Sicherheitslabor

Rainer Meier & Daniel Frey
skybeam@skybeam.ch; www.dafrey.ch

Klasse: 8Ibb-a
2002 - 2006

2006-07-08

1. Inhaltsverzeichnis

2. Einleitung

Diese Laborarbeit befasst sich mit der Windows Account Datenbank (SAM), deren Struktur, Sicherheit und Angriffsmöglichkeiten.

Da Microsoft unter Windows 2k/NT/XP's die SAM durch eine zusätzliche Verschlüsselungsschicht mit dem SYSKEY (Bootkey) verschlüsselt, werden wir uns vor allem mit dieser Verschlüsselung auseinander setzen.

Wir werden zeigen, dass es sich dabei um Sicherheit nach dem Motto „Security through obscurity“ handelt und werden ein kleines Programm schreiben, welches den SYSKEY aus der Registry auslesen kann. Zudem werden wir das Prinzip der SYSKEY – Verschlüsselung veranschaulichen und das grosse Microsoft Geheimnis lüften.

Um dies zu ermöglichen, ist eine kurze Übersicht über die Microsoft Registry und deren Aufbau unumgänglich.

Um die Theorie in die Praxis umzusetzen und zu beweisen, dass die Entschlüsselung der NT-Passwörter bei lokalem Zugriff möglich ist, werden wir eine Live-Boot CD erstellen. Diese baut auf der Basis von Windows PE (Preinstallation Environment) auf, welche durch den Bart's PE-Builder erstellt werden kann. Der Bart PE-Builder wurde von Bart Lagerweij, einem genialen Windows Profi, entwickelt und lässt sich durch beliebige Plugins erweitern. Unser selbst entwickelter SAM-Decrypter Krypto**N**ie**T**e werden wir, neben anderen Security Programmen, als Plugin mit auf der Windows-PE Boot CD installieren.

Am Schluss dieser Arbeit wird eine Enthüllung der SYSKEY Verschlüsselung und deren Anwendung auf einer Boot-CD verfügbar sein.

3. Windows Registry

Da unter Windows sehr viele Informationen in der so genannten Registrierungsdatenbank abgelegt sind, wird hier zuerst die Struktur und Idee hinter dem Konzept kurz erläutert.

Die Registrierungsdatenbank (nachfolgend kurz Registry genannt) ist eine hierarchisch aufgebaute Datenbank (ähnlich der Baumstruktur eines Dateisystems). Unter allen Windows-Versionen existieren Werkzeuge um die Registry zu bearbeiten. Das bekannteste Werkzeug dazu ist regedit.exe. Unter Windows NT und Windows XP existiert beispielsweise noch das Werkzeug regedt32.exe. Alle diese Werkzeuge erlauben es auf die Strukturen und Daten der Registry lesend sowie manchmal auch schreibend zuzugreifen.

3.1. Hives

Die Registry ist dabei in Hives aufgeteilt. Jeder dieser Hives hat eine bestimmte Aufgabe und beinhaltet Konfigurationsdaten. Folgende Hives

Tabelle 1 Standard Registry Hives

Hive Bezeichnung	Beschreibung
HKEY_CLASSES_ROOT (HKCR)	Dieser Schlüssel nicht ein logischer Schlüssel und verweist auf HKEY_LOCAL_MACHINE\Software\Classes. Dabei werden aber die Schlüssel unter HKEY_USERS/<UserSID>_Classes ebenfalls eingeblendet. Hier legt Windows die Informationen ab, welche Datei-Endungen mit welchen Symbolen dargestellt werden und mit welchen Programmen sie geöffnet werden können.
HKEY_USRS (HKU)	Dieser Schlüssel beinhaltet die Benutzer-Hives aller Benutzer auf dem System sowie der Standardschlüssel einiger systemrelevanter Benutzerkonten. Die Unterschlüssel tragen als Namen die eindeutige SID des Benutzers zu dem sie gehören. Um den Zugriff für Programme auf den korrekten Schlüssel des angemeldeten Benutzers zu erleichtern wird dieser auch unter dem Wurzel-Schlüssel HKEY_CURRENT_USER zur Verfügung gestellt.
HKEY_CURRENT_USER (HKCU)	Beim ersten Login wird dieser Schlüssel vom „default“ Benutzer übernommen und kopiert. Dieser Schlüssel beinhaltet die persönlichen Einstellungen des aktuell angemeldeten Benutzers. Der Inhalt wird konsequenterweise im Profil des Benutzers abgelegt und bei der Profilsicherung mit gesichert. Ein Benutzer ohne Administrator-Rechte darf nur in diesen Zweig der Registry schreiben. Hier können viele der globalen Einstellungen (unter HKEY_LOCAL_MACHINE) „überschrieben“ werden um Windows den persönlichen Bedürfnissen anpassen zu können.
HKEY_LOCAL_MACHINE (HKLM)	Hier werden globale Konfigurationsdaten abgelegt. Die hier gemachten Einstellungen gelten für alle Benutzer des Systems. Ein Benutzer ohne Administrator-Rechte darf nicht in diesen Schlüssel (oder deren Unterschlüssel) schreiben.

Unter dem Wurzel-Schlüssel HKEY_LOCAL_MACHINE werden noch weitere Hives eingebunden. Da diese zu einem späteren Zeitpunkt noch wichtig werden sollen sie hier kurz aufgeführt werden.

Tabelle 2 HKLM Hives

Hive Bezeichnung	Beschreibung
HKEY_LOCAL_MACHINE\ SAM	Der Security Accounts Manager (SAM) Hive beinhaltet Informationen über die Benutzerkonten, Gruppen und Berechtigungen. Dieser Hive ist naturgemäss am besten gegen Benutzermanipulationen geschützt. Dieses Dokument beschäftigt sich hauptsächlich mit den Daten in diesem geschützten Bereich.
HKEY_LOCAL_MACHINE\ HARDWARE	Dieser Hive beinhaltet für Windows relevante Informationen über die Hardware und deren Konfiguration. Mit diesen Informationen werden wir uns hier nicht näher befassen.
HKEY_LOCAL_MACHINE\ SECURITY	
HKEY_LOCAL_MACHINE\ SOFTWARE	Unter diesem Schlüssel werden die globalen Konfigurationsdaten der Installierten Software abgelegt.
HKEY_LOCAL_MACHINE\ SYSTEM	Dieser Hive beinhaltet systemspezifische Konfigurationsdaten. Insbesondere enthält er zwei so genannte „Control Sets“. Diese sind doppelt ausgelegt um im Falle einer Beschädigung oder einer falschen Konfiguration in einem der Control Sets auf die letzte lauffähige Variante zurückgreifen zu können. Der Schlüssel HKEY_LOCAL_MACHINE\ SYSTEM\CurrentControlSet ist dabei eine Verknüpfung mit dem aktuell aktiven Control Set.

3.2. Schlüssel und Werte

Die eigentlichen Daten werden in der Registry in form von Werten abgelegt. Man kann sich die Struktur analog zu einem Dateisystem vorstellen. Die Schlüssel entsprechen dabei den Ordnern bzw. Verzeichnissen und die Werte entsprechen den Dateien. Dabei werden verschiedene Werte-Typen unterschieden:

Tabelle 3 Registry Datentypen

Datentyp	Bezeichnung	Beschreibung
REG_SZ	Zeichenfolge/String	Alphanumerische Zeichenfolge
REG_BINARY	Binärwert	Binärer wert aus 0 und 1 (meist hexadezimale Notation).
REG_DWORD	Boolean	Enthält entweder 1 (true) oder 0 (false)
REG_MULTI_SZ	String-Array	Enthält mehrere Zeichenfolgen die durch ein Null-Byte getrennt sind. Ähnlich einem Array
REG_EXPAND_SZ	Expandierbare Variable	Enthält eine Zeichenkette mit ersetzbaren Zeichen (Umgebungsvariabeln). Beispielsweise wird der Wert „%windir%\system32“ beim Auslesen zu „C:\WINDOWS\system32“.

3.3. Binäre Hive Struktur

Leider ist die Struktur der Registry Hives nur sehr schlecht und lückenhaft dokumentiert. Glücklicherweise haben wir unter [SAM] eine brauchbare Dokumentation gefunden. Das Dokument kann die Struktur zwar auch nur lückenhaft dokumentieren aber für unsere Zwecke reicht dies gerade so aus. Für diese Arbeit waren insbesondere die Beschreibungen im Kapitel „REGISTRY STRUCTURE“ relevant.

Wir möchten an dieser Stelle nicht den Inhalt des Dokumentes kopieren sondern weiterführende und erklärende Informationen geben. Insbesondere weil die Quelle weder selbsterklärend noch absolut vollständig ist. Die Informationen ergeben keinesfalls eine lückenlose Dokumentation des Dateiformates sondern erklären lediglich die Grundlagen.

Eine Hive Datei ist in logische Blöcke zu 4096 Bytes (4kB) aufgeteilt. Die Daten sind in so genannte Zellen (Englisch: Cell) aufgeteilt. Nachfolgend werden die wichtigsten Zellen-Typen aufgelistet. Öffnet man die SAM Datei mit einem HEX-Editor ist der Datentyp der entsprechenden Felder zu beachten. Intel Prozessoren arbeiten im dem Little-Endian Modus (siehe auch [ENDIAN]). Deshalb sind die Werte in umgekehrter Byte-Reihenfolge abgelegt. Dies trifft allerdings nicht ganz für alle Werte zu da einige als Byte-Array abgelegt sind. Beispielsweise wird die Zeichenkette „regf“ im Header als Byte-Array abgelegt und nicht als 4-Byte Integer codiert. Die Kennungen der Zellen (z.B. „kn“) sind dabei als 2-Byte Short abgespeichert und erscheinen im Bytestrom deshalb als „nk“.

Bei Offset Angaben ist zusätzlich zu beachten, dass diese Relativ zum ersten HBin Eintrag angegeben werden. Somit muss zu allen Offsets immer 0x1000 addiert werden.

Tabelle 4 Registry Zellen

Zelle	Identifikation	Beschreibung
Header	regf	Steht am Anfang einer Hive Datei und beinhaltet Informationen, die vom NT Configuration Manager verwendet werden. Hier befinden sich noch keine Registry Schlüssel oder Werte.
HBin	hbin	Diese Einträge befinden sich jeweils nach 4096 Bytes (0x1000) im Datenstrom und dienen zur Segmentierung und effizienteren Verwaltung durch den Configuraiton Manager. Die Länge beträgt immer 32 Bytes und anschliessend folgen die in der Zelle abgelegten Daten.
Knoten	kn	Dies ist der wichtigste Zellentyp. Eine Knotenzelle entspricht einem Registry Schlüssel. Dieser kann Verweise auf die folgenden Zellen enthalten: • Wertlisten-Zelle • Unterschlüssel-Listen Zelle • Audit-Zelle • Klassen-Zelle Der Wurzelknoten (Root-cell) befindet sich gleich nach dem ersten hbin Eintrag also bei Offset 0x1020.
Unterschlüssel-Liste	kl, fl, hl	Diese Zellen enthalten eine Liste von „Links“ zu Unterschlüsseln eines Knotens. Zusätzlich zu den Links (Offset Adressen der Unterschlüssel) sind die ersten vier Buchstaben des Unterschlüssels eingetragen. So lässt sich in den meisten Fällen das komplette Auslesen eines Unterschlüssels zur Bestimmung des Namens verhindern.
Werteliste		Diese Liste hat keine Identifikation sondern besteht lediglich aus einem Array von Offset Adressen (4 Byte Integer). Die ersten 4 Bytes geben zwar die Länge des Arrays an. Über diese Länge lässt

		sich aber nicht die Anzahl der verknüpften Werte ermitteln. Nach ein paar fehlgeschlagenen Versuchen mussten wir feststellen, dass nur die Anzahl der Werte, die im Knoten eingetragen sind massgeblich sind. Die Länge der Werte-Liste kann aber auch länger angegeben sein und somit noch ungültige/leere Verweise umfassen.
Sicherheit	ks	Hierbei handelt es sich um die Berechtigungen auf den Registry Einträgen. Diese werden für unser Projekt nicht benötigt.
Class		Auch diese Zelle hat keine Identifikation. Die Class Einträge werden von keinem Windows Registry Tool angezeigt. Die Komponenten des Bootkey/Syskey werden in Class Zellen gespeichert.
Werte	kv	Hier werden die effektiven Werte gespeichert. Die Unterschlüssel-Liste verweist jeweils auf eine solche Werte-Zelle. Die Daten können dabei auf zwei unterschiedliche Arten abgelegt sein. Entweder sind sie direkt in der Werte-Zelle abgelegt (inline) oder verlinkt. Im zweiten Fall enthält die Zelle ein Daten-Offset an dem sich die Daten befinden.
Daten		Diese Zelle hat keine Identifikation und wird von den Werte-Zellen verlinkt. Die ersten 4 Bytes (Integer) definieren die Länge der Daten.

Vereinfacht gezeigt sieht ein Registry Baum folgendermassen aus:

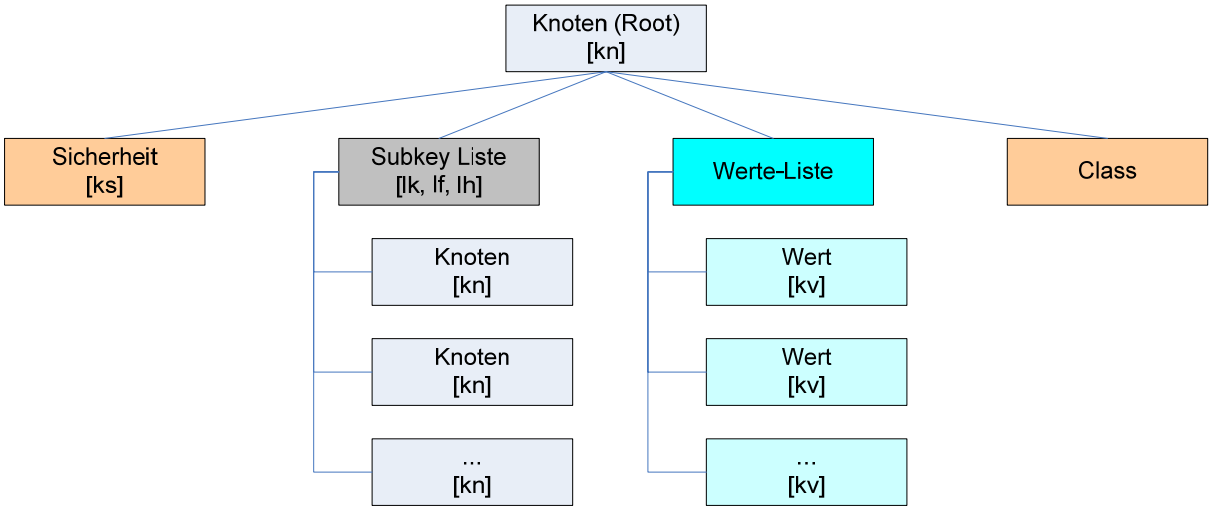

Abbildung 1 Registry Baum

Die verschiedenen Zellen-Typen deuten bereits darauf hin, dass ein Registry Hive nicht einfach zu lesen ist. Diese Befürchtung bestätigte sich auch bei der Entwicklung der Registry Lesefunktion von „Krypto**N**ie**T**e“. Eine Schwachstelle des Dateiformates scheint insbesondere die redundante Ablage verschiedener Daten dar. Beispielsweise schreit die Speicherung der Anzahl Werte in der Knotenzelle und die indirekte Speicherung derselben Information in der Unterschlüssel-Liste geradezu nach Inkonsistenz. In der Unterschlüssel-Liste entsteht auch die Gefahr, dass die gespeicherten ersten vier Buchstaben des Unterschlüsselnamens nicht mit dem realen Wert im Unterschlüssel übereinstimmen.

In Verbindung mit Java stellen sich zudem noch einige weitere Probleme wie die Umwandlung der binären Datentypen, die Byte-Reihenfolge (Java arbeitet mit Big-Endian) und der fehlenden „unsigned“ Datentypen.

4. SYSKEY

Die Syskey Verschlüsselung wurde mit Windows NT 4.0 Service Pack 2 eingeführt. Damals war es noch optional und musste explizit aktiviert werden. Das Ziel war es die Passwortdatenbank (SAM) durch eine 128 bit Verschlüsselung selbst bei unerlaubtem Zugriff vor Angriffen zu schützen. Das Hauptproblem Dabei besteht darin, dass die Verschlüsselung transparent sein muss und der Schlüssel somit zwangsläufig auf der lokalen Festplatte abgelegt wird. Somit ist er natürlich auch von weiteren Programmen auffindbar. Wir werden beweisen, dass bei reinem physikalischen Zugriff auf den Rechner die Verschlüsselung geknackt werden kann.

4.1. Kryptografische Grundlagen

Auf die kryptografische Grundlage wird nicht gross eingegangen, da nur Standard Verschlüsselungsverfahren eingesetzt werden. Dennoch werden in diesem Kapitel alle verwendeten Verfahren aufgelistet und kurz beschrieben. Die Algorithmen werden jedoch nicht im detail erklärt. Diese können aus zahlreichen Beiträgen im Internet studiert werden.

4.1.1. MD5

MD5 bedeutet Message Digest Algorithm 5 und ist eine Hashfunktionen, die von Professor Ronald L. Rivest am MIT entwickelt wurden. Er wurde als sicheren Ersatz für MD4 entwickelt, als Analysen dessen Unsicherheit ergaben.

Das folgende Beispiel aus dem Wikipedia Artikel über [MD5] zeigt den 128 Bit langen MD5 Hashwert des angegebenen Strings, welcher als 32-stellige Hexadezimalzahl notiert wird:

```
md5("Franz jagt im komplett verwahrlosten Taxi quer durch Bayern") =
a3cca2b2aa1e3b5b3b5aad99a8529074
```

Eine kleine Änderung der Nachricht erzeugt (mit sehr großer Wahrscheinlichkeit) einen komplett anderen Hash. Mit Frank statt Franz (nur ein Buchstabe verändert) ergibt sich:

```
md5("Frank jagt im komplett verwahrlosten Taxi quer durch Bayern") =
7e716d0e702df0505fc72e2b89467910
```

Der Hash eines Strings der Länge Null ist:

```
md5("") = d41d8cd98f00b204e9800998ecf8427e
```

Unter der Quelle [MD5] erfährt man mehr über den Algorithmus und deren Einsatzgebiete.

4.1.2. RC4

RC4 bedeutet Ron'sCipher 4 und ist eine einfache Stromchiffre. Der Algorithmus wurde über viele Jahre geheimgehalten. Im September 1994 veröffentlichte eine anonyme Person einen Algorithmus, der identische Ergebnisse erzeugte wie RC4. Dieser Quelltext verbreitete sich schnell und wurde ausgiebig analysiert und diskutiert.

Im Gegensatz zu DES ist die Länge des Schlüssels variabel und kann bis zu 2048 Bit (256 Zeichen) betragen. Es wird immer ein Byte (ein Zeichen) auf einmal verschlüsselt. Der Algorithmus ist sowohl einfach und kann besonders effizient programmiert werden.

RC4 läuft in drei Schritten ab:

- Initialisierung einer S-Box. Dazu wird ein Array mit 256 Zeichen gefüllt und die Feldelemente mittels Schlüssel untereinander vertauscht.
- Danach wird für jeden Buchstaben des Eingabetextes ein Zufallsbyte aus der S-Box ermittelt, wobei die Elemente im Array jedes mal vertauscht werden.
- Schliesslich werden das ermittelte Zufallsbyte und das Textzeichen durch XOR miteinander verknüpft.

Trotz der Einfachheit des Algorithmus, wurden bisher keine Schwächen entdeckt. Da die Verschlüsselung in keiner Form vom zu verschlüsselnden Text abhängig ist, ist der Algorithmus für Klar- oder Ge-

heimtextangriffe anfällig. Es ist daher zu empfehlen einen Schlüssel nicht zu oft zu verwenden (Siehe Problematik bei der Wireless LAN WEP Key-Verschlüsselung).

Mehr über RC4 findet man unter [RC4] und im restlichen Web.

4.1.3. DES

DES bedeutet Data Encryption Standard und wurde als offizieller Standard für die US-Regierung. DES wird auf Grund der verwendeten Schlüssellänge von 56 Bits für viele Anwendungen als nicht ausreichend sicher erachtet. Deshalb wird durch mehrmalige Anwendung von DES die Schlüssellänge erhöht. Am häufigsten wird heute DES als 3DES eingesetzt und als sicher taxiert.

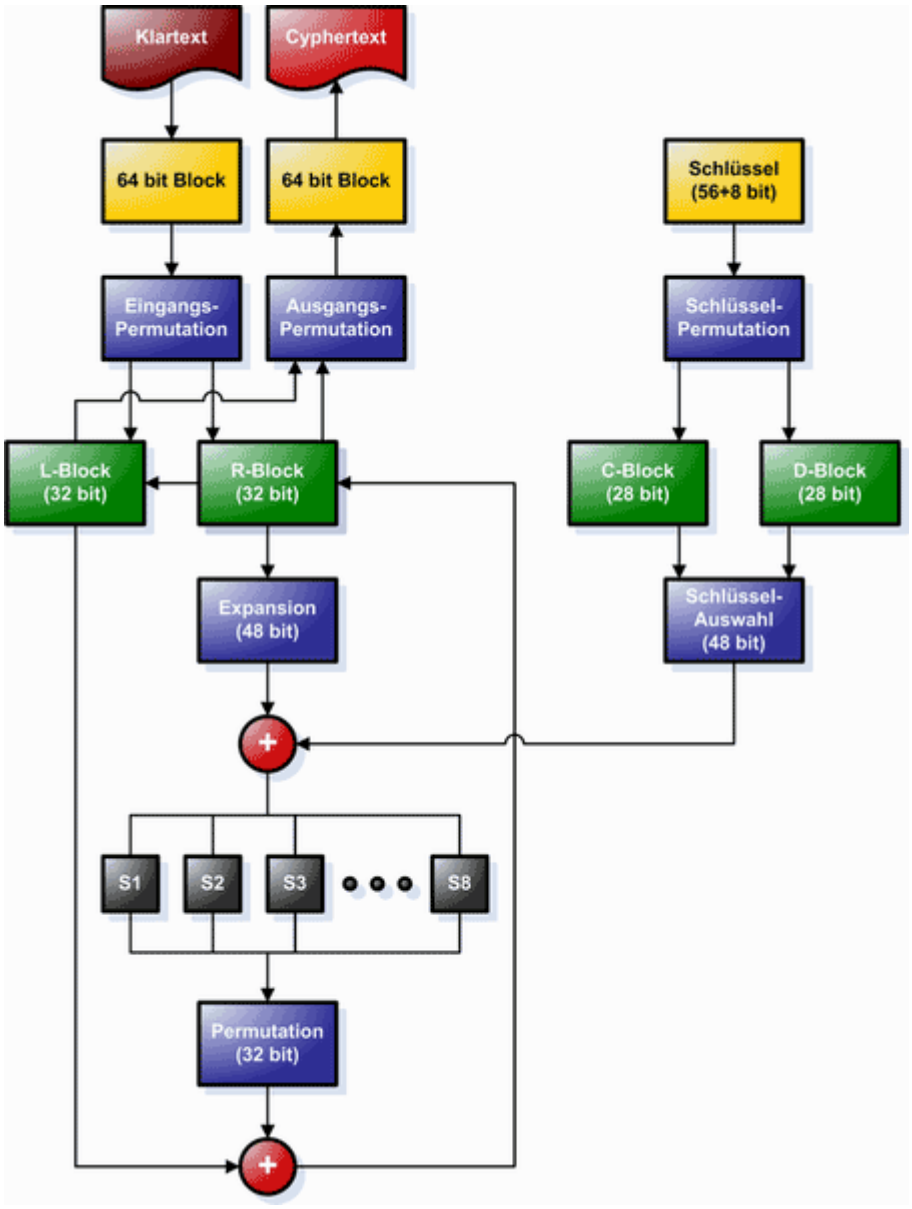

Abbildung 2 DES

Mehr über den DES Algorithmus findet man in der Quelle [DES].

4.2. Syskey Verschlüsselung erkennen

Zuerst muss festgestellt werden ob die Syskey Verschlüsselung aktiv ist. Dazu kann folgender Registry-Wert ausgelesen werden:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\"SecureBoot"
Type: dword:0000000X
```

Die Werte sehen wie folgt aus:

Tabelle 5 Syskey Operations-Modi

Wert	Bedeutung
0	Syskey ausgeschaltet.
1	Speicherung des Schlüssels auf dem System. Dies erlaubt den unbeaufsichtigten Start des Rechners.
2	Syskey wird aus Passwort (abgefragt beim Systemstart) erzeugt. Der Nachteil dabei ist, dass beim Systemstart ein Passwort eingegeben werden muss.
3	Der Syskey wird auf einer Diskette abgelegt. Diese muss beim Systemstart eingelegt werden. Der Nachteil besteht ebenfalls darin, dass beim Start die Diskette eingelegt werden musss.

4.3. Syskey konfigurieren

Wie aus den Werten aus Kapitel 4.2 zu entnehmen ist kann die Syskey Verschlüsselung in drei verschiedenen Operations-Modi betrieben werden:

- Syskey lokal auf dem Rechner speichern. Dies ist die Standardmethode und auch die einzige Methode in der keine Benutzeraktionen notwendig sind. Der Nachteil liegt natürlich in der lokalen Speicherung und somit der Möglichkeit den Schlüssel auslesen zu können.
- Syskey per Passworteingabe beim Systemstart abfragen. Diese Methode wäre zwar viel sicherer aber bietet den Nachteil, dass man sich ein weiteres Passwort merken muss und dieses bei jedem Systemstart einzugeben hat. Wird ein PC von mehreren Personen verwendet müssen alle das Passwort kennen, was selten praktikabel ist.
- Syskey wird auf einer Diskette abgelegt. Da viele PCs heute nicht mal mehr ein Diskettenlaufwerk besitzen ist diese Methode häufig nicht einsetzbar. Windows unterstützt nämlich wirklich nur Diskettenlaufwerke und keine weiteren Speichermedien. Beim Systemstart muss zudem natürlich immer die entsprechende Diskette eingelegt werden. Ein automatischer Systemstart ist somit nicht möglich und bei einem Verlust oder Defekt der Diskette wird ein Login unmöglich.

Der Betriebsmodus kann unter dem laufenden Windows mit dem Programm „syskey.exe“ geändert werden. Dies ist natürlich nur Administratoren erlaubt. Um das Programm zu starten füllt man am besten unter Start -> Ausführen die Zeichenkette „syskey“ ein und klickt auf OK. Danach erscheint der nachfolgend aufgeführte Dialog.

Abbildung 3 Syskey Konfiguration

Nach einem Klick auf „Update“ erscheint dann das Fenster in dem der Betriebsmodus der Syskey Verschlüsselung konfiguriert werden kann.

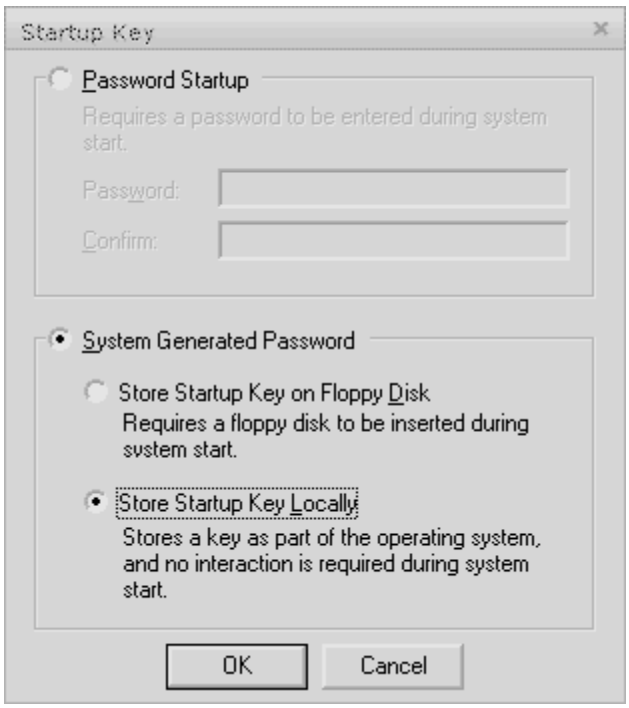

Abbildung 4 Syskey Operations-Modi

Nach der Auswahl des Disketten-Modus und einem Klick auf OK wird man aufgefordert eine Diskette einzulegen.

ACHTUNG: Arbeitet man an einem PC ohne Diskettenlaufwerk bleibt das Programm in einer Endlosschleife hängen bis eine Diskette eingelegt wird.

HINWEIS: Durch den Export des Syskey auf Diskette wird nicht der momentan aktive Syskey exportiert sondern ein neuer generiert. Windows entschlüsselt also alle Passwörter, generiert einen neuen Syskey, speichert diesen auf der Diskette und verschlüsselt dann alle Passwörter wieder.

4.4. Syskey Verschlüsselung (Algorithmus)

Windows speichert zwei Passwort-Hashes (NT und LM) im SAM Registry Hive. Beide sind nahezu identisch zusätzlich verschlüsselt um ein einfaches Auslesen und Knacken zu verhindern.

Der Algorithmus für den LM Hash sieht folgendermassen aus:

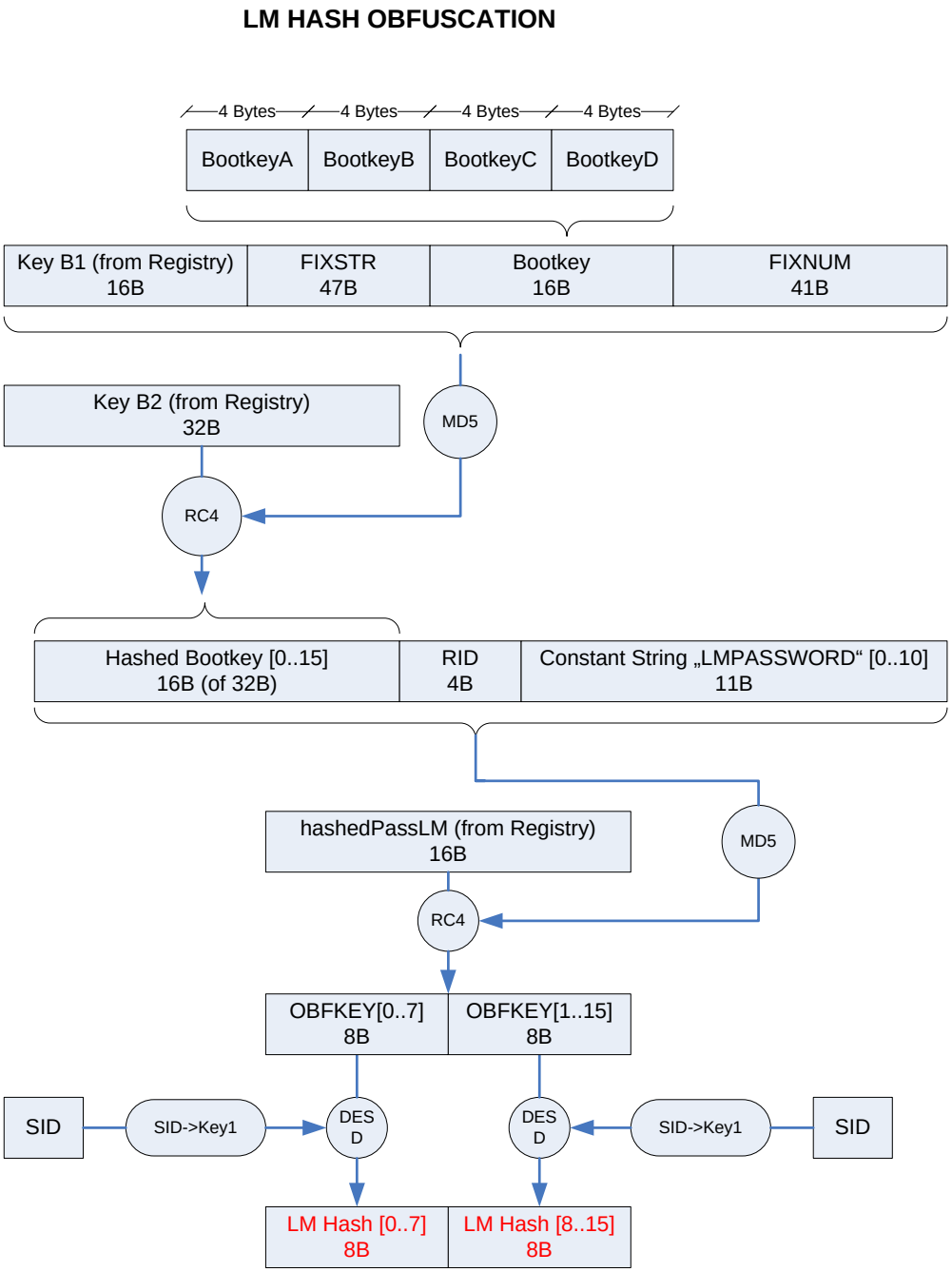

Abbildung 5 LM Hash obfuscation

Die Verschlüsselung des NT Hashes sieht fast identisch aus. Lediglich der zur Verschlüsselung des Hashes verwendete Schlüssel (MD5 Checksumme) wird mit Hilfe eines anderer anderen statischen Zeichenkette „NTPASSWD“ anstatt „LMPASSWD“ gebildet:

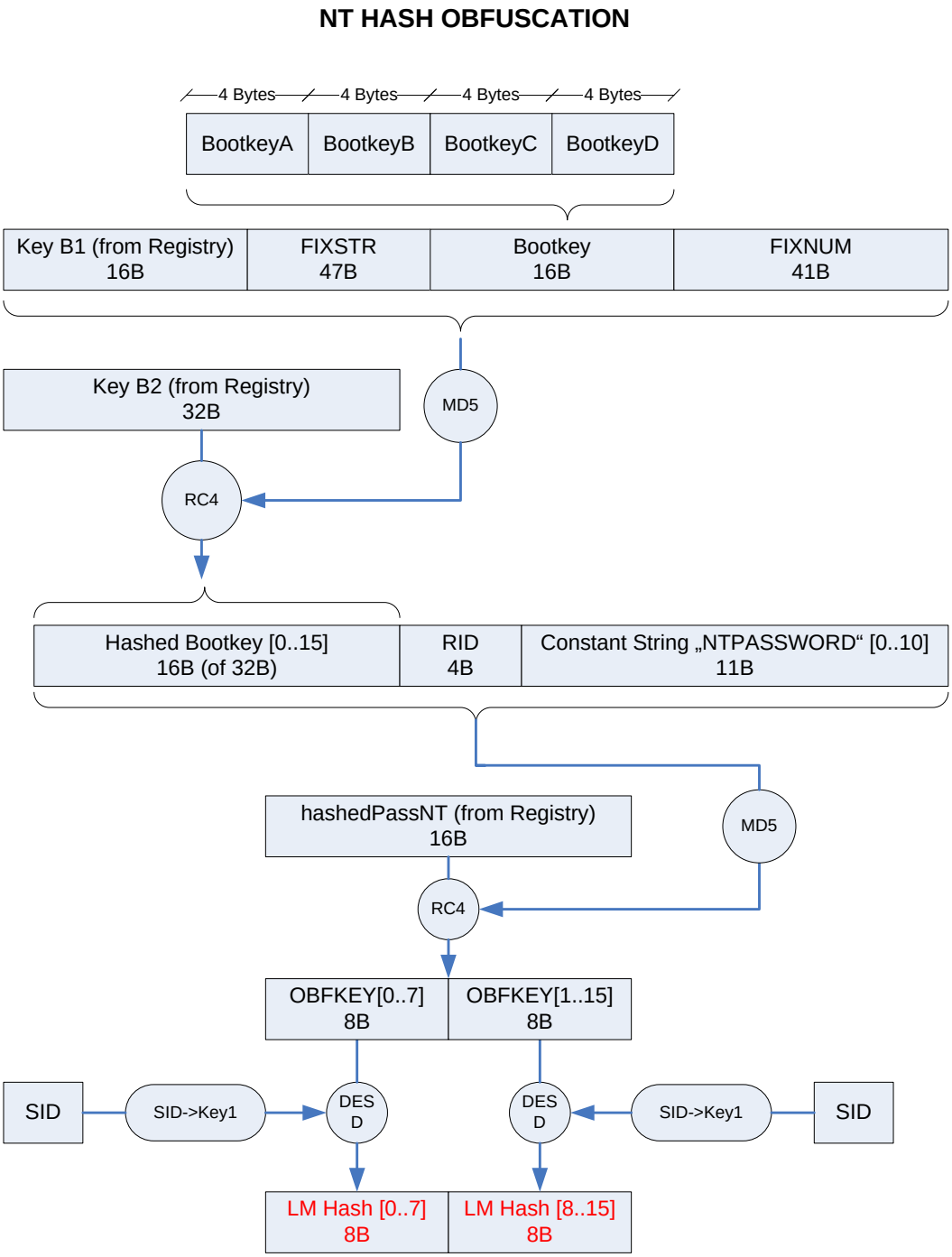

Abbildung 6 NT Hash obfuscation

Tabelle 6 Syskey Legende

Bezeichnung	Inhalt
BootkeyA	Die ersten 8 Unicode-Zeichen aus den Class-Daten des Schlüssels ControlSet001\Control\Lsa\JD interpretiert als HEX-String ergeben einen Integer-Wert (4 Bytes). Dieser wird im Little Endian Format ins Bootkey Array (16 Bytes) abgelegt (Bytes 1-4).
BootkeyB	Die ersten 8 Unicode-Zeichen aus den Class-Daten des Schlüssels ControlSet001\Control\Lsa\Skew1 interpretiert als HEX-String ergeben einen Integer-Wert (4 Bytes). Dieser wird im Little Endian Format ins Bootkey Array (16 Bytes) abgelegt (Bytes 5-8).
BootkeyC	Die ersten 8 Unicode-Zeichen aus den Class-Daten des Schlüssels ControlSet001\Control\Lsa\GBG interpretiert als HEX-String ergeben einen Integer-Wert (4 Bytes). Dieser wird im Little Endian Format ins Bootkey Array (16 Bytes) abgelegt (Bytes 8-12).
BootkeyD	Die ersten 8 Unicode-Zeichen aus den Class-Daten des Schlüssels ControlSet001\Control\Lsa\Data interpretiert als HEX-String ergeben einen Integer-Wert (4 Bytes). Dieser wird im Little Endian Format ins Bootkey Array (16 Bytes) abgelegt (Bytes 12-16).
Key B1 [16 Bytes]	Byte 112 bis 128 vom Wert “SAM\Domains\Account\F” (SAM Registry Hive)
Key B2 [32 Bytes]	Byte 128 bis 160 vom Wert “SAM\Domains\Account\F” (SAM Registry Hive)
FIXSTR	!@#$%^&*()qwertyUIOPAzxcvbnmQQQQQQQQQQQQ)(*@&%
FIXNUM	0123456789012345678901234567890123456789
LMPASSWORD	Konstanter String „LMPASSWORD“. Achtung: Es werden 11 Bytes verwendet. Der String ist aber nur 10 Bytes lang. Deswegen muss ein Null-Byte „\0“ angehängt werden.
NTPASSWORD	Konstanter String „NTPASSWORD“. Achtung: Es werden 11 Bytes verwendet. Der String ist aber nur 10 Bytes lang. Deswegen muss ein Null-Byte „\0“ angehängt werden.
RID	Relative Identifier. Dieser Wert entspricht dem Schlüsselnamen des Benutzers unter „SAM\Domains\Account\Users“ (SAM Registry Hive) interpretiert als HEX Wert. Dieser Wert muss in ein 4-Byte Integer Wert im Little Endian Format abgelegt werden.
hashedPassLM	Verschlüsselter LM-Hash aus der Registry: Pfad: SAM\Domains\Account\Users\<RID>\V An der Position 0x9C befindet sich der interne Offset wo der Hash abgelegt ist. Zu diesem Wert muss dann noch 0xCC addiert werden um die Position des LM Hashes zu ermitteln. Der Wert ist 16 Bytes lang. Offset: Int(0x9C) + 0xCC
hashedPassNT	Verschlüsselter NT-Hash aus der Registry: Pfad: SAM\Domains\Account\Users\<RID>\V Gleich hinter dem Hash für das LM Passwort befindet sich der Hash für das NT Passwort (plus einem Offset von 0x08). Der Wert ist 16 Bytes lang. Offset: Int(0x9C) + 0xCC + 0x10 + 0x08.

SID	Entspricht der RID (Integer)
SID->Key	Durch diese Funktion wird die SID in einen DES Schlüssel umgewandelt. Die SID wir dabei durch verschiedene bitweise Schiebeoperatoren und durch eine Permutation umgewandelt. Die genaue Umwandlung ist im Code ersichtlich.

5. Passwort Recovery

Um Windows NT-Passwörter zu entschlüsseln, sind die oben beschriebenen Vorgänge nötig. Als aller erstes ist der SYSKEY auszulesen, um die LM- und NT-Hash's aus der SAM Datei zu erhalten. Danach kann mit einer Dictionary-, Bruteforce einer kombinierten Attacke das NT-Passwort entschlüsselt werden.

Da die beiden Registry Files SAM und SYSTEM während der Laufzeit des Betriebssystems für jeglichen Zugriff gesperrt sind, müssen diese passiv von der Festplatte gelesen werden. Um dies zu erreichen gibt es verschieden Verfahren. Theoretisch könnte man von einem Floppy booten, und die Dateien auf Diskette kopieren. Dies ist in der Praxis nicht möglich, da der SYSTEM Hive mehrere MB gross ist. Am einfachsten geht es mit einem OS, welches direkt vom CD-ROM Laufwerk gebootet werden kann. Auch hier gibt es verschieden Möglichkeiten. Dabei ist entscheidend, dass das OS auf Datenträger mit NTFS Dateisystem zugreifen kann.

Wir werden eine Version des Windows PE, Bart PE, verwenden, damit das Passwort Recovery in einem Zug durchgeführt werden kann und das System nicht mehrmals neu gestartet werden muss.

5.1. Windows PE/Bart PE

Der Bart's PE-Builder wurde von Bart Lagerweij entwickelt. Mit dem Bart's PE-Builder kann eine bootfähige Windows XP CD/DVD mit integrierten Tools (Plugins) erstellt werden, die zusammen eine geniale Rettungs CD bilden. Dabei können die Plugins selber konfiguriert werden. Für unsere Boot-CD werden wir verschiedene Plugins aus dem Security-Bereich erstellen und zusammenstellen. Neben Antiviren Programmen werden auch Analysetools und unser selbst entwickeltes Programm „Krypto**N**ie**T**e" einen Platz auf der CD finden.

5.2. Erstellen einer Bart PE Boot-CD

Um eine Bart PE Boot-CD zu erstellen, benötigt man folgende Komponenten:

- Ein lauffähiges Windows 2000/XP/2003 System
- Eine gültige WindowsXP oder WindowsServer 2003 Lizenz
- Bart's PE-Builder. Siehe [BartPE].
- Plugins

Folgend wird die Vorgehensweise für eine erste BartPE Bootdisk beschrieben:

- Zuerst wird die komplette Windows XP-CD auf die Festplatte in ein beliebiges Verzeichnis z.B. **D:\WinXP_Professional\PRO_WITH_SP2** kopiert.
- Danach kann der Bart PE Builder von der oben aufgeführten Webseite heruntergeladen und in ein Verzeichnis z.B. **D:\Projects\PEBuilder** extrahiert werden.
- In diesem Verzeichnis findet man nun die Datei pebuilder.exe, welche der Ausführung des PE-Builders dient.
- Nach der Ausführung von pebuilder.exe erscheint der Folgende Dialog:

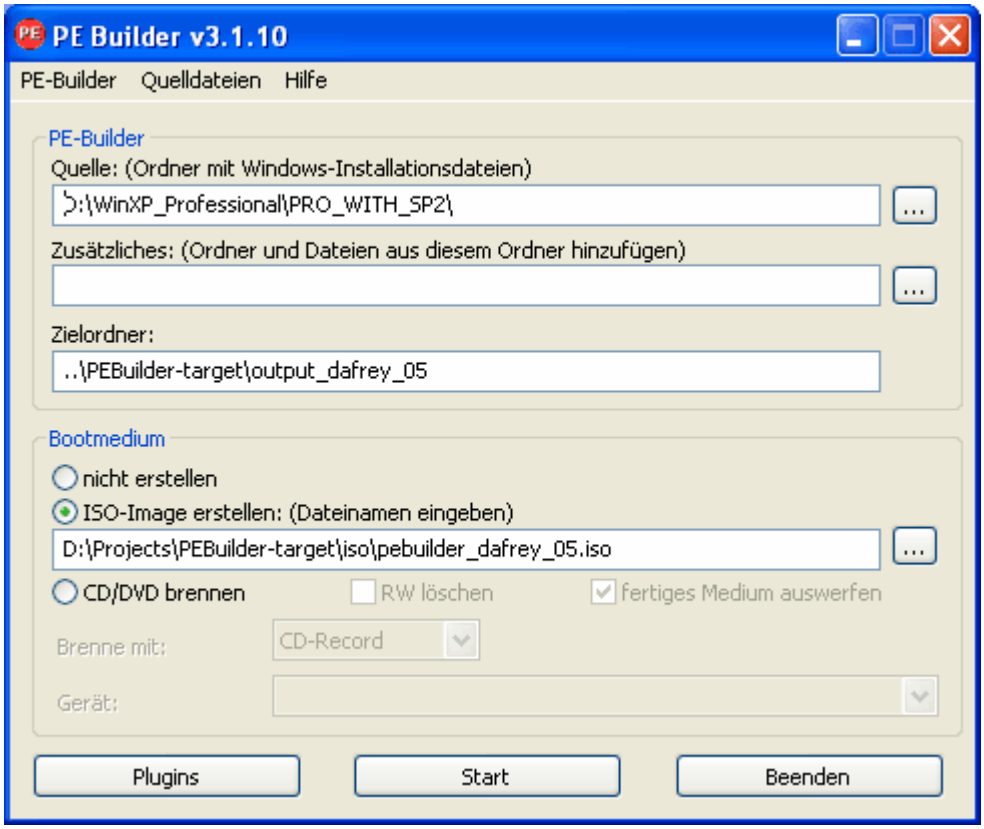

Abbildung 7 PE Builder

- Nun muss der Source Path der Windows XP CD angegeben werden, den Zielordner in welchem die Bart PE Files gespeichert werden sollen und den Speicherort für das ISO-Imagefile.
- Mit dem Button Plugins, lassen sich Plugins aktivieren / deaktiveren.
- Mit Start wird der Build-Vorgang gestartet.

5.3. Bart PE Plugins

Um Programme oder Tools der WinPE Boot-CD hinzuzufügen, müssen Plugins erstellt werden. Dieser Vorgang wird hier kurz beschrieben, damit eine Konfiguration einer angepassten Bart PE CD möglich ist.

Da wir das Windows XP-Plugin von Sherpya verwenden, müssen wir vom herkömmlichen Vorgehen etwas abweichen, um die neuen Programme zusätzlich der Startliste hinzuzufügen.

Die Plugins befinden sich im Verzeichnis Plugin. Dieses ist bei uns in 4 Unterverzeichnisse unterteilt:

Verzeichnis	Beschreibung
plugin\application\	In diesem Verzeichnis werden alle Plugins für Programme / Applikationen erstellt.
plugin\drivers\	In diesem Verzeichnis werden alle Plugins für Programme / Applikationen erstellt.
plugin\system\	Alle PE-spezifischen Systemkonfigurations Plugins (bsp. Sherpya WinXP) werden hier angelegt.
plugin\games\	Um sich während der manchmal langen Entschlüsselungszeit zu beschäftigen, werden in diesem Verzeichnis die gewünschten Spiele angelegt.

5.3.1. Plugin erstellen - KryptoNieTe

Um das Prinzip der Pluginkonfiguration genauer zu beschreiben, wird hier als Beispiel das Plugin für unser eigenes SAM-Decrypter Programm „Krypto**N**ie**T**e“ erläutert:

- Es wird ein neues Verzeichnis „custom_kryptonite“ unter plugin\application\ angelegt.
- In diesem Verzeichnis werden folgende Files und Ordner angelegt:

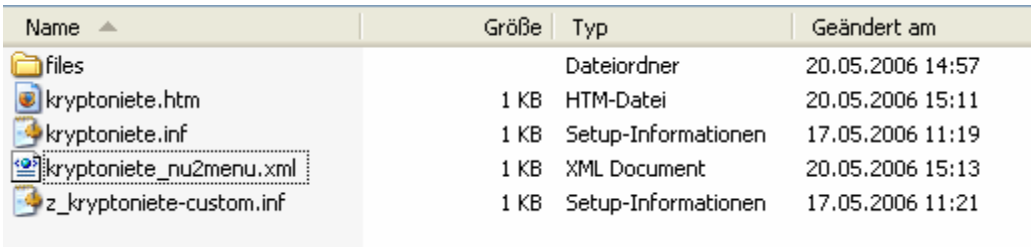

Abbildung 8 KryptoNieTe Plugin

Name	Beschreibung
Files	Verzeichnis mit den Programmfiles
kryptoniete.htm	Hilfe innerhalb PE-Builder
kryptoniete.inf	Plugin Definition
z_kryptniete-custom.inf	Plugin Definition (Benutzerdefinierte Einstellungen)

Im folgenden Abschnitt werden die beiden für uns relevanten *.inf Files genauer beschrieben.

Beschreibung des kryptoniete.inf File:

```
; PE Builder v3 plug-in INF file
; Created by Rainer Meier & Daniel Frey

[Version]
Signature= "$Windows NT$"

[PEBuilder]
Name="kryptoniete"
Enable=1
Help="kryptoniete.htm"
Version=1.0.0.0

[WinntDirectories]
a="Programs\kryptoniete",2

[SourceDisksFiles]
files\kryptoniete.jar=a,,1
files\kryptoniete.ico=a,,1

[Append]
nu2menu.xml, kryptoniete_nu2menu.xml
```

Inifile Section	Beschreibung
[PEBuilder]	Definition der allgemeinen Plugin Infomationen für PE (Name, Status, Hilfeseite und Version)
[WinntDirectories]	Zielverzeichnis auf dem Boot-Medium
[SurceDisksFiles]	Liste der Programmdateien mit Angabe des Zielverzeichnisses
[Append]	Erweiterung der nu2menu Struktur

Beschreibung des z_kryptoniete-custom.inf File:

```
; PE Builder v3 plug-in INF file
; Created by DaFrey

[Version]
Signature= "$Windows NT$"

[PEBuilder]
Name="WRR, custom"
Enable=1
Help="kryptoniete.htm"
Version=1.0.0.0

[Software.AddReg]
; Add to start menu
0x2,"Sherpya\XPEinit\StartMenu","Security\KryptoNieTe","
%ProgramFiles%\java\bin\javaw.exe|-jar
%ProgramFiles%\kryptoniete\kryptoniete.jar|%ProgramFiles%\kryptoniete\
kryptoniete.ico,0"
```

Inifile Section	Beschreibung
[PEBuilder]	Definition der allgemeinen Plugin Infomationen für PE (Name, Status, Hilfeseite und Version)
[Software.AddReg]	Hier wird das Programm dem Startmenu hinzugefügt. Programmaufruf: Filename | Parameter | und Icon

5.3.2. Java Runtime Plugin

Da unser SAM Decrypter in Java programmiert wurde, muss natürlich unter Windows PE eine Java Runtime zur Verfügung stehen. Auch dafür hat Sherpya bereits ein Plugin erstellt welches auf seiner Homepage verfügbar ist. Siehe dazu auch [Sherpya].

5.4. Anwendung

In diesem Kapitel wird die Verwendung unserer Boot-CD und das Passwort Recovery mit den beiden Programmen Krypto**N**ie**T**e und Cain & Abel beschreiben. Auf weitere integrierte Plugins wird an dieser Stelle nicht weiter eingegangen, da sie nicht Bestandteil der Laborarbeit waren.

5.4.1. Starten der BartPE Boot-CD

Damit das Betriebsystem von der CD auf demjenigen Rechner gestartet wird auf welchem das Passwort Recovery durchgeführt werden soll, muss einzig sichergestellt werden, dass im Bios die Einstellung, welche das Booten vom CD Rom Laufwerk erlaubt, aktiviert ist.

Danach kann die CD ins Laufwerk eingelegt und der Computer einschalten werden. Es sollte nun oben links die Meldung erscheinen, dass der Computer auf Hardwarekonfiguration untersucht wird.

Nach erfolgreichem Booten kommt die Meldung, ob Netzwerkkomponenten aktiviert werden sollen. Der Betrieb einer Netzwerkkarte ist jedoch nicht erforderlich, da wir lokales Passwort Recovery durchführen werden. Es sind jedoch Standard und Marvell Yukon Netzwerkkarten Treiber installiert. Sollte Ihre Netzwerkkarte nicht dabei sein, kann diese im PE-Build unter dem Ordner plugin\drivers\ hinzugefügt und das PE-Build neu erstellt werden.

Nach erfolgreicher Netzwerkkomponenteninstallation, ist das Betriebssystem geladen und betriebsbereit:

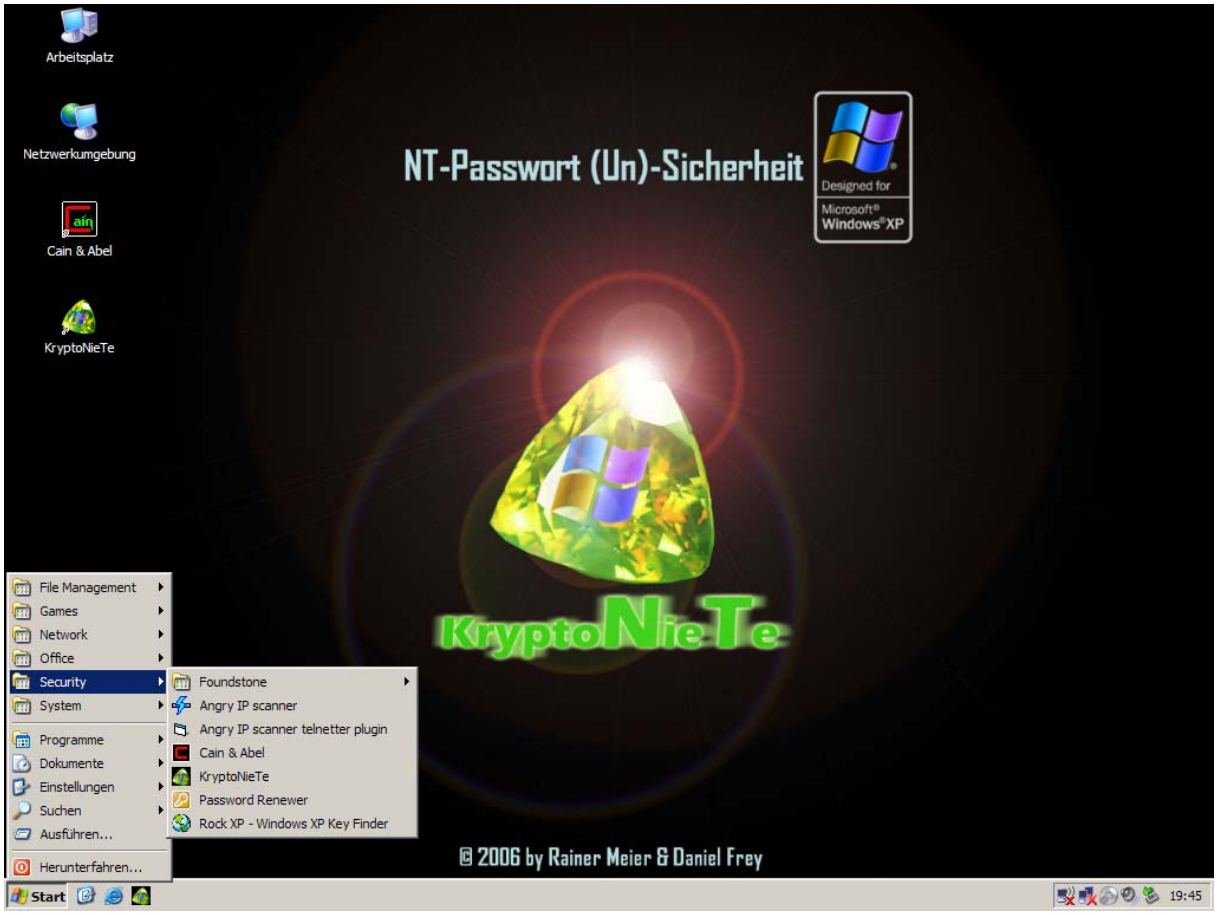

Abbildung 9 Boot CD Desktop

5.4.2. Auslesen der Passwort Hashes mit KryptoNieTe

Krypto**N**ie**T**e war die eigentliche Herausforderung dieser Laborarbeit. Obwohl es uns bewusst war, dass es Programme gibt, welche die Passwort Hashes aus der SAM Datenbank lesen können, haben wir unser eigenes Programm geschrieben. Damit war es uns möglich die SAM Verschlüsselung mit dem SYSKEY zu verstehen und sie zu beschreiben.

Das Programm befindet sich im Startmenu unter Security / Krypto**N**ie**T**e. Die SAM Datei befindet sich im Verzeichnis WINDOWS/system32/config/. Das Auslesen der Passwort Hashes aus der SAM Datei muss mittels syskey/bootkey geschehen.

Der Syskey kann auf drei verschiedene Varianten für die Entschlüsselung eingegeben werden:

1. Direktes auslesen aus dem SYSTEM hive. Dazu muss das File SYSTEM aus dem Verzeichnis WINDOWS/System32/Config/ ausgewählt werden (oder eine Kopie davon).
2. Einlesen aus einer Datei (wenn der Bootkey auf einem externen Medium gespeichert ist)
3. Manuelle Eingabe des Keys als HEX-String (32 Zeichen HEX ergeben 16Byte/128bit).

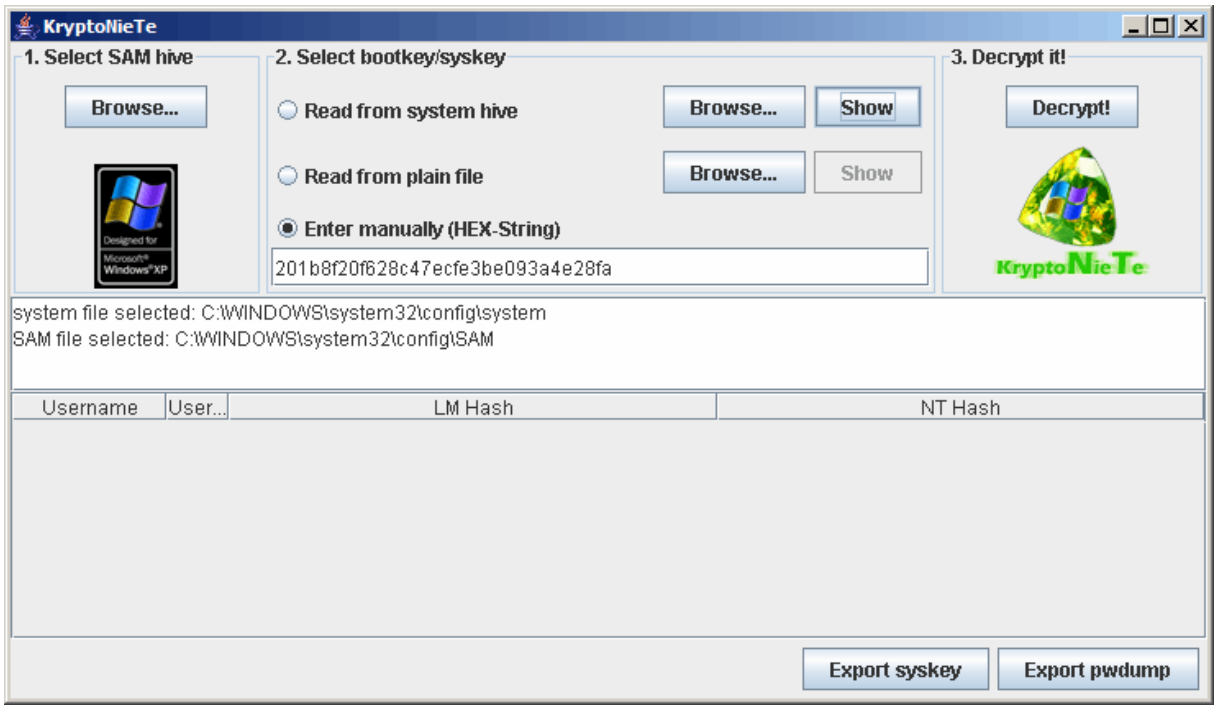

Abbildung 10 KryptoNieTe Hauptfenster

Nach dem Drücken des Button „Decrypt“, werden alle User mit UserID, LM- und NT Hash in der Tabelle abgebildet:

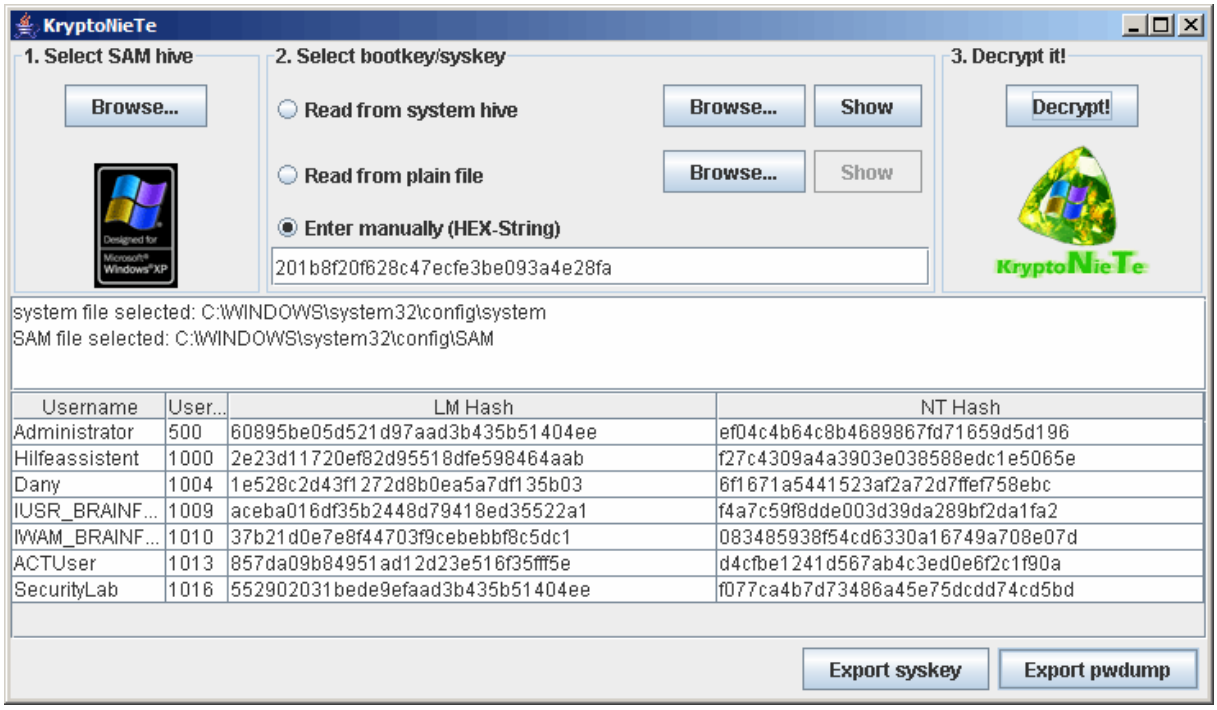

Abbildung 11 KryptoNieTe Entschlüsselte Hashes

Danach können die Passwort Daten mittels „Export pwdum“ in eine „PWDump“ Datei exportiert werden. Dabei handelt es sich um ein standardisiertes Format welches dem Passwortdatei-Format unter Unix (/etc/passwd) entspricht. Die Werte werden dabei durch Doppelpunkte voneinander getrennt. Diese PWDump Datei kann nachher in Cain & Abel eingelesen werden um die Hashes zu knacken entschlüsselt werden.

```
Administrator:500:60895be05d521d97aad3b435b51404ee:ef04c4b64c8b4689867
fd71659d5d196:::
Hilfeassistent:1000:2e23d11720ef82d95518dfe598464aab:f27c4309a4a3903e0
38588edc1e5065e:::
Dany:1004:1e528c2d43f1272d8b0ea5a7df135b03:6f1671a5441523af2a72d7ffef7
58ebc:::
IUSR_BRAINFOOD:1009:aceba016df35b2448d79418ed35522a1:f4a7c59f8dde003d3
9da289bf2da1fa2:::
IWAM_BRAINFOOD:1010:37b21d0e7e8f44703f9cebebbf8c5dc1:083485938f54cd633
0a16749a708e07d:::
ACTUser:1013:857da09b84951ad12d23e516f35fff5e:d4cfbe1241d567ab4c3ed0e6
f2c1f90a:::
SecurityLab:1016:552902031bede9efaad3b435b51404ee:f077ca4b7d73486a45e7
5dcdd74cd5bd:::
```

5.4.3. Entschlüsselung der Passwort Hashes mit Cain & Abel

Cain & Abel ist ein frei verfügbares Tool um diverse Arten von Passwörtern zu entschlüsseln. Wir werden uns nur auf den Passwort Recovery Teil konzentrieren und diesen näher Beschreiben.

Nach dem Start des Programms, kann unter „Cracker“ die LM & NTML Hashes angewählt werden um folgende Darstellung zu erhalten:

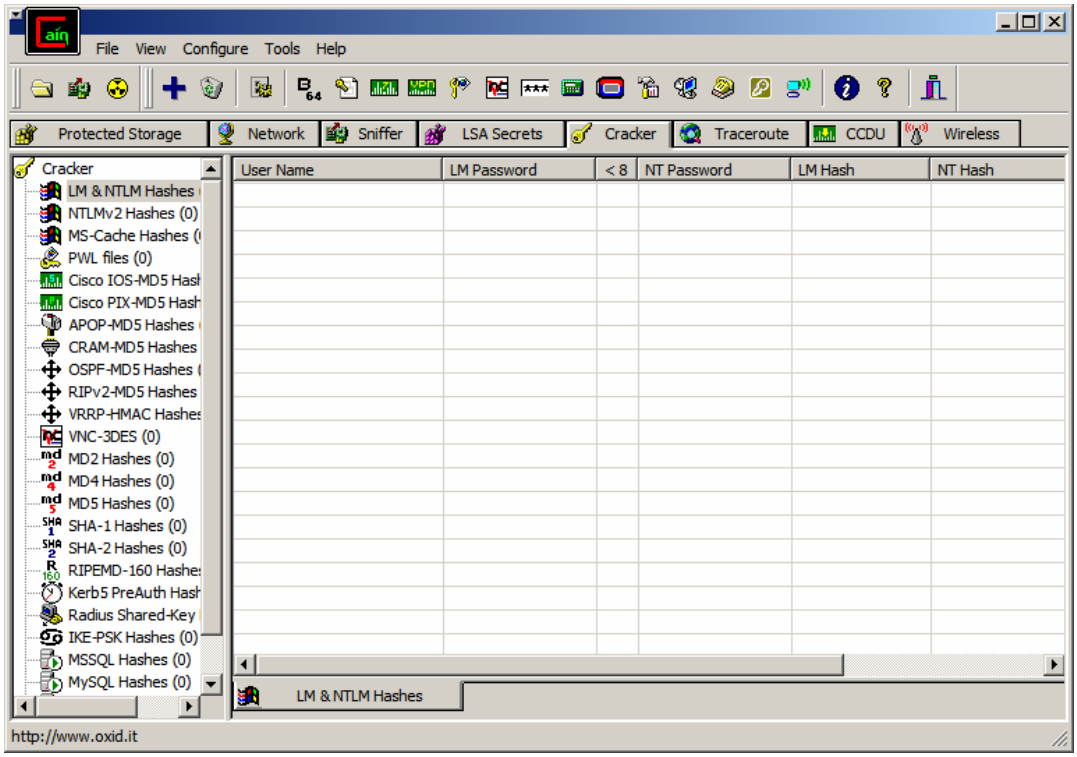

Abbildung 12 Cain Hauptfenster

Nun können die Passwort Hashes, welche wir mit Krypto**N**ie**T**e erstellt haben, aus dem Textfile mit dem + Zeichen importiert werden:

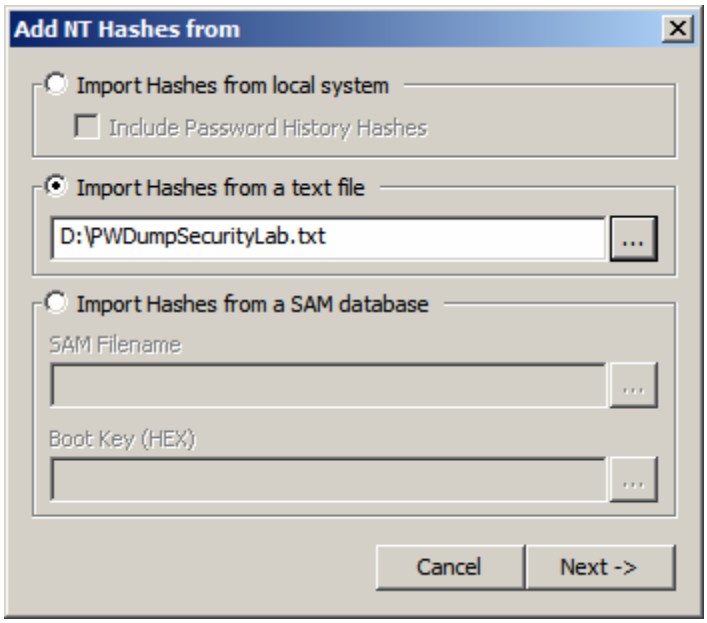

Abbildung 13 Cain Hash Import

Danach sind alle Benutzer und Passwort Hashes in der Liste eingetragen. Diejenigen, welche nicht von Interesse sind, können mit der Rechten Maustaste und Remove entfernt werden:

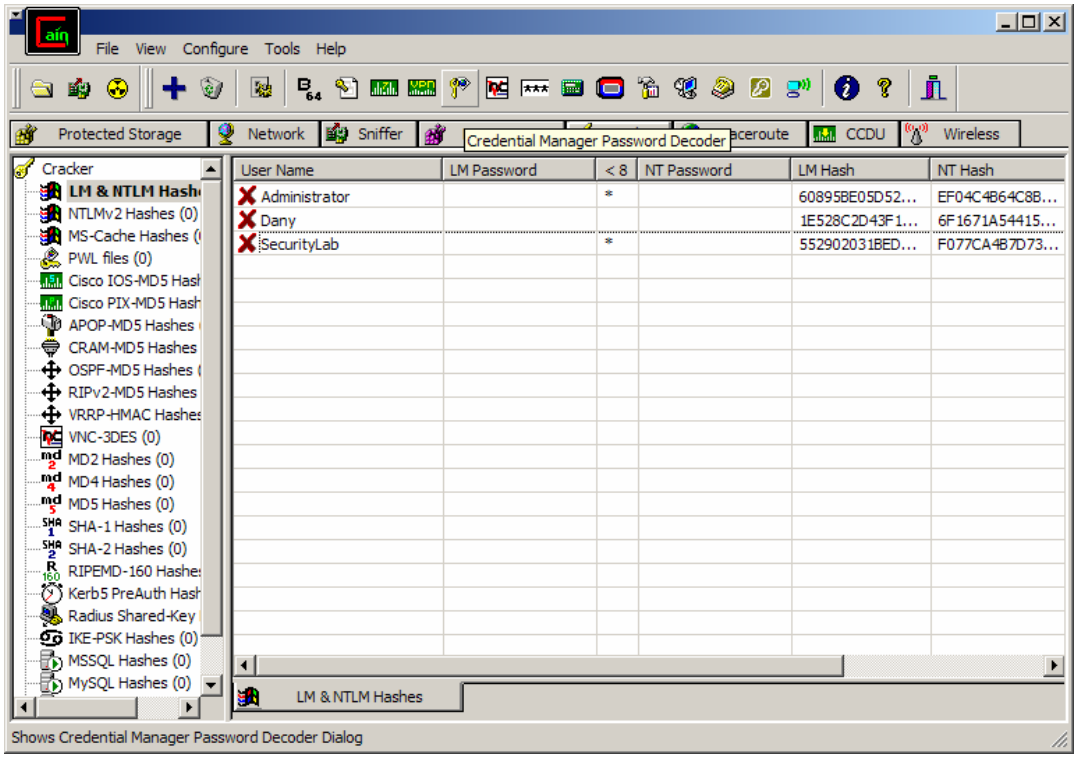

Abbildung 14 Cain Passwort Crack

Durch klick auf die rechte Maustaste auf dem Account, welcher entschlüsselt werden soll, öffnet sich das Kontextmenu mit den verschiedenen Angriffsmöglichkeiten. Da für diesen Test ein Account SecurityLab erstellt wurde, werden wir auf diesem eine Dictionary Attacke durchführen:

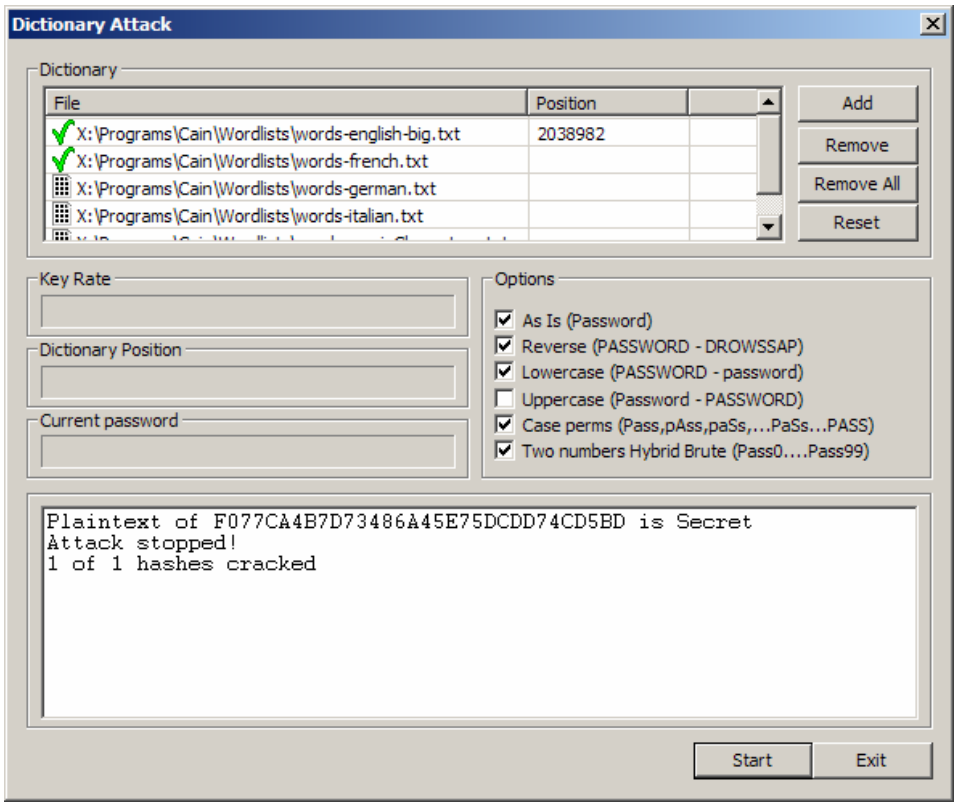

Abbildung 15 Cain Dictionary Angriff

Wie wir aus dem oberen Bild sehen, war die Attacke nach einiger Zeit erfolgreich und brachte folgendes Ergebnis zum Vorschein:

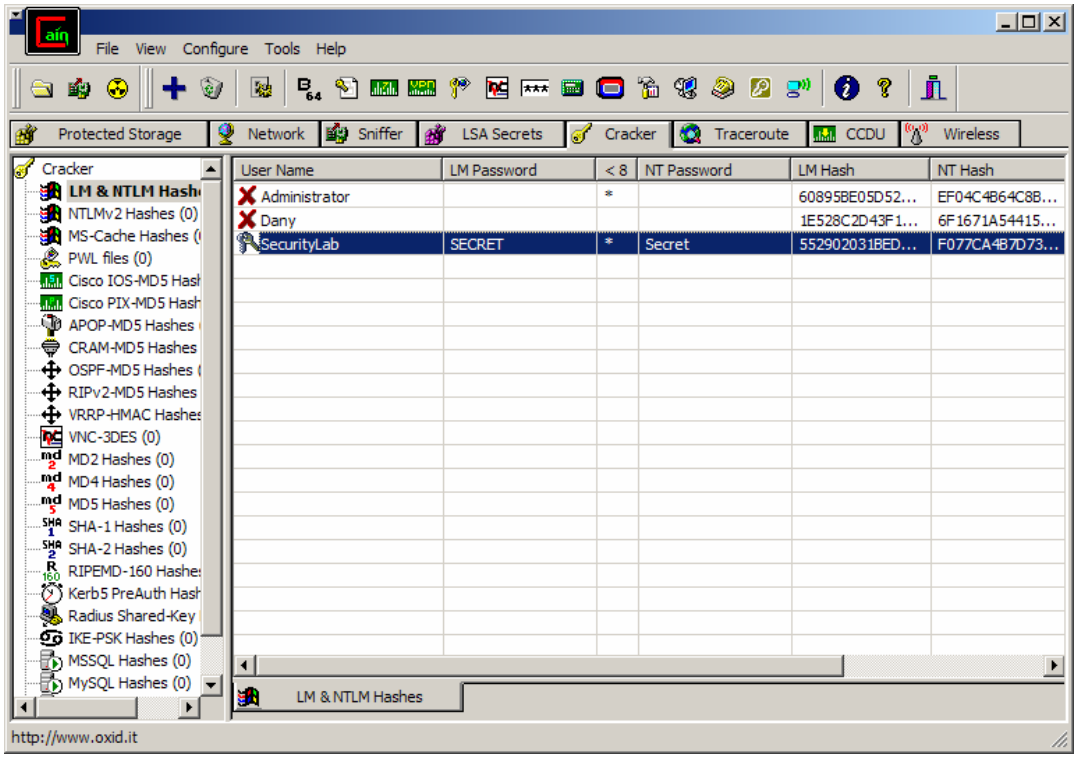

Abbildung 16 Cain Hash Angriff

Das Passwort lautete tatsächlich Secret und wurde in relativ kurzer Zeit entschlüsselt. Cain ist nach unserer Erfahrung nicht die schnellste Entschlüsselungsvariante, ist aber frei verfügbar und deshalb das Tool welches wir auf unserer Boot CD einsetzen.

Mit LC4 von @Stake dauerte die Dictionary Attacke weniger lang. Diese Software ist aber Kommerziell und deshalb nicht für unsere Boot CD geeignet, ist aber im Internet als Plugin erhältlich. Der Hersteller @stake wurde kürzlich von Symantec übernommen, deshalb ist die Zukunft des Produktes im Moment unklar. Ein weiteres Programm welches die SAM mittels SYSKEY entschlüsseln kann ist SAMInside von InsidePro (siehe [InsidePro]). Auch dieses ist kommerziell und deshalb nicht auf unserer Boot CD zu finden.

Die Geschwindigkeit der Werkzeuge ist stark unterschiedlich und hängt massgeblich von der Effizienz der Hash Algorithmen ab.

6. Technische Dokumentation KryptoNieTe

Wie bereits im Praxiseinsatz beschrieben ist das Programm fähig Registry Hives einzulesen und zu verarbeiten. Zu diesem Zweck haben wir einen minimalen Registry-Reader geschrieben, der es erlaubt die Grundlegenden Datenstrukturen und Werte auf einfache Weise auszulesen. Dies erlaubt es einerseits die SAM Datenbank auszulesen und andererseits den System-Hive und damit den Syskey zu lesen.

Die Anwendung ist nach dem Model View Control (MVC) Konzept aufgeteilt, was auch die Paketstuktur wiedergibt:

Tabelle 7 KryptoNieTe Paketstruktur

Paket	Inhalt/Beschreibung
kryptoniete	Enthält nur eine einzige Klasse: KryptoNieTe. Dies ist die Start-Klasse.
kryptoniete.control	Enthält den zentralen Kontroller (Klasse Controller). Der Kontroller ist für die Datenverwaltung und den Datenfluss innerhalb der Anwendung zuständig.
kryptoniete.model	Dieses Paket enthält die Datenstrukturen sowie auch den Hive Reader und die Klasse für die SAM Entschlsselung. Ausserdem befindet sich dort eine Klasse für die RC4 Ver- und Entschlüsselung. RC4 ist der einzige benötigte Algorithmus, der nicht von der Java Runtime zur Verfügung gestellt wird. Des weiteren befindet sich dort eine Klasse für häufig gebrauchte Konvertierungsfunktionen wie diejenige um Bytes in Hexadezimale Werte umzuwandeln und umgekehrt.
kryptoniete.view	Hier befinden sich die GUI Klassen. Das GUI wurde bewusst möglichst klein gehalten.

Das Programm verwendet Log4J um Fehlermeldungen und Statusmeldungen auszugeben. Dies erlaubt es sowohl die Tiefe des Loggings (Log Level) als auch die Art (Konsole, Datei usw.) anzupassen ohne die Anwendung umschreiben zu müssen. Im Debug-Modus werden sehr viele Werte (Zwischenergebnisse, ausgelesene Werte, Statusmeldungen) ausgegeben, die für ein „Release-Build“ nicht notwendig sind und einfach ausgeschaltet werden können.

Im Folgenden werden einige Probleme, die während der Programmierung auftraten kurz aufgeführt.

6.1. Controller

Hier gab es erwartungsgemäss wenige Probleme. Der Controller arbeitet ausschliesslich mit Standarddatentypen aus Java. Lediglich der Austausch der Syskey Informationen in verschiedenen Formaten (aus einer Datei, als Hex-String oder vom System-Hive) war vom Design her nicht ganz trivial.

Ein kleineres Problem hat sich auf Windows PE ergeben. Dort hat das Windows Look-and-Feel nicht funktioniert (Null pointer Exception). Das Standard (Swing) Look-and-Feel hat aber keine Probleme verursacht.

6.2. View

Das GUI wurde wie schon erwähnt möglichst schlicht gehalten. Alle Aktionen spielen sich auf einer einzigen Maske ab. Nur ein Fehlermeldungs-Dialog wurde zusätzlich erstellt. Dieser ist generisch gehalten und kann beliebige Nachrichten anzeigen.

In der Mitte des GUIs befindet sich ein kleines Log-Fenster in dem die wichtigsten Aktionen protokolliert werden.

Die komplexeste Aufgabe war hier ebenfalls die drei Möglichkeiten der Syskey Auswahl.

6.3. Model

Hier gab es diverse Probleme zu lösen. Hier eine Auswahl:

Datentypen:

Java unterstützt keine unsigned Datentypen. Werden also binäre Werte in Datentypen wie int, short oder long eingelesen, so interpretiert Java diese immer als signed. Dessen muss man sich bei jeder Operation mit den Datentypen im Klaren sein. Durch die Maskierung bei Casts kann der unsigned Charakter beibehalten werden.

```
Long longValue = (long)intValue && 0xFFFFFFFF;
```

Byte Order:

Java arbeitet immer im Big Endian (siehe [ENDIAN]) Modus. Die Hive Dateien sind aber immer im Little Endian Format abgespeichert. Somit muss beim Auslesen von Datentypen, die länger als 1 Byte sind die Byte Reihenfolge umgekehrt werden. Glücklicherweise gibt es eine Klasse genannt ByteBuffer. Diese erlaubt es die Byte Anordnung festzulegen. Beim Lesen von werden die Bytes dann automatisch richtig sortiert.

```
ByteBuffer length = ByteBuffer.allocate(4);
length.order(ByteOrder.LITTLE_ENDIAN);
file.read(length.array());
length.rewind();
int dataLength = length.getInt();
```

Datentypen-Konvertierung:

Java bietet zwar Möglichkeiten um beispielsweise Strings als Hexadezimale Werte zu interpretieren und in entsprechenden Datentypen abzulegen. Diese geht aber nach eigenen Erfahrungen nicht mit beliebig langen Hex-Strings. Das Problem mussten wir jeweils durch eigene Konvertierungs-Algorithmen lösen.

Insbesondere das Mapping der Datentypen auf die binäre Struktur der Registry Hives erzeugt mit Java signifikant mehr Aufwand als mit C. Das Ergebnis zeigt aber auch deutlich, dass es durchaus machbar ist.

7. Fazit

Wie man aus dem Laborbericht entnehmen kann, ist die Passwort Sicherheit bei Microsoft Windows Betriebssystemen nicht über jeden Zweifel erhaben. Das Problem liegt daran, dass der Schlüssel, mit welchem die SAM verschlüsselt wird, auf dem lokalen System liegt. Der Algorithmus für die Entschlüsselung der SAM ist nicht veröffentlicht. Die Verschlüsselung der SAM durch den SYSKEY ist wie Eingangs erwähnt "Security through Obscurity". Die einzige Möglichkeit den Algorithmus herauszufinden, war die genaue Analyse der beiden C/C++-Programme Bkhive ([Bkhive]) und und samdump2 ([samdump2]) und die genaue Dokumentation über den SAM und SYSTEM Hive. Anhand dieser beiden Quellen, ist es gelungen das Rätsel zu lüften und ein eigenes Programm für die SAM/SYSKEY Entschlüsselung in Java zu schreiben.

Für die Entschlüsselung der NTLM Hashes gibt es viele frei verfügbare Tools. In einer weiteren Arbeit, wäre es denkbar, eine sehr effiziente Cluster Lösung über RMI zu erstellen. Dies war aber leider wegen des engen Zeitrahmens nicht unmöglich.

Aufgrund des Know-Hows, welches wir im Verlauf der Laborarbeit aufgebaut haben, ist uns bewusst geworden, dass es nur eine sichere Art für den Schutz von NT-Passwörtern gibt. Der SYSKEY darf nicht lokal gespeichert sein. Ist der SYSKEY beispielsweise auf einer Diskette abgelegt, wird ein erfolgreicher Angriff sehr unwahrscheinlich. Leider ist die Speicherung auf Diskette (wer benutzt heutzutage schon noch Disketten?) die einzige von Windows unterstütze Export-Möglichkeit. Memory Stick oder andere Speichermedien werden nicht unterstützt. Die Passworteingabe beim booten ist insbesondere bei Computern, die von mehreren Benutzern verwendet werden unpraktikabel, da alle das Passwort kennen müssen. Einen zusätzlichen, wenn auch nur geringen Schutz liefert ein Passwortgeschütztes Bios, in welchem ein Booten von anderen Laufwerken untersagt wird. In diesem Falle muss der Angreifer erst diese Hürde nehmen, bevor er ein Betriebssystem von einem anderen Laufwerk starten kann.

Grundsätzlich ist über die NT-Passwort Sicherheit zu sagen, dass es Möglichkeiten gibt, unerlaubten Zugriff zu verhindern, diese aber für den User sehr umständlich und unkomfortabel sind. Es ist für jedes System abzuschätzen, ob nun Komfort oder Sicherheit wichtiger ist. Ausserdem nützt auch die Speicherung auf einer Diskette wenig, wenn diese dann permanent für jeden zugänglich im Laufwerk verbleibt.

8. Quellen und Links

Quelle	Beschreibung
[samdump2]	Samdump2, C++ Programm um die SAM Datei auszulesen und im pwdump Format auszugeben. Der Source Code ist offen aber die offizielle Homepage (http://studenti.unina.it/~ncuomo/syskey/) war leider nicht mehr verfügbar.
[Bkhive]	Bkhive, C++ Programm um den Syskey aus dem SYSTEM Hive auszulesen. Der Source Code ist offen aber die offizielle Homepage (http://studenti.unina.it/~ncuomo/syskey/) war leider nicht mehr verfügbar.
[BartPE]	Bart's PE Builder: http://nu2.nu/pebuilder/
[Sherpya]	Sherpya's PE/XPE Plugin Homepage: http://winpe.sourceforge.net/
[InsidePro]	IndidePro, Hersteller von SAMInside, einem Programm um Windows Passwörter zu knacken, http://www.insidepro.com/
[SAM]	Security Accounts Manager: http://www.beginningtoseethelight.org/ntsecurity/ Das Dokument bescchreibt die Struktur der Registry Hives ausführlich.
[NTMAG]	Windows NT Magazin, inside the registry, http://www.microsoft.com/technet/archive/winntas/tips/winntmag/inreg.mspx
[ENDIAN]	Wikipedia, Byte Reihenfolge, http://de.wikipedia.org/wiki/Byte-Reihenfolge Wikipedia, Endianness, http://en.wikipedia.org/wiki/Endianness
[MD5]	Beschreibung der MD5 Hashfunktion http://de.wikipedia.org/wiki/MD5
[RC4]	Beschreibung der Stromchiffre RC4 http://de.wikipedia.org/wiki/RC4
[DES]	Beschreibung der Blockchiffre DES http://de.wikipedia.org/wiki/Data_Encryption_Standard

9. Abbildungsverzeichnis

10. Tabellenverzeichnis