Patentsystem eigentlich verhindern.

Eine anderes Argument von Befürwortern ist, dass es in anderen Branchen auch Patente gibt. Warum nicht in der Informatik? Auch hier wollen Unternehmen Investitionsschutz. Wo ist der Unterschied? Grundsätzlich gilt es hier zu bedenken, dass diese Argumentation nur greift, wenn man das Patetensystem an sich als etwas Positives anerkennt. Ohne die Grundlagen des Patensystem hier diskutieren zu wollen, wirft zumindest die gängie Praxis schwere Zweifel am Vorteil für die Allgemeinheit auf. Doch selbst wenn die Praxis den eigentlichen Ideen des Patentsystems folgen würde, gibt es grundsätzliche Unterschiede zwischen der Informatik und Ingenieursbranchen, so dass die Veraussetzungen für ein Patentsystem mit Vorteil für die Allgemeinheit nicht gegeben sind.

Software ist Mathematik. Eine Software läuft auf einer universellen Maschine mit einer Spezifikation. Diese Spezifikation ist eine mathematische Beschreibung der Funktionalität der Maschine. Die Anwendung dieses universellen Machine, ein Programm, ist damit eine Sammlung von mathematischen Beschreibungen. Es liegt in der Natur von mathematischen Beschreibungen, dass sie umformuliet werden können, ohne ihre Aussage zu ändern. Der Zusammenhang zwischen zwei mathematischen Beschreibungen für die selbe Aussage ist im Allgemeinen nicht offensichtlich. Deshalb kann es einem Patentprüfer einfach passieren, dass er ein Patent auf eine Idee vergibt, die eigentlich schon patentiert ist; noch eher als bei technischen Patenten, wo das selbe allein auf Grund der schwammigen Formulierungen in den Patenttexten in der Vergangenheit schon geschehen ist. "Ein Patent pro Idee" wird sich also für Softwarepatente kaum halten lassen, was jedoch ein zentraler Aspekt eines Patentsystem ist.

Viele Probleme eines Ingenieurs stellen sich für einen Informatiker gar nicht: Resonanzfrequenzen, EMV, Verschleiß, Robustheit gegenüber Umwelteinflüssen, usw. Vor allem gibt es in der Informatik keine Diskrepanzen zwischen den Modellen und der Realität. Wenn man annimmt, dass Informatiker und Ingenieure im Schnitt die selbe Intelligenz besitzen, dann folgt daraus, dass ein Software viel komplexer aufgebaut werden kann, bevor die Probleme darin die mentalen Fähigkeiten des Entwicklers überschreiten. Je komplexer ein System ist, desto mehr Einzelkompontenten sind im Allgemeinen darin verbaut. Deshalb verletzt eine typische Software gleich mehrere Softwarepatente, was u.a. dem starving Genious das Genick bricht.

Alle oben beschriebenen Probleme mit der gängigen Praxis bei Patenten würden sich bei Softwarepatenten ergeben, selbst wenn die gängige Praxis sich an die Ideen des Patentsystems halten würde. Realistisch betrachtet werden sich also die Probleme mit Patenten für die Informatikbranche verschärfen. Eine geschlossene Gemeinschaft von Megakonzernen wird sich bilden, die sich den Markt aufteilen und gemeinsam kontrollieren. Für den Endanwender bedeutet das, dass es kaum Alternativen zu einer Software geben wird, und man sich mit seinen Wünschen komplett in der Hand und damit in der Willkür einzelner Unternehmen befindet.

Die Investitionskosten in der Informatikbranche sind im Vergleich zu denen aus Ingenieursbranchen sehr viel kleiner. Das liegt daran, dass man keine Labore und Produktionshallen mit vielen Auflagen, teuere Messgeräte, Maschinen und Spezielwerkzeuge braucht. Der Inverstitionsschutz ist deshalb nicht so zentral wie in anderen Branchen. Deshalb stellt sich die Frage, ob es hier auch legitim ist, dafür die Freiheitsrechte anderer zu beschränken und staatlich durchzusetzen, was das Patentsystem tut.

Es wird argumentiert, dass es falsch wäre, wenn sich andere an den Früchten der Arbeit eines Erfinders bereichern. Deshalb muss ein Patentsystem her, auch in der Informatik. Das ist aber falsch. Man tut dabei so, als ob die geniale Idee aus dem Nichts entstanden wäre, ohne jegliches Vordenken und ohne Inspiration durch Ideen anderer. Das ist aber fast nie der Fall. Warum darf der Erfinder alles Vorwissen umsonst nehmen um es dann, wenn er es mit einer Idee von sich angereichert hat, auf einmal den anderen vorzuenthalten? Das ist nicht konsequent. Man müsste alle seine Vordenker entlohnen, und eigentlich deren Vordenker auch. Es ist klar, das das nicht praktikabel ist, weshalb es auch niemand ernsthaft fordert. Doch eigentlich ist es der einzig richtige Weg, wenn man die Entlohnung von Erfindern verlangt. Deshalb ist diese Forderung falsch.

Richard Stallman vergleicht Software mit Musik. Beides ist eine Komposition von bekannten Einzelteilen und bei beiden besteht die Kunst darin, bekannte Elemente so zusammenzusetzen, dass schöne Musik bzw. gute Software entsteht. Ab und zu findet jemand ein neues Element das er in ein Gesamtwerk aus sonst bekannten Elementen einbettet. Angenommen, es gäbe Patente auf Musik. Wie genial hätte dann ein Mozart oder ein Bethoven sein müssen, um Musik zu komponieren, die erstens von den Leuten akzeptiert wird und zweitens keine bis dahin bekannten Elemente verwendet? Da der Sinn von Musik ist, gehört zu werden, würde ein Komponist mit diesem Problem konfrontiert werden. Genau das selbe Problem hat ein Programierer, der Software schreiben will, die vom Benutzer verwendet wird. Es ist einfach nicht möglich, Software zu schreiben, ohne dabei bekannte Ideen zu verwenden.

[1] *http://www.stallman.org/*
[2] *http://ulm.ccc.de/chaos-seminar/rms/video-de.html*
[3] *http://www.free-soft.org/*
[4] *http://www.gnu.org/*
[5] In Deutschland gibt es kein Copyright Gesetz, sondern das Urheberrecht. Die Unterschiede sind für die Argumentation aber bedeutungslos.
[6] *http://www.ipmenu.com/archive/AUI_2001100012.pdf*

Teaching Hacking -

Most universities teach information security - if at all - in a defensive way. Often things like intrusion detection systems, firewalls and encryption are covered, but topics like buffer overflows or forensics are forgotten. To change this, the Laboratory for Dependable Distributed Systems at RWTH Aachen University offered a Summer School on applied IT security and this article will present our experiences.

If you take a look at the courses offered at major universities in the world, you will notice that most of them do not care about practical security, aka hacking. Instead, they teach theoretical foundations (e.g. encryption and hash functions) or defensive approaches like intrusion detection systems, firewalls and virtual private networks. In our view, this leaves computer science graduates ill-prepared for their job because they lack hands-on knowledge on how to deal with security technology or they always trail active adversaries in their efforts to master them. To understand the importance of information security, students should have the possibility to gain practical experience how security systems fail, using offensive techniques. And hacking is fun, so teach it :-)

In summer term 2004 the Laboratory for Dependable Distributed Systems at RWTH Aachen University offered a novel course in the area of information security: The Summer School "Applied IT Security" gave graduate and Ph.D. students an insight into common techniques in the area of information security. During a three week period, we covered topics like exploiting of programming errors, wireless (in-)securities and malware.

Because some people from the CCC in Cologne and Berlin were involved - either as lecturer or attendee - we want to present our experiences and give an insight into the course. At first we want to give an overview over related courses at other universities and afterwards we present in detail our Summer School.

[Hacking at other universities]

There have been some good attempts to incorporate practical elements of information security into university degree courses. For example, the computer science department of Darmstadt University of Technology, Germany, regularly runs a so-called "Hacker Contest" for several years.

The Hacker Contest is similar to a Linux Deathmatch, as organized at some Congresses and the Camp: Students form teams that have to set up systems and then use common exploitation techniques to attack the systems of the other teams. The attendees also have to analyze attacks to their own systems and increasingly deploy stronger defense measures. The University of Magdeburg and our Lab also offer a similar lab course.

There are some universities that offer courses in which the students learn the underlying concepts of offensive IT security and can also apply their knowledge. As an example, the Distributed Systems Group of the Technical University of Vienna offers courses on Internet security: Practical aspects of IT security like race conditions, viruses and reverse engineering are covered and exercises in which the students have to implement the attacks must be solved.

http://www.auto.tuwien.ac.at/~chris/teaching/inetsec2.html
http://www.infosys.tuwien.ac.at/teaching/courses/InetSec2/index.html

"Wargames" and "Capture the Flag" (CTF) contests are also offered at some universities. Most famous is probably the annual CTF contest of the UCSB, in which several educational institutions spread across the world battle against each other. This year, a team from our Lab has also competed against the other teams. Surprisingly, we managed to achieve the second place :)

The Information Technology and Operations Center (ITOC) at the U.S. Military Academy West Point has a curriculum which also teaches offensive information security techniques. ITOC organizes a yearly "Cyber Defense Exercise" which has similarities to the Capture the Flag contests. U.S. authorities with an information security education branch like the United States Military Academy, the United States Air Force Academy, and the Naval Postgraduate School, participate in these exercises. Machines maintained by the participants are attacked by the NSA 92nd Aggressor Squadron -- Land Information Warfare Activity, over the course of several days and participants have to counter these attacks.

[Summerschool]

After pointing out some other courses in the area of information security, we now want to give an overview over the Summer School, especially on the schedule during these three weeks.

The staff organizing the Summer School consisted of two research assistants of the Laboratory and two students of whom one is a research student of the laboratory and the other a regular student at another university. Three of them are or were involved with the CCC in Cologne.

The intended audience were students and young scientists from Europe with a profound interest not only in information security but also in offensive techniques in relation to security. Applicants had to fill out a questionnaire and we chose fifteen people. Most of them study at RWTH Aachen University, but we had also four attendees from the University of Cambridge (three of them from Ross Anderson's security group) and two from TU Berlin (also members of CCC Berlin). Furthermore, someone from netric.org, a CCC Cologne member and a student from the University of Applied Science in Gelsenkirchen attended. So a motley crew of people - partially with profound knowledge - participated at the Summer School.

At our Lab, we had to prepare a few things beforehand: To foster informal communication between students, we converted an office to a coffee hall with sofas, an overall relaxed ambiance, and a choice of snacks and refreshments. A lab room was converted to some kind of hackcenter, most exercises were hosted in this room. Furthermore, there was a plain room for temporary use by students which had the feeling they need concentration for specific tasks. The housing was left to the students themselves and their choices showed a very wide variety of solutions ranging from the universities' guest-house to the local camping ground. Three week long camping is hard, but they survived it without problems.

Lectures started at a quarter to nine in the morning (due to the Verpeilungsfaktor, most lectures began later...) and covered two topics until noon. After lunch, the lab session started, during which students applied the techniques learned in the lectures and developed them further. For an outline of the Summer School see the following table.

The lab session was interrupted by a so called "coffee table talk": We invited external contributors to give a short statement related to a topic of their interest or

Day	Lecture1	Lecture2	Lab	Day	Lecture1	Lecture2	Lab
I-1	Introduction	Hardware Security	Hardware Wargames	II-4	Introspection	Projects	Projects
I-2	Web Applications	Web Applications	Web Applications	II-5	Projects	Projects	Projects
I-3	Buffer Overflows	Other Programming Errors	Exploiting Overflows	III-1	Misc. Forensics	Disk Forensics	Forensics
I-4	Advanced Exploitation	Networking	Network Programming	III-2	Disk Forensics	Disk Forensics	Forensics
I-5	Sniffing: Layer 1&2	Spoofing & (D)Dos	Sniffing & Spoofing	III-3	Malware Unix	Unix infection	Honeynets
II-1	Network Topology	Application Fingerprinting	Network Mapping	III-4	Excursion	Excursion	Excursion
II-2	Bluetooth	Wireless Attacks	Wardriving	III-5	Wargame	Wargame	Wargame
II-3	Hidden Data	Honeynets	Wardriving				

Table: Schedule of the Summer School

a short introduction one some project they work on. The coffee table talks were intended to broaden the view of the participants. We aimed at showing the students problems being worked on in the real world and to teach them looking at problems not only from the security perspective.

Speakers were sent by corporations ranging from Microsoft and Pixelpark up to TÜV Rheinland and a major German Bank. But we had also participants from other groups like the CCC or the "Verein digitale Kultur", which covers artistic expression via digital means, and various academic research groups. Topics covered in the coffee table talk ranged from computer-ethics, civil liberties in relation to the Internet, infos about phishing attacks, up to technical subjects like heap-based overflows and XML security.

A typical day ended with a meeting: usually around six in the evening everybody presented his work of the day. But students often stayed through large parts of the night to develop their projects further.

[[Running the School]]

[[[First week]]]
On the first day (September 20) we started with a brief introduction and covered "Hardware Hacking". The students got an insight into limitations and vulnerabilities of hardware devices. During the lab session, the students opened some devices (e.g. cable modems, router, switches) to inspect the insides. They also played some wargames like www.hackthispage.tk and averaged at about five levels.

The topic of the second day was "Attacks Against Web Applications": One lecture concentrated on general problems in web applications (e.g. SQL injection and cross site scripting), while the other had the special focus on programming errors on web-applications written in PHP. The corresponding lab session had no very tight focus. The students should find bugs in real web applications. In the afternoon, George Danezis, one of the participants and research assistant at the university of Cambridge, gave a talk on "Anonymous Communication".

On Wednesday, the lectures concentrated on buffer overflows and other programming errors. This is a very broad topic and probably two lectures are not enough to cover the whole field. Therefore, these two lectures concentrated on the basics and two other coffee table talks deepened the knowledge of the participants. The lab session was quite popular at this day: The students should exploit various programs and they apparently had much fun doing this. One student also found some bugs in real applications (e.g. a format-string vulnerability in Tor, an anonymizing overlay network for TCP) and further examined them. In cooperation with the authors of the software, these holes are now fixed - so no 0day for you now :).

"Phishing" was the topic of the coffee table talk and an IT-security specialist of a large German bank explained the threat.

A lecture on "Advanced / Automatic Exploitation" showed the participants some techniques and tools that attackers can use, e.g. the Metasploit Framework, search engines like Google to collect information, etc. In addition, the art of fuzzing, a technique to find errors in a given program in a semi-automated fashion, was presented. The second lecture repeated necessary knowledge on communication networks and gave an introduction to network programming and the important libraries (e.g. libnet, libpcap). Implementation of covert channels, ARP-spoofing and various other tools were the focuses of the lab session. Jens Ohlig from the CCC gave a historical overview about political activism and hackers during the coffee table talk.

On Friday, sniffing & spoofing and (distributed) denial-of-service (DDoS) attacks were covered. The lectures gave an overview of tools and techniques used to sniff passwords and other sensitive information on networks. The lectures also explained how to spoof packets in order to receive interesting packets and gave a background on (distributed) denial-of-service attacks.

During the lab session, the student experimented with the available tools and also implemented some small programs for ARP-spoofing and similar techniques. The coffee-table talk entitled "XML Security" was given by Christian Geuer-Pollman from the European Microsoft Innovation Center (EMIC), Aachen.

[[[Second week]]]

The next week started with a lecture on network reconnaissance. This lecture covered techniques to find useful information about targets and first steps of an attacker after a successful compromise. The second lecture focused on further steps after an intrusion and covered application fingerprinting in depth. The students were encouraged to do portscans of the network of our university during the lab session. The center for computing and communication (CCC) explicitly allowed us to do this and we found some security holes during these tests: One student found several routers with default passwords and another tracked down some printers that could be managed remotely via a web-interface. Penetration-testing for phone-systems was the topic of the coffee-table talk on this day. Rolf von Stein from TÜV Secure IT explained the motivation behind this and gave some background information.

Wireless Security was the focus on Tuesday: The first lecture with the title "Bluetooth Security" introduced the attendees to the basics of the Bluetooth standard and pointed out some attacks. The focus of the second lecture was on WLAN (mainly 802.11b) and also a small introduction to RFID and its concerns was given. During the afternoon, the students did wardriving and found more than 100 wireless LANs in total. The question "Where is information security today and what in the future?" was answered during the coffee-table talk by Dogan Kesdogan, a researcher from RWTH Aachen University.

In the evening, we had a social event: We drank beer, ate pizza and watched "Office Space" and some movies from the demo scene - perhaps a typical evening for hackers :)

On Wednesday we demonstrated some of the current research topics of our Lab: The first lecture covered "Hidden Data In Documents" and explained that there are many interesting data in proprietary formats that can be misused. The second lecture gave an introduction to honeynets and related security research. Like the day before, interested students had the possibility to search for wireless LANs in the city. Some participants stayed at the lab and developed their projects further. One project was especially interesting: One attendee wrote a crawler which automatically searches for images with an included thumbnail in the Exif header. At the end of the Summer School, he had downloaded about 3.000.000 images, from which about one percent had a significant difference between the image and the included thumbnail. This shows that a significant amount of images has interested data hidden inside the document itself. More on this at the Congress, At this day, one of the students prepared the coffee-table talks: Ilja van Sprundel filled with his talk on heap-based overflows a gap that was left by the lectures on programming errors.

The rest of the week gave the students the possibility to do some research on their own and implement novel techniques. To prepare this, we collected ideas for further research on Thursday morning. From Thursday noon until Friday evening, the participants had time to implement their projects. The resulting projects covered tools for covert channels and fingerprinting applications.

Furthermore, a fuzzing framework and low level network libraries were implemented and an embedded device was introspected. The coffee-table talk on Friday gave an insight into the relation between project management and security. Rainer Lingmann from Pixelpark gave a talk entitled "IT-Security: Das Gretchenprojekt aus Projektmanagment Sicht".

[[[Third week]]]

The last week started with two lectures about computer forensics. These lectures covered the basic of disc forensics and gave an introduction on computer forensics in general. During the lab session the students had to solve a game: We gave them a Compact Flash card which we pretended to have retrieved from a suspected terrorist. They had the task to reconstruct the corrupted files and retrieve as much information as possible, especially the place and time for the next "terrorist meeting". It was much fun and many students were able

to recover the damaged files. The second task for this afternoon was forensic imaging of used hard discs that we had previously bought at ebay. The students found some hard discs which contained interesting information: One disc was obviously the former hard disc of a cashier terminal or cashier computer because we found bills and accounting information on it.

Another hard disc was used in a bookstore before and contained many hundreds of e-mails with partially sensitive information. And a third hard disc was obviously used in a mailbox and the students were able to reconstruct the user database. How to realize "Software Detection of Currency" was presented by Steven Murdoch, a participant of the Summer School and Ph.D. student from Cambridge.

Photo credits for this article: George Danezis

An external guest gave two lectures on Tuesday: Knut Eckstein from the NATO C3 Agency talked about "Advanced Filesystem Forensics - Journaling Filesystems". More forensics of disk images was the topic of the lab session. This was too much forensics, and the students got bored. During the coffee table talk, fd0 from CCC Cologne gave a talk about grsecurity.

The lectures on Wednesday concentrated on malware for Linux. The first talk gave a general introduction to the topic and presented rootkits and backdoors. In contrast to that, the second lecture concentrated on code infection for ELF binaries. During the lab session, the students deepened the knowledge gained during the lectures and some wrote ELF modification code. One of the students provided a small challenge for the lab session: Lisa Thalheim prepared an ELF binary and posed some questions. Again, the coffee table talk was prepared by Ilja van Sprundel. He talked about format-string attacks and filled another gap that was not covered during the lectures on exploitation of programming errors.

An excursion to some abandoned industrial sites was arranged for Thursday. We wanted to have an alternation to the usual schedule and therefore organized a trip to a coking plant in Essen, a gasholder in Oberhausen and some geocaching locations in Wuppertal. The main goal for this tour was fun and getting to know each other better.

We used the last day of the Summer School to get feedback on the three weeks. But the highlight of this day was the Wargame with several levels and increasing difficulty, which Christian Klein had prepared beforehand. It offered the students the possibility to apply all techniques they had learned in the previous days. Fun prevailed during this game, with some minor exceptions due to missing tools.

[[After the Summer School]]

After the three weeks, some further noteworthy things happened. The participants of the Summer School had submitted six talks to the Chaos Communication Congress and all six submission were accepted for presentation. The participants of the Summer School also agreed to meet at the Congress again. You will probably recognize us by our t-shirts :)

The biggest lesson we learned: Hacking is fun for students. For example, when the students exploited their first buffer overflow, they were very enthusiastic and worked hard to learn more.

At the end of the Summer School participants were asked to fill out a questionnaire to give use some pointers for improvements. In summary, the feedback was mostly positive. The main criticisms were:

• Three weeks are too long, especially if the students have to work so hard and spend so much time at the courses as they did

• Some lectures need improvements, on one hand the depth of the talks and on the other hand the understandability

In conclusion, the concept of the Summer School was very sound and it is already planned to repeat that event in 2005.

You can find our wiki used during the Summer School at *http://weltwissen.koeln.ccc.de/wiki/index.php/Summerschool_Aachen_2004*

It has links to all slides and photos.

Deutsche Homeland-Security: E-Mail-Überwachung 2005.

Volker Birk <volker.birk@ulm.ccc.de>

Flirten, Lästern, Tratschen. Und wir lesen Ihre Mails mit: Die Polizeien und Geheimdienste! Ein Blick auf ein Neusprech-Dokument der besonderen Art: die TKÜV.

Nun ist es soweit: die wohl anrüchigste Maßnahme im Rahmen der Telekommunikationsüberwachungsverordnung - kurz TKÜV - wird, längst beschlossen, ab dem 1.1.2005 Wirklichkeit in Deutschland. Die E-Mails sind's diesmal, die der Staat furchtbar gerne mitlesen möchte.

Von vielen unbemerkt ist diese Verordnung auch zum E-Mail-Überwachen längstens Realität; allein eine "Gnadenfrist", formuliert in § 30 "Übergangsvorschriften", bewahrte die Maildienstleister bisher, jetzt schon Überwachungsschnittstellen für Polizeien und Geheimdienste in die Mailserver einzubauen. Diese Gnadenfrist läuft nun an Neujahr ab.

Damit schafft es die Regierung, gleich zwei Grundrechte der Bürger auf einen Streich ad absurdum zu führen: das Brief- und das Fernmeldegeheimnis (beide Art.10 GG). Und wie es inzwischen üblich ist, erlegt unsere Regierung die Kosten dafür den Providern auf - der Staat ist halt grade knapp bei Kasse, da muss man Verständnis haben.

Was sind schon Arbeitsplätze, wenn es um die Staatssicherheit geht?

Es darf jetzt jeder seine private E-Mail-Überwachung selber bezahlen, denn die E-Mail-Anbieter legen die Kosten natürlich auf die Verbraucher um - was sollen sie auch sonst machen? 20.000 EUR und mehr kostet allein so eine "Überwachungseinrichtung" pro Stück, so der Verband der deutschen Internetwirtschaft, ECO. Ab 64.000,- EUR erhält der "verpflichtete" Provider auf dem Markt eine "Komplettlösung" zur Spitzelei. Eine Summe, die manchem Maildienstleister zu schaffen macht. Kein Pappenstiel also in einem hart umkämpften Markt, in dem es um tausende von Arbeitsplätzen geht.

Diese Kosten gehen nämlich vor allem kleineren Providern, die zwischen 1.000 und 10.000 Kunden haben, an die Substanz. So erklärte beispielsweise Canhost, ein Provider aus Hamm, seinen Kunden in einer Rundmail, dass ihn diese Maßnahme der Bundesregierung dazu zwinge, die "enormen Kosten ... auf den Endkunden" umzulegen. Pro Kunde werden so jeweils zunächst mal 5 EUR fällig, die dieser Kunde für die Bespitzelungsinstallation für seine eigenen Mails beisteuern darf.

Martin Seeger von NetUse, einem Provider in Kiel, geht davon aus, dass die Kosten fürs Abhören künftig 15 Prozent der Telekommunikations-Preise ausmachen werden. Hoffen wir mal alle, dass die Kunden dieses und vieler anderer Provider Verständnis zeigen werden - und nicht zu E-Mail-Providern im Ausland wechseln werden, die von der Maßnahme nicht betroffen sind. Zu Providern, deren Arbeitsplätze im Ausland sind.

lieschen@gmx.de? Aber bitte CC an den Verfassungsschutz, nicht wahr? Oder besser gesagt: BCC.

Denn merken darf der Bespitzelte davon natürlich nichts (bzw. der "Teilnehmer" unter seiner "Kennung", wie es im Beamtendeutsch heisst). Den Providern ("Verpflichtete") wurde gleich ein Maulkorb mit auferlegt. So heisst es in § 15 "Verschwiegenheit":

(1) Der Verpflichtete darf Informationen über die Art und Weise, wie Überwachungsmaßnahmen in seiner Telekommunikationsanlage durchgeführt werden, Unbefugten nicht zugänglich machen.

(2) Der Verpflichtete hat den Schutz der im Zusammenhang mit Überwachungsmaßnahmen stehenden Informationen sicherzustellen. Dies gilt insbesondere für Informationen darüber, welche und wie viele Kennungen einer Überwachung unterliegen oder unterlegen haben und in welchen Zeiträumen Überwachungsmaßnahmen durchgeführt worden sind

Oder anders formuliert: Bespitzeln müssen die Provider, aber sie dürfen niemand was davon sagen. Schon gar nicht dem Bespitzelten selber.

Kontrolle? Ach was, wozu denn, habt Vertrauen!

Das reicht doch, wenn die jetzt schon mit vielen Aufgaben überladene Regulierungsbehörde für Telekommunikation und Post bei Interesse mal ins Protokoll schauen darf (§ 17).

An obligatorische Prüfung, wozu denn eigentlich überwacht wurde, und warum das so notwendig war, und wieviel Bespitzelung und damit Grundrechtsverletzung denn nötig ist, um zu den doch erfahrungsgemäß recht bescheidenen Ergebnissen zu kommen, daran wurde in der gesamten Verordnung in keinem der 31 Paragraphen gedacht.

Und das, obwohl der Nutzen der ganzen Spitzelei höchst fragwürdig erscheint; die Zahl der Spitzelmaßnahmen ist zwar insgesamt explosionsartig nach oben gegangen - bei der Telefonüberwachung z.B. von 4.674 Maßnahmen 1995 auf 24.441 Maßnahmen in 2003. Ein ähnliches Verhältnis wird wohl auch bei der E-Mail-Überwachung zu beobachten sein. Der große Ermittlungserfolg dagegen durch die massive Abhöraktion stellte sich nicht ein - es ist nichts wesentliches Erfreuliches in der Hinsicht zu beobachten.

In einem Rechtsgutachten zur Telefonüberwachung, das die Bundesregierung beim Max-Planck-Institut für ausländisches und internationales Strafrecht, Freiburg, beauftragt hat, kommen die Experten deshalb auch zu einer kritischen Sicht der Zustände im (Noch-)Rechtsstaat. So erklären die Gutachter, dass der gesetzliche Richtervorbehalt nicht gelockert werden dürfe. Auch seien Berichtspflichten für die Strafverfolgungsbehörden zur Kontrolle der Entwicklung bei Überwachungsmaßnahmen notwendig. Der Umfang der Benachrichtigungspflichten von Betroffenen sei zu erweitern. Gespräche zwischen Beschuldigten und zeugnisverweigerungsberechtigten Personen dürften grundsätzlich nicht verwertet werden. Der Umfang des Straftatenkataloges des § 100 a Strafprozessordnung müsse wieder reduziert werden; dieser wurde in den letzten Jahren ständig erweitert.

Statt nun diese Probleme anzugehen, die das Max-Planck-Institut ihrem Auftraggeber, der Bundesregierung, ins Stammbuch geschrieben hat, tat die Bundesregierung lieber - nichts. Und jetzt kommt die E-Mail-Überwachung dazu.

Das alles ist doch nur gut für Euch, wir fangen damit die bösen Terroristen!

Schaut man z.B. die Personen um Mohammed Atta an, so erblickt man Männer, die an der TU Hamburg-Harburg Elektrotechnik studiert haben, junge Leute, die mit dem Netz groß geworden sind. Und die sollen keine Ahnung haben, wie man der Überwachung entgeht?

Es ist imer dasselbe Argument, das ständig in unseren Ohren klingelt: wir müssen Eure Bürgerrechte ignorieren, weil wir Terroristen jagen. Dazu richten wir jetzt Bespizelungsmöglichkeiten für Euer Privatleben ein. Aber treffen diese Maßnahmen überhaupt auch die Zielgruppe? Immerhin bedeuten sie eine Grundrechtsverletzung für alle deutschen Bürger, da muss dann aber schon gewaltig was rüberkommen an Ergebnis, wie soll sonst das Prinzip der Verhältnismäßigkeit noch gewahrt bleiben?

Tatsächlich aber gilt wohl eher: es kommt so gut wie gar nichts bei rüber.

Nur ein geistig sehr armer Terrorist wird ausgerechnet über deutsche E-Mail-Dienstleister mailen - und das noch unverschlüsselt - so dass Polizei und Geheimdienst mit Tatütata ausrücken und ihn verhaften können, wenn er seine subversiven Abmachungen per deutschem Webmailer an eine deutsche Mailadresse absendet. Ein Szenario, das eher in den Bereich "nur in Deinen Träumen, mein sehr junger Padavan" fällt, als in den Bereich "realistische Projekte".

Die Bespitzelung wirkt nur bei den Leuten, die kein PGP einsetzen - oder Ihre Nachrichten nicht per Mail, schon gar nicht über leicht zu überwachende SMTP-Server laufen lassen. Die gesamte Truppe um Atta war aber nicht durch besondere Ignoranz und technisches Unvermögen, sondern von Kompetenz und organisatorischem Sachverstand geprägt.

Wie will man von solchen Leuten erwarten, dass sie unverschlüsselt per Mail kommunizieren über überwachte Dienste? Man wird es nicht erwarten können. Und das wirft die Frage auf: wer soll hier wirklich überwacht werden?

Willkommen in Ozeanien!

Das Ministerium für Wahrheit erklärt, dass alle diese Maßnahmen notwendig und sinnvoll sind, gleich welche Kritik die unwissenden Mahner wie die Riege der Datenschutzbeauftragten oder irgendwelche Rechtsgutachter auch immer äußern werden. Deshalb braucht ihr vor allem eines nicht: OpenPGP.

Was Ihr da am allerwenigsten braucht, findet Ihr unter *http://www.gnupg.org*. Bitte setzt es auf keinen Fall ein, erst recht nicht in Zusammenarbeit mit *http://www.mozilla.org/projects/thunderbird/* und *http://enigmail.mozdev.org/*

So helft Ihr mit, dass wir auch in Zukunft die Maßnahmen des Ministeriums für Frieden unterstützen können, Frieden und Glückseligkeit über die Welt zu verbreiten.

"Those who would give up liberty for a little temporary safety deserve neither liberty nor safety, and will lose both." (Benjamin Franklin)

Links:

http://www.bmwa.bund.de/Redaktion/Inhalte/Pdf/TKUEV1,property=pdf.pdf

Ein Friedensangebot im Kampf ums Copyright

von Matthias "wetter" Mehldau <wetter@ccc.de>

Als Kompromiss zwischen faschistischem Kontrollsystemen für Musik und anarchischen Zuständen in Filesharingnetzwerken wird die Kulturflatrate vorgeschlagen.

Der Kampf ums Copyright im Internet ist entflammt und die Fronten sind klar: Auf der einen Seite stehen die Computerfreaks als technische Anführer der Internetrevolution. Sie entwickeln ständig neue Raffinessen, um "ihre" Musik Kopieren und Tauschen zu können. Auf der anderen steht die Musikindustrie. Sie möchte "ihre" Musik mit immer neuen Kopierschutzmaßnahmen davor schützen, unerlaubt kopiert zu werden. Mit periodischen Klagewellen sollen rechtsbewusste Filesharinguser von illegalen Downloads abgehalten werden.

Für die mächtige Musikindustrie sieht es allerdings schlecht aus: Die Technik und der Geist von Filesharing ist aus der Flasche und lässt sich nicht wieder einholen. Lieber möchten die Manager Songs wie bisher verkaufen, als ob sie weiterhin an einen physischen Träger gebunden seien. Doch die dazu notwendigen Kopierschutzsysteme wurden nun immer aufs Neue von Teenagern geknackt und auch künftig werde ich immer, wenn ein Song meine Ohren erreichen können soll, ihn auch aufzeichnen können. DRM - Digital Restrictions Management - zwecklos.

Doch so dramatisch braucht die Diskussion nicht geführt zu werden. Beide Seiten wollen ja nur, was ihnen zusteht: Der Musikhörer will privat Musik kopieren dürfen - etwa auf CD unter Freunden oder heruntergeladen aus einem Filesharingnetzwerk. Der Musikmacher möchte von seiner Kunst leben können und seine Kosten gedeckt haben. Dieser Wunsch wird von der Musikindustrie deutlich übertriebener formuliert, als es der Musiker tut, um im Wesentlichen ihren Profit zu maximieren.

Laut der GEMA verdienen Musiker heute ihr Geld zu etwa je einem Drittel durch Konzerte, Merchandising und dem CD-Verkauf. Doch der Musikhörer ist schon lange in der Lage, auch anders Musik zu empfangen: Aus dem Radio kann er Musik auf aufnehmen, etwa auf Kassette. Um sicherzustellen, dass der Musiker auch für diese Kopien Geld bekommt, fliesst eine Abgabe vom Kassettenpreis an die Verwertungsgesellschaft für Musiker, die Gema. Nach diesem Prinzip gibt es bereits seit den 60er Jahren Pauschalabgaben auf leere Kopiermedien (Kassetten, CDs, DVDs) und Kopiergeräten (Brenner, Fotokopierer), die auch an die Verwertungsgesellschaften für Bild, Ton, Wort und Film gezahlt und von diesen unter den Urhebern aufgeteilt werden.

Künftig gibt es keinen absehbaren Grund, weshalb durch das Internet weniger Konzerte besucht und T-Shirts gekauft werden. Da der Mensch bequem ist, würden wohl aber weniger CDs verkauft - das würde genauso passieren, wenn es Musik im Netz nur noch kopiergeschützt und direkt kaufbar gäbe. Im Netz können allerdings grade kleine Musiker einfach Alben im Netz anbieten und direkt

Eine fast in Vergessenheit geratene Tradition bei der Datenschleuder: Kabelbilder. Was bei der 2600 die "Payphones from around the World" sind bei uns Leserfotos von aussergewöhnlichen Kabelsituationen. Besten Dank geht an Enno Lenze für das oben abgebildete Prachtexemplar! Keep 'em comin'!

vertreiben. Wer meint, keine CDs zu brauchen, aber die Musik trotzdem geil findet, kann über elektronische Bezahlsysteme auch kleine Beträge freiwillig spenden.

Nun, wenn aber freiwillig nichts passiert im Land der Schmarotzer, braucht es Zwang: Die Pauschalabgabe, als Kompensation für Privatkopien in Deutschland erfunden, soll auf Computer oder Internetzugänge ausgedehnt werden. Für fünf Euro im Monat würden nicht-kommerzielle Kopien über Filesharingnetze vergütet und die Kosten für unwirksame Kopierschutzsysteme und wahnwitzige Strafverfahren gespart, lauten die Argumente der Befürworter. Sie sind bei Netzaktivisten, wie dem Chaos Computer Club, den Globalisierungskritikern attac und bei grünen Politikern zu finden. Für diese sind Kopierschutzsysteme schon allein aus bürgerrechtlichen Gründen tabu: Der gläserne Musikhörer soll nur in den Köpfen der Datenschützer existieren. Um die Forderung nach einer solchen "Kulturflatrate" zu manifestieren, haben sie Anfang Dezember die Kampagne "Fairsharing" ins Leben gerufen, auf deren Website [1] dazu Unterschriften gesammelt werden.

Aber gleich zu einer general-verpflichtenden Pauschalabgabe - auch wenn sie den schönen Namen "Kulturflatrate" trägt - zu greifen, scheint manchem überzogen: Braucht es eine verpflichtende Vergütung im Netz - oder können sich Musiker anpassen und der Markt reguliert sich von selbst? Zwischen diesen beiden Modellen - Kopierschutz und Pauschalabgabe - bewegt sich die Freiwilligkeit. Was wäre, wenn jeder, aus seiner Sicht angemessen, seine Lieblingsmusiker bezahlt? Es wäre spannend zu beobachten, wie es guten und schlechten Musikern ergeht, die diesen Weg wagen. Noch eine abschließende Überlegung: Als "verpflichtende Freiwilligkeit" lässt sich das "Straßenkünstler Protokoll" umschreiben; Dabei veröffentlicht Musiker sein nächstes Werk erst dann, wenn ihm seine Fans einen Betrag gezahlt haben, von dem er meint, dass er das Recht wert ist, seinen Song privat zu kopieren.

Es gibt durch die Internetrevolution nicht nur mehr ungeahnte Probleme, sondern auch noch mehr ungeahnte Lösungsmöglichkeiten. Das Musikgeschäft ist an einem Punkt, an dem es wagen und ausprobieren muss. Das Problem der Vergütung von geistigem Gut im Netz ist eine sehr brisante: Sie stellt sich gleichermaßen auch für Filme, Texte und Fotos. Die Musiker befinden sich hier in einer Vorreiterrolle.

/join silcnet

von Alien8 <fb@c3d2.de>

Was waren das für Zeiten, als das Internet noch frei von Flash-Animationen, vorgeschriebenen Impressen und Anwälten war?

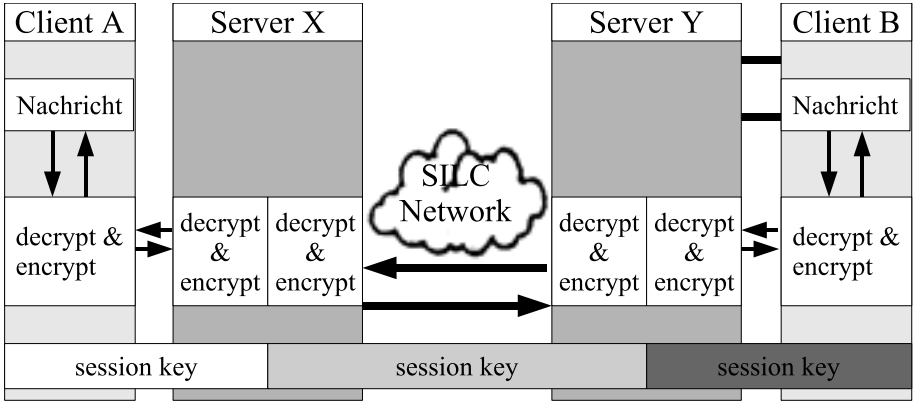
Eigentlich egal. Die heutigen Zeiten sind nicht weniger spannend. Studentennetze rüsten auf, sich von den Schilys dieser Zeit beschüffeln zu lassen, MP3 Tauscher werden hart aber gerecht bestraft und Steuergelder die einst für die Entwicklung des Netzes dienten, werden jetzt ausgegeben, um Überwachungsbots für Chatkanäle zu schreiben. Wahrlich, es werden noch wundersame Dinge geschehen.

Zeit, die Klartext-Urgesteine des Internets abzulösen. Was für telnet recht ist, sollte für IRC billig sein. Dieser Artikel soll Lust auf *das Chat* und Instant Messagingsystem SILC (Secure Internet Live Conferencing) machen, es kurz vorstellen.

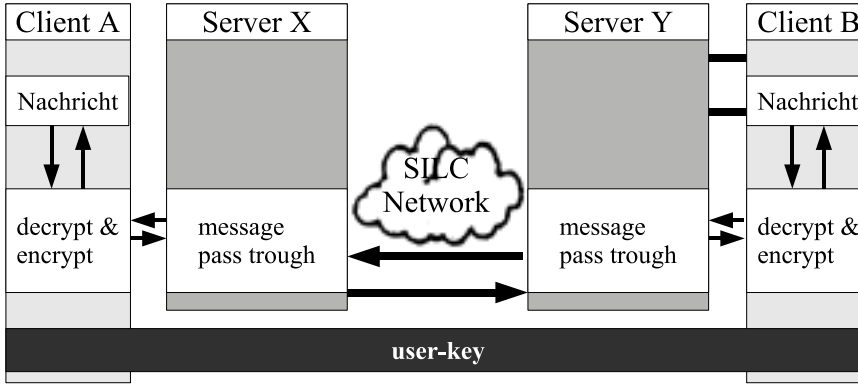
Was braucht es also für ein Chatsystem, damit es authentifizieren und verschlüsseln kann? Könnte man nicht, wie bei http, einen SSL Layer einziehen? Nein, denn was für eine Client/Server Verbindung entworfen wurde, muss nicht auf Systeme mit mehreren Servern und unzählig vielen Clients passen. Mal vom Zertifikats-Dilemma abgesehen, wie können mehrere Clients den selben Sessionkey nutzen? Woher weiß man, dass die Session wirklich bis zu jedem Client verschlüsselt und authentifiziert ist? Das Protokoll gibt es jedenfalls nicht her.

Initiert von Pekka Riikonen, einem Finnen - (was sonst?), wird SILC seit '96 entwickelt. Mittlerweile kann man es gut nutzen. Die Clientversion ist über 1.0 hinaus, Gaim z. B. unterstützt es seit ein paar Versionen, die Server Software und das Toolkit stehen kurz vor der 1.0.

Meldet sich ein Client beim Server an, wird zu erst mittels Diffie-Hellman ein Session Key ausgetauscht. Der Server wählt die Methoden für Cipher, Hash, HMAC und Public Key Funktion aus dem aus, was der Client anbietet. Das heißt Silc Key Exchange (SKE) Protokoll. Ausgerüstet mit einem Sessionkey authentifizieren sich die Kommunikationpartner per Passphrase oder Public Key über eine Challange. Momentan werden nur SILC-eigene Keys unterstützt, OpenPGP- und X.509-Zertifikate sind vorgesehen. Damit sind der Client und der Server gegen Man-in-the-middle Attacken so gut geschützt wie bei ssh: "First time's free, kid". Die Fingerprints der Server gibt's auf *http://silcnet.net*.



Private Nachricht mit Session Keys



Private Nachricht mit Private Key

Natürlich kann aus Performance- und Redundanzgründen nicht alles über einen Server laufen. Also muss es mehrere Server geben, die wiederum mit einem sogenannten SILC-Router verbunden sind. Dieses Konstrukt nennt sich Cell. Die Router lassen sich auch miteinander verbinden und bilden einen Ring. Fällt ein Router aus, reden die anderen über ihre Backup-Router. Das Konzept nennt sich Hybrid Ring-Mesh.

Nachrichten werden also von Client A zum Server über den Router wieder zum Server und schließlich zum Client B übertragen. Alles authentifiziert und verschlüsselt. Dem aufmerksamen Leser wird es schon aufgefallen sein: der Server, oder besser noch der Router, ist ein idealer Punkt, die Kommunikation abzuhören. Er muss alle Nachrichten ent- und verschlüsseln. Um diese Schwäche zu umgehen, haben die Protokollentwickler eigene Keys vorgesehen. D. h. die Server schalten einfach auf Message pass through und mitlesen kann nur, wer auch den Key (preshared oder public) kennt. Den Trade-off Bequemlichkeit vs. Kontrolle kann jeder selbst wählen.

Was kann man damit nun so anfangen? Bei einem Chatsystem ist das nicht schwer zu erraten. Channelnamen müssen nicht mit "#" beginnen. Nicks können doppelt vorkommen, Fingerprints nicht. Damit sind Nickwars passé. Der Founder eines Channels kann diesen als permanent setzen. Authentifiziert wird der Founder über seinen Public Key, wenn nich anders angegeben. Dieser beruft weitere Operators, die im Channel ihre Vorstellung von Ordnung durchsetzen können. Wenn man schon ein Schlüsselpaar hat, kann man Nachrichten ähnlich PGP natürlich auch signieren, was dem Leser nur nützt, wenn er den Public Key vorher geladen und überprüft hat. Vorsicht! Da Nicks nicht eindeutig sind, erscheinen akzeptierte Keys immer als gültig, auch wenn der Nick wechselt. Filetransfer läuft über sftp sozusagen Peer to Peer. Die Schlüssel werden per SKE ausgehandelt und die Datei danach direkt von Client zu Client geschickt. Mittels von irssi bekannten Scripterweiterungen kann man den Client erweitern. Eine nette Sache ist die silc-mime.pl Erweiterung. Damit lassen sich Dateien an einen Channel schicken, die per .mailcap einer Applikation vorgeworfen werden können.

Eine andere nützliche Sache ist, dass man eine SILC Session detatchen kann. Damit bleibt man im SILC Netzwerk, kann z. B. den Client updaten und bekommt seine Session zurück.

Die Clients

Der Standard ist silc-client, ein irssi-Ableger, der, wie das irssi-Plugin, von c0ffee gepflegt wird. Grafische Clients gibt's im gtk-Gewand, silky war zuerst da, Alleskönner Gaim unterstützt SILC seit ein paar Versionen, auch wenn sich das noch nicht bei allen Paket-Maintainern rumgesprochen hat, sowie einen auf irssi basierenden IRC/SILC-GUI-Client für MacOS X mit Namen Colloquy.

Beim Verbinden zu einem Server sollte man beachten, dass *silc.silcnet.net* round-robin DNS verwendet. Ggf. kommt man besser, wählt man sich einen in seiner Nähe aus. IPv6 ist natürlich auch kein Problem.

Hoffentlich hat Euch das ganze hier etwas Lust gemacht, mal SILC auszuprobieren. Wenn es noch Fragen gibt oder Ihr einfach einen Startpunkt in SILC sucht: `/join c3d2`

CU there!

whois <<</>>

von Mirko Swillus <mechko@thur.de>

Wohin in Deutschland, wenn man sich als chaosnahen Hacker versteht und gleichgesinnte sucht? Gibt es vielleicht in der Nähe Treffen von kompatiblen Wesen? Sind die dort Anwesenden auch echte Nerds? Um diese Fragen zu klären gibt die Redaktion Datenschleuder Erfakreisen und solchen, die es werden wollen, Gelegenheit, sich vorzustellen. Hier nun eine Selbstdarstellung des Chaostreff und Erfa-Kandidaten Dresden.

Es dämmerte bereits, als er sich auf den Weg ins Hechtviertel machte. Er hatte schon den ganzen Nachmittag überlegt, wie er wohl den Abend zubringen sollte und war zu keinem rechten Entschluss gekommen. Seine Kommilitonen hatten schon am Mittwoch angekündigt, Freitag "mal wieder ordentlich einen trinken zu gehen". Aber die Vorstellung, am Freitag Abend in einer der zahlreichen Neustädter Kneipen ein Bier nach dem anderen zu kippen und zum Schluss völlig fertig im "Dürüm Döner" zu versauern, fand er nicht gerade sehr verlockend.

Die Jungs im silc hatten ebenso keine Perspektive gehabt, und man hatte daraufhin gemeinsam beschlossen, sich spontan im "Büro" zu treffen. Er musste grinsen, als er an die neue Bezeichnung ihres gemeinsamen Raumes in einem alten, völlig unsanierten Haus dachte. Schon wegen dem Zustand der Immobilie war sie ein Kracher. So blöd ist das allerdings nun auch nicht, dachte er weiter, immerhin organisieren wir da auch wichtige Sachen. Früher mussten sie sich dafür immer in Neustädter Kneipen treffen, was schon auch Vorteile hatte, allerdings kamen die wichtigen Dinge da immer etwas zu kurz, wie er fand. Viel macht man ja auch im silc und über das wiki, aber so ein Treffen ist da eben doch besser. Damals bei der Vorbereitung für das große Symposium "Datenspuren - Privatsphäre war gestern." waren die Treffen, weil in Kneipen überm Bier abgehalten, meistens auch arm an Ergebnissen geblieben. Trotzdem hatten sie es zum Schluss reissen können und die eintägige Veranstaltung mit 300 Besuchern und 20 Referenten zum Erfolg führen können. Noch heute ist ihm der Dreiklang "Datenschutz. Privatsphäre. Bürgerrechte." im Ohr, den sie damals unter der Warnung "Ich weiss, was Du letzten Sommer gemailt hast." auf zwanzigtausend Postkarten gedruckt und damit ganz Dresden geflutet hatten. Seitdem war ihm klar, dass die Gruppe was erreichen konnte, dass das hier neben Spaß am Gerät auch eine ernsthafte Veranstaltung ist, dass es um was geht. So ernsthaft, dass sie mittlerweile auch ein Büro braucht, musste er wieder grinsen.

Inzwischen hatte er den Rand der Neustadt erreicht. Immer größer wurde der Strom von jungen Leuten, die ihm mit bierdurstigen Gesichtern entgegenkamen. Wie so oft versuchte er, die Leute bestimmten Subkulturen zuzuordnen und diese dann nach Häufigkeit zu ordnen. Und wieder nahm er es am Ende seiner kleinen Erhebung mit einem leisen Grunzen hin, dass die Gruppe "Yuppie" am stärksten vertreten war. Freitag Abend eben.

Als er in die Hechtstraße einbog, überlegte er, welcher Subkultur er sich wohl selbst zuordnen würde. Rein äußerlich würde das schon

schwer werden, vielleicht so eine Mischform zwischen Gruppe Linksalternativ, den Emocore-Poppern und der Öko-Abordnung. Vom Aussehen her gibt es den Typ "Hacker" jedenfalls schon mal nicht, da war er sich sicher. Denn er hatte nicht das Gefühl, dass die Leute, die zu den c3d2-Themenabenden kamen, frisur- oder klamottentechnisch in irgendein Muster passten. Immerhin veranstalteten sie diese Abende seit nun schon fast einem Jahr, und bei durchschnittlich einer Veranstaltung im Monat mit immerhin 60 bis 70 Gästen kann man da schon von äußerlicher Repräsentativität sprechen, wie er fand.

Und "innerlich"? Niemand bezeichnet sich selbst als Hacker, jedenfalls nicht, ohne danach ein echtes Glaubwürdigkeitsproblem zu haben, soviel ist mal klar. Auf der anderen Seite haben die Punks, mal als Beispiel, selten Probleme, sich selbst als Punk zu bezeichnen - oder später irgendwann mal weintrinkend im Ledersessel zu sagen "Das war damals, in meiner Punkzeit". Da kann man schon neidisch werden, dachte er. Aber er fühlte sich doch zugehörig und irgendwie heimisch bei den Dresdner Hackern. Ansonsten würde er kaum seinen Freitagabend mit ihnen verbringen, dachte er weiter.

Er kam auf seinem Weg immer weiter ins Hechtviertel und wie jedesmal überkam ihn ein Gefühl der Vertrautheit. Hier hatte er sein Leben in Dresden begonnen und dann einige Jahre gewohnt. Als er noch neu in der Stadt war, war er ab und zu einem der Usergruppen-Treffen gegangen. Aber dort stellte er bald fest, dass es da eben tatsächlich nur um das jeweilige spezielle Interesse ging, ob nun bei der Linux User Group oder dem Unix-Stammtisch. Ihm fehlte neben den Diskussionen über die technische Funktionsweise die Frage nach der gesellschaftlichen Auswirkung. Vor welchen Entwicklungen haben wir Angst, wodurch werden diese bedingt und vor allem: Was können wir tun?

Ein paar Punks grölten auf der anderen Straßenseite durcheinander, und er glaubte einen Ramones-Klassiker wiederzuerkennen. Die Häuser waren hier zum größten Teil unsaniert, die noch bezahlbaren Mieten und die Nähe zur Neustadt hatte eine spezielle Mieterklientel angezogen, was das Hechtviertel mit einem ganz eigenen Charme erfüllte. Sicherlich gab es solche Konstellationen auch woanders in der Republik, allerdings war dies für ihn typisch Osten. Im Gehen stellte er nun die Theorie auf, dass der Chaostreff sich in der Anfangszeit auch deswegen so schnell entwickelt hatte, weil Dresden als Stadt vielleicht besonders fördernde Eigenschaften für die Bildung solcher kleinen Gruppen bietet. Oft hatte er den Satz "Dresden ist wie ein groß ausgedehntes Dorf" gehört. Immerhin mit Technischer Universität, mehreren Fachhochschulen und angesiedelten Halbleitergrößen.

Der Osten an sich scheint jedenfalls nicht sonderlich fördernd für chaosnahe Aktivitäten zu sein, überlegte er weiter. Von Berlin (Ost) abgesehen, gab es nach seinem Wissen keine nennenswerten Chaosaktivitäten. Früher hatte er ab und an mal was von Chaostreffs in Weimar oder Jena gehört, aber seit einiger Zeit schienen auch die nicht mehr zu leben. Während er in die Straße einbog, in der das Haus mit dem "Büro" lag, fragte er sich, warum Ost und West auch in dem Punkt immer noch verschieden waren. Er erinnerte sich dabei an eine der nächtlichen Diskussionen auf dem 17C3. Sie saßen damals komplette Nächte bei elektronischer Musik und Clubmate in der Lounge und redeten über Verschwörungstheorien oder Marxismus. Einmal ging es auch darum, dass gerade die Ostler durch ihre Geschichte ein natürliches Gefühl zu dem haben müssten, was der Club im Westen unter "Hacken" verstand. Kreativer Umgang mit Technik als fester Bestandteil des Alltags, polytechnische Schulbildung als eine der Voraussetzungen dafür. Aber die Landkarte der Chaostreffen und Erfakreise zeigte im Gegensatz zu diesen Theorien im Osten praktisch nur wenig Punkte, wie er wieder wehmütig feststellte.

Er fragte sich, wer wohl schon da war, als er die Zahlenkombination in das elektronische Türschloss eingab. Und schon im Treppenhaus hörte er die bekannten Stimmen aus dem "Büro". Er freute sich auf den Abend, auf die Nacht. Und er freute sich, dass c3d2 nicht nur für ihn wichtig war, sondern auch für eine immer größer werdende Gruppe. Mit interessanten Leuten, einer stabilen Struktur - und sogar einem eigenen Büro.

ICMP 2 - Koffeinsucht stillen und in der Sonne chillen

von Lars Weiler <pylon@ccc.de>

Campen macht dem gemeinen Hacker immer mehr Spaß. So ist es nicht verwunderlich, dass in den Jahren zwischen den großen Camps, wie dem Chaos Communcation Camp oder jenen in den Niederlanden, ein kleineres Camp stattfindet. Schließlich wollen die Zelte auch genutzt werden. [1]


Camp

Bei grandiosem Sonnenschein und über 30°C im Schatten haben etwa 100 Personen vom 5. bis 8. August 2004 den Weg ins fränkische Münchsteinach gefunden. Münchsteinach? Irgendwo habe ich das schon mal gelesen.... Klar, denn dort ist der Firmensitz der Loscher KG [2], dem Hersteller der köstlichen Club-Mate [3], ein auf Mate-Basis hergestellter Eistee, der in Hackerkreisen seit einigen Jahren sehr beliebt ist und auf den üblichen CCC-Veranstaltungen zu erwerben ist.

So kam vor drei Jahren die Idee auf, in die geheimen Hallen des Herstellers vorzudringen und gleichzeitig nebenbei ein wenig zu Campen. Der weniger als eine halbe Stunde Autofahrt entfernt liegende Erfa-Kreis Erlangen/Nürnberg/Fürth [4], in seiner Form als Bits'n'Bugs e.V., übernahm federführend die Organisation der Veranstaltung. So konnte eine anliegende Wiese des Sportplatzes vom SVS Münchsteinach zum Campen genutzt werden. Da dieser etwa einen Kilometer abseits vom Ort liegt, brauchten wir uns keine großen Sorgen über die Belästigung von Anwohnern machen.


Satellitenuplink

Seit der ersten ICMP im Jahre 2002 hat der Sportverein ein schickes WC- und Duschhaus gebaut, so dass keine Einbußen in der Hygiene in Kauf genommen werden mussten. Dixi ade! Das nahe Freibad mit Naturquellwasser sorgte für die kühle Erfrischung bei Tage von Außen und die im Kühl-Anhänger gelagerten Getränke der Loscher KG von Innen.

Insbesondere der "Partner-Tarif" - zwei Getränke zum Preis von eineinhalb - konstruierte ein interessantes soziales Netzwerk, um sich gegenseitig ein Getränk auszugeben.

Ein großes Bierfest-Zelt bot Platz für ein Hack-Center und die Workshops, für die viel Zeit eingeplant wurde. Jedoch mussten diese in die Abendstunden rücken, in denen der Beamer gegen die Kernfusion am oberen Horizont wieder ankam. Erst dann konnte sich die angesammelte Meute den Erläuterungen zu PDF als coolen Datencontainer, Informationen zu GSM und UMTS, Finetuning von Firewalls und NAT-Netzen oder dem aktuellen Status zur 21C3-Planung [5] hingeben. Nur die obligatorische Brauereibesichtigung fand an einem Vormittag statt.

Die BlinkenArea-Crew stellte ihre Nachbauten zu Blinkenlights und Arcade [6] ebenso aus, wie etliche neuere Basteleien, die stundenlang zum Verweilen und Anschauen einluden. Das Event-Phone-Team [7] sorgte mit ihrer von etlichen CCC-Veranstaltungen bekannten Telefonanlage für die interne direkte Kommunikation mittels DECT und Festnetz und testete damit ein weiteres Mal das Setup für die Feldtauglichkeit. Beim CERT - dem Chaos Emergency Response Team [8] - konnten sich von Mückenstichen und Sonnenbrand geplagte Hacker behandeln lassen. für das leibliche Wohl sorgte ein über alle Stunden geöffnetes Büffet und das Grillteam, welches abends die Camper mit Würstchen, Kotelett und selbst Spanferkel am Spieß versorgte.


Spanferkelgrillen

Eine dauerhafte Verbindung zum Internet konnte ab dem zweiten Tag mittels eines Satelliten-Uplink bereitgestellt werden. Nach ein wenig Trickserei stand schließlich ein UDP-Tunnel zu einem Server nach Berlin, welcher das Camp mit Raten zwischen 30 und 60 kByte/s in die Welt verband. Somit dürfte auch bei künftigen Veranstaltungen 'in der Pampa' eine Möglichkeit gegeben sein, Internet bereit zu stellen.

Dass Campen wieder in Mode ist, konnte an der Größe der Zelte festgestellt werden. Waren noch in den Vorjahren Iglu-Zelte aka Hundehütte 'in', so standen diesmal etliche mehrere Quadratmeter einnehmende Hauszelte auf dem Gelände. Schließlich sind diese kleinen Camps ein optimaler Test für die größeren Camps.

Ob eine ICMP 3 weiterhin auf dem bisherigen Gelände stattfinden wird, ist bei einer Besucherzuwachsrate von 100% fraglich. Die Kapazität des Platzes war schon nahezu erreicht. Doch können wir sicher sein, dass die Erlangener eine Lösung finden werden.


Blinkenarea

[1] *http://www.icmp2.de/*
[2] *http://www.brauerei-loscher.de/*
[3] *http://www.club-mate.de/*
[4] *http://erlangen.ccc.de/*
[5] *http://www.ccc.de/congress/2004/*
[6] *http://www.blinkenlights.de/*
[7] *http://www.eventphone.de/*
[8] *http://www.c-e-r-t.de/*

Link State Routing - Optimized?

von elektra <onelektra@gmx.net>

Die Freenetwork Community auf dem Weg zum sich selbst organisierenden drahtlosen Überallnetz.

Militärisches Forschungsgebiet und intelligenter Staub

Überall wird mit grossem Aufwand an der Technik sich selbst organisierender Funknetzwerke (Mesh-Clouds) geforscht. Die USA statten Soldaten mit PDAs aus, die sich auf dem Schlachtfeld drahtlos miteinander vernetzen. Zur Überwachung gegnerischer Armeen, zur Steuerung von Minen und anderer Nettigkeiten sollen drahtlos vernetzte Kleinstcomputer dienen - der 'Smart-Dust'. Das Ziel der Entwicklung von Smart-Dust ist die Größe eines Sandkorns und ein Preis unter 5 Dollar pro Stück - diese modernisierten Wanzen sollen u.a. in großen Mengen aus Flugzeugen abgeworfen werden. Die Pläne zur zivilen Nutzung des SmartDust beschränken sich auf Überwachungsaufgaben - auf Sensor-Netzwerke.

Dynamisches Ad-Hoc-Routing im Community-Netzwerk

Es ist wenig erstaunlich, dass bestimmte Implementierungen von Mesh-Routing-Protokollen überhaupt nicht zugänglich sind. Meshing ist andererseits viel zu interessant um es Generälen und Spitzeldiensten zu überlassen. Bei Wlan-Communities, die ein frei zugängliches Überallnetz aufbauen wollen, wird an freien Implementierungen / Verbesserungen für Mesh-Routingprotokolle gebastelt. Campaign Urbana Wireless (Cu-Wireless) ist dabei, HSLS (Hazy Sighted Link State routing protocol) zu implementieren. Die Freifunk-Community in der Berliner C-base arbeitet daran, eine Linux-Implementierung von OLSR [1] zu erweitern und verbessern, die Andreas Tønnesen in Norwegen als Diplomarbeit für die UniK programmiert hat. Nach dem Abschluss des Diploms wäre die Implementierung von Andreas ohne Unterstützung von Freifunk vielleicht in Schneewittchenschlaf gefallen, da Andreas selbst kein großes Interesse an Meshing hat.

Dank der Arbeit von Thomas Lopatic gibt es inzwischen auch Ports für OS-X, FreeBSD und Windows. Inzwischen wurde eine veränderte Form von ETX (Erklärung gibt's im weiteren Text) in OLSR implementiert. Soviel sei schon verraten: Der Freifunk-OLSR-Dämon weicht in vielen Punkten vom OLSR-Standard ab und ist nicht mehr kompatibel, wenn man die entsprechenden Features aktiviert. Das sollte man aber auf jeden Fall tun...

Verschiedene Routingmechanismen für Mesh

Die Probleme in einem mobilen Mesh sind zahlreich: Die Topologie ändert sich dauernd und Bandbreite ist immer knapp. Links weisen Paketverluste auf, sind nur kurzfristig vorhanden und ändern ihre Übertragungsqualität ständig. Das verwendete Protokoll muss schnell reagieren - darf sich aber auch nicht kirre machen lassen...

Proaktiv

Bei proaktiven Protokollen wie OLSR, DSDV oder MMRP (Mobilemesh) wird versucht, immer alle Nodes, die Links und Routingpfade zwischen ihnen zu kennen, egal ob diese Routen gerade benötigt werden oder nicht. Diese Protokolle basieren auf Link-State-Algorithmen.

Jeder am Mesh beteiligte Rechner pflegt eine Datenbank in der die gesamte Struktur des Netzes festgehalten ist. Man kann davon ausgehen, dass proaktive Protokolle in sehr großen Mesh-Netzen schlecht skalieren. Die Datenbanken werden sehr groß und die Menge der Informationen, die ausgetauscht werden müssen, um die augenblickliche Struktur des Netzes zu erfassen, wird zu einem Hemmschuh. Bei hochgradig mobilen Nodes, die sich alle gleichzeitig in unterschiedliche Richtungen bewegen können, wird die Datenbank

innerhalb kurzer Zeit wertlos und muss immer wieder neu angelegt werden.

Aus diesem Grund eignet sich OSPF nicht. OSPF erstickt in dieser Situation am selbstgenerierten Traffic.

Reaktiv
Ein anderer Weg sind reaktive Protokolle wie DSR oder AODV. Um den Rechen- und Kommunikationsaufwand zu minimieren, werden die Routingpfade nur bei Bedarf ermittelt, geprüft und gegebenenfalls neu verhandelt. Braucht ein Teilnehmer einen Link zu einem entfernten Mesh-Knoten, verbreitet er einen Route-Request (RREQ). Der RREQ wird so lange weitergereicht, bis er entweder beim gewünschten Node ankommt, oder auf einen Node trifft, der in seinem Cache bereits eine Route zum Zielrechner hat. Dieser sendet daraufhin einen Route Reply (RREP) auf dem Pfad wieder zurück, den der RREQ zuvor nahm.

Steht die Route, spielen reaktive Routingalgorithmen so lange keine Rolle mehr bis einer der am Weiterreichen der Pakete beteiligten Rechner feststellt, dass die Route nicht mehr funktioniert. Dieser Rechner verbreitet eine Route-Error-Meldung (RERR) mit einer Liste aller Nodes, die nicht mehr über den defekten Link erreichbar sind. Damit der Routingmechanismus nicht ziellos innerhalb des Netzes verläuft oder gar Endlosschleifen bildet wird eine sogenannte 'Destination Sequence Number' in den RREQ, RREP, RERR übertragen, anhand der das Protokoll erkennt ob es sich zielgerichtet verhält.

Die Entwickler versprechen sich von ihrem Ansatz, dass er selbst mit mehreren zehntausend Netzteilnehmern noch funktionieren soll. Reaktive Protokolle brauchen länger um eine Verbindung aufzubauen, da diese erst bei Bedarf gesucht wird.

Ausserdem existieren noch geographische oder hybride Protokolle. Geographische Protokolle routen anhand geographischer Koordinaten. Hybride Modelle wie ZRP arbeiten auf kurze Entfernung proaktiv und bei größerer Distanz reaktiv.

Probleme, Probleme

Die protokollbedingte Netzlast reduziert die nutzbare Bandbreite
Die Protokolle sollen schnellstmöglich die effektivste Route zu einem beliebigen Knoten im Mesh finden - ohne durch das ständige Übertragen von Informationen über die Struktur des Netzes eine große Netzlast (Overhead) zu erzeugen. Im schlimmsten Fall erstickt das Netz im selbst erzeugten Datenverkehr oder wird wesentlich gebremst.

Die kürzeste Route zu finden ist nicht ausreichend
Die meisten Protokolle berechnen die kürzeste Route. Das ist nicht immer günstig. Je länger eine Funkstrecke zwischen zwei benachbarten Rechnern ist, desto niedriger ist die Übertragungsrate. Der Paketverlust steigt und macht das erneute Übertragen von Daten notwendig.

Eine Route, die über zwei Funkstrecken mit jeweils 90 Prozent Paketverlust bei niedrigster Übertragungsrate führt, ist kaum benutzbar. Eine Alternativroute, die stattdessen einen Sprung mehr macht und dafür Funkstrecken mit geringen Paketverlusten verwendet, kann bedeutend schneller sein.

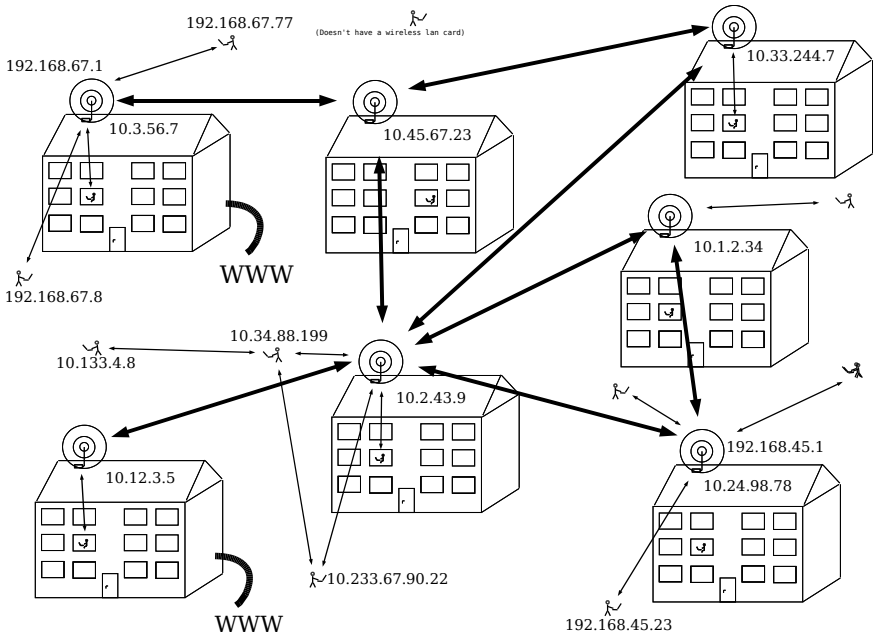
Protokolle, die derartige Erwägungen bei der Berechnung der Routen berücksichtigen, sind im Durchschnitt doppelt so performant wie jene, die einfach nur den kürzesten Pfad wählen. Die praktische Erfahrung zeigt, dass am Rande eines Mesh eine derartige Strategie darüber entscheidet ob man überhaupt noch eine brauchbare Verbindung hat oder nicht.

Dazu braucht es eine Bewertung der Links (Metrik), der Routingalgorithmus muss die Routen anhand der Metrik berechnen.

Am MIT wurde ein einfacher Algorithmus zur Ermittlung der Metrik entwickelt (ETX - Expected Transmission Count), der dieser Anforderung Rechnung tragen soll. Bei ETX schicken die Netzknoten in einem festgelegten Intervall eine Anzahl UDP-Pakete. Die Empfänger können anhand der Anzahl der empfangenen Pakete erkennen wie gut die Übertragung ist. ETX lässt sich leicht in vorhandene Protokolle integrieren. Die Idee, zusätzlichen UDP-Traffic zugenerieren gefällt überhaupt nicht. Deshalb hat der Freifunk-OLSRD eine abgewandelten ETX-Mechanismus: Es wird einfach eine Statistik über die empfangenen/verlorengegangenen Hello-Nachrichten der Nachbarn von jedem Node gebroadcastet - dieser Traffic entsteht bei OLSR sowieso.

Fairness
Das Routing muss bei der Berechnung der Routen vermeiden, dass ein zu großer Teil des Datenverkehrs durch einen einzelnen oder wenige Netzwerkknoten geschickt wird und somit ein Flaschenhals entsteht, der unter der Netzlast zusammenbricht und den Datendurchsatz behindert. Das lässt sich jedoch nicht immer vermeiden, etwa wenn es nur einen Link über einen einzelnen Netzknoten zwischen zwei Gruppen von Mesh-Nodes gibt. Links zu einem Flaschenhals haben aber ordentlich Paketloss und bekommen deshalb vom Freifunk-OLSRD eine schlechtere Metrik.

IPv6 ist wichtig
Die Konfiguration der Mesh-Rechner über DHCP ist ein schwieriges Unterfangen. Ohne eigene IP-Adresse funktioniert IP-basiertes Ad-Hoc-Routing nicht - man kann also nicht mit dem nächstgelegenen DHCP-Server kommunizieren, wenn dieser nicht in unmittelbarer Reichweite ist. In kleineren Gruppen kann man die Adressen manuell zuweisen. Die IP-Vergabe in Berlin geschieht bislang über das Wiki olsr.freifunk.net. Mit IPv6 kann man den Prozess automatisieren.



Link State Routing - Optimized?

OLSR versucht, die protokollbedingte Belastung im Netz zu minimieren, indem jeder Node nicht alle One-Hop-Neighbours zum redundanten Weiterleiten von Topologieinformationen verwendet. Einzelne One-Hop-Neighbours werden als sogenannte MultiPointRelays definiert - gerade so viele Neighbours, wie notwendig sind um alle anderen Two-Hop-Neighbours zu erreichen. Nur diese MPRs werden zum Weiterleiten von Topologieinformationen verwendet. OLSR hat ausserdem einen Hysterese-Mechanismus, der es wesentlich effektiver machen soll: Ein Link, der unregelmäßig funktioniert, und damit ein ständiges Neuberechnen von Routen herbeiführen würde, wird nicht dauernd berücksichtigt. Gehen drei Hello-Nachrichten nacheinander verloren (Default-Einstellungen) wird der Link gelöscht. Genau dieses Gimmik bereitete aber beim Testen auf der WizardOfOS3 und im alltäglichen Betrieb im Berlin-Mesh [2]Probleme. Durch die Hysterese wurden Links verworfen, die zwar schlecht, aber vorhanden waren. Dummerweise waren das aber des öfteren zufällig die MultiPointRelays, da OLSR die MPRs per Zufall bestimmt solange sie nicht aus der Hysterese fallen. Eine Hysterese kennt eben nur null oder eins - Ich hab nen Link, Ich hab ihn nicht... Das Resultat war eine lange Routingtabelle die immer wieder vollständig zusammenbrach um im nächsten Moment wieder komplett zu sein. Der Fix war: Hysterese aus oder bremsen, größere Redundanz bei den Multipointrelais. OLSR zu optimieren bedeutete also die Optimierungen teilweise abzuschalten oder ihren Einfluss zu verkleinern. Der Gedanke der Entwickler den protokollbedingten Overhead zu verkleinern führte zu Instabilität. Multipointrelais könnten ja schon Sinn machen - aber dann mit einer brauchbaren Metrik die die Wahl des MPR nicht dem Zufall überlässt. Deswegen ist ETX seit olsr-0.48 in olsr implementiert.

OLSR auf dem Weg zu ....

Die WLAN-Community verzichtet inzwischen auf die Hysteresefunktion von OLSR, die sich per Konfigurationsdatei abschalten lässt. Es lässt sich eben theoretisch kaum entscheiden was ein Mesh tatsächlich optimiert. Die Bewertung der Links anhand des Paketloss liefert dagegen eine brauchbare Metrik, deren Einsatz in ersten Tests eine enorme Verbesserung gebracht hat. Multipointrelais bringen nur dann etwas wenn das Mesh tatsächlich sehr dicht ist, also wenn viele Nodes zahlreiche direkte Nachbarn haben. Bislang überschreitet aber kein bekanntes Mesh eine Anzahl von 50 Nodes, die sich zu einer meshcloud zusammenschliessen. Im Moment ist die Implementierung von olsr.org IMHO die beste Basis für ein Communitynetzwerk. Bis zu welcher Teilnehmerzahl die gerade verfügbare Implementierung skaliert, ist noch ungetestet. Dazu müssen die Meshes erstmal eine bestimmte Größe überschreiten. Was vermutlich nicht lange dauern wird. Darauf freue ich mich schon.

[1] *http://www.olsr.org/*
[2] *http://olsr.freifunk.net/*

Bestellungen, Mitgliedsanträge und Adressänderungen bitte senden an:

CCC e.V., Lokstedter Weg 72, D-20251 Hamburg, Fax +49.40.401.801.41

Adressänderungen und Rückfragen auch per E-Mail an *office@ccc.de*

- Chaos CD Blue, alles zwischen 1982 und 1999 EUR 23 + EUR 3 Porto
- Alte Ausgaben der Datenschleuder auf Anfrage
- Datenschleuder-Abonnement, 8 Ausgaben
  Normalpreis EUR 32
  Ermäßigter Preis EUR 16
  Gewerblicher Preis EUR 50 (wir schicken eine Rechnung)
- Satzung und Mitgliedsantrag
  EUR 2,50 oder zum Selberausdrucken unter *http://www.ccc.de/club/membership*

Die Kohle

- liegt als Verrechnungsscheck bei
- wurde überwiesen am _____._____._____ an

  Chaos Computer Club e.V., Konto 59 90 90-201
  Postbank Hamburg, BLZ 200 100 20

Name: _____

Straße / Postfach: _____

PLZ, Ort _____

Tel.* / Fax* _____

E-Mail: _____

Ort, Datum: _____

Unterschrift _____

*freiwillig

Ab jetzt in Ihrem Reisepass: