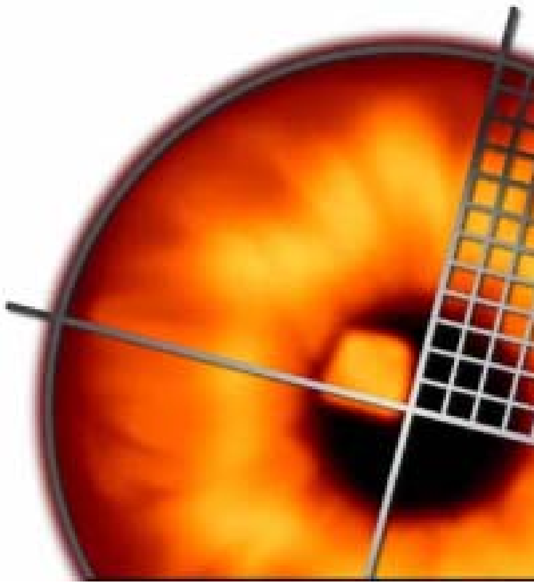
eEye® Digital Security



Inside Share

Shareの解析と可視化システム「Sharebot」の仕組み

Yuji Ukai

Senior Research Engineer
Senior Software Engineer

eEye® Digital Security

http://www.eeye.com

はじめに

Shareネットワークを介して拡散する暴露ウイルス

➡ 深刻な情報漏えいが相次ぐ

⬇

- アップロードした本人も削除できない
- ファイルを保持しているユーザーを特定する仕組みも提供されいない
- ファイル発信者情報を隠すための様々な工夫がなされている

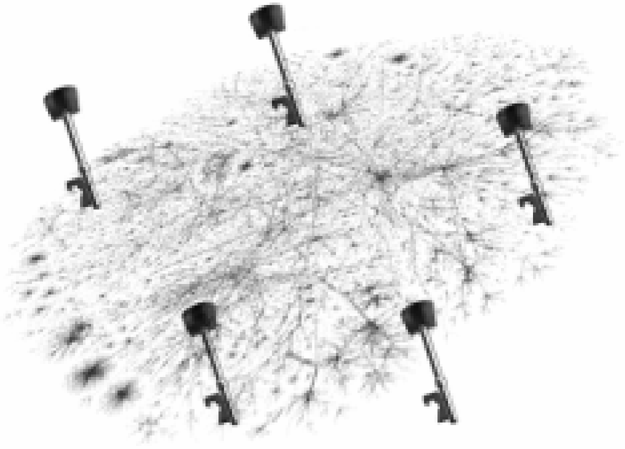
⬇

一旦情報漏えいが発生すると、現状では手の施しようが無い

目的

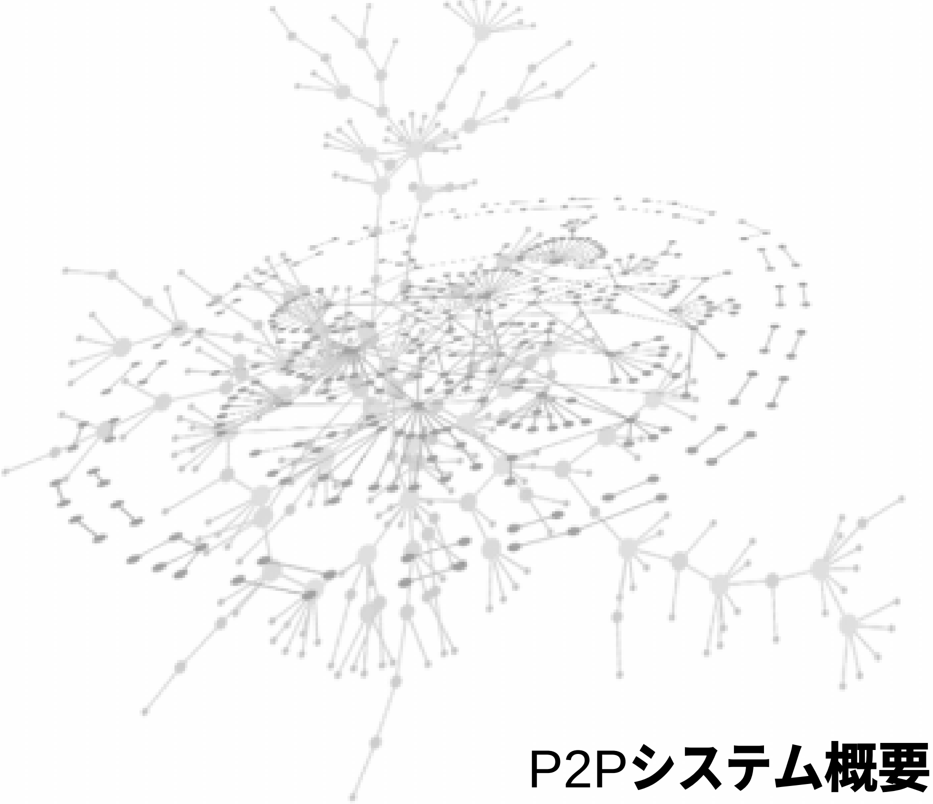
この状態に対処

- ファイルを保持しているノードのIPアドレスを知る事が第一歩
- Shareネットワークをクロールし、ネットワーク内に発生しているキー情報を蓄積。ファイル保持ノードを特定するシステムを開発
- 情報漏えいの事後処理、著作権侵害への対処など



概要

Overview



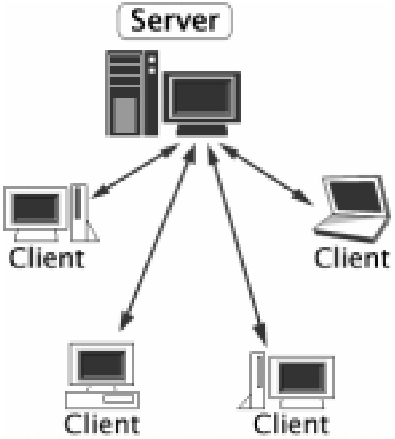
P2Pシステム概要

P2Pファイル交換・共有システム概要

Shareネットワークを可視化するには

Shareネットワーククローラの概要

Shareの基礎 - P2Pシステム

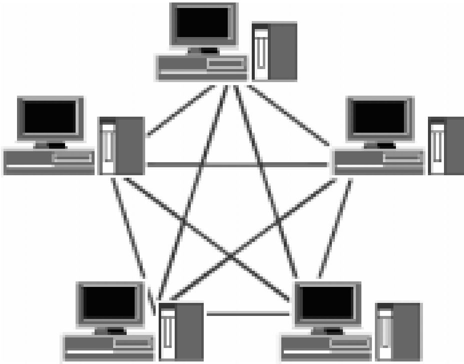


- P2P(peer-to-peer)システム

  定まったクライアント、サーバを持たず、ネットワーク上の他のノードに対し、クライアントとしてもサーバとしても働くようなノードの集合で形成されるシステム。

  - 管理コスト削減
    → サーバやネットワークのコストを削減できる
  - 資源分散
    → ネットワーク負荷、サーバリソースの増大を抑制できる
  - 耐故障性の実現
    → ノードの一部が停止してもサービスを継続できる

  - データの短時間同期が難しい
  - 構成によっては、制御ができない

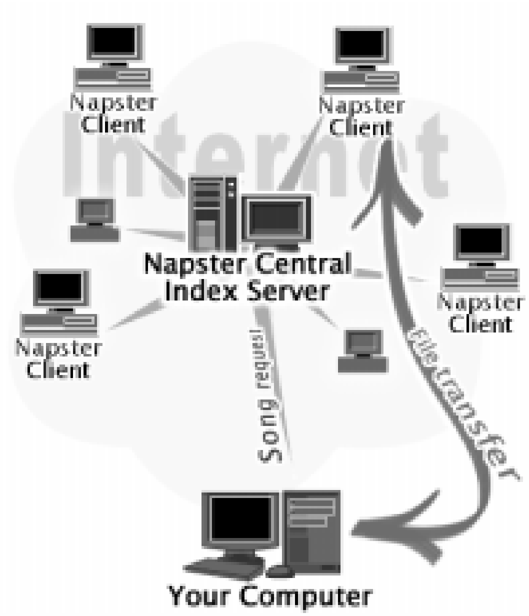


Shareの基礎 - P2Pファイル交換･共有システム

P2Pモデルで構成されたファイル交換･共有システム

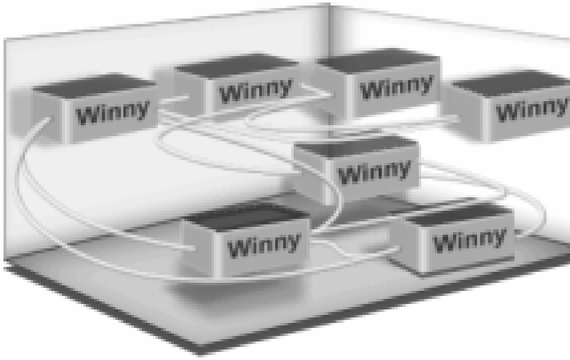
■ ハイブリッドP2P ファイル交換システム

ノード情報やファイルの所在は中央サーバが管理
Napsterなど



■ ピュアP2P ファイル交換システム

全ての処理をP2Pで行う
Winny、Gnutellaなど



Shareは、ピュアP2Pファイル交換システム

Shareネットワークを可視化するには

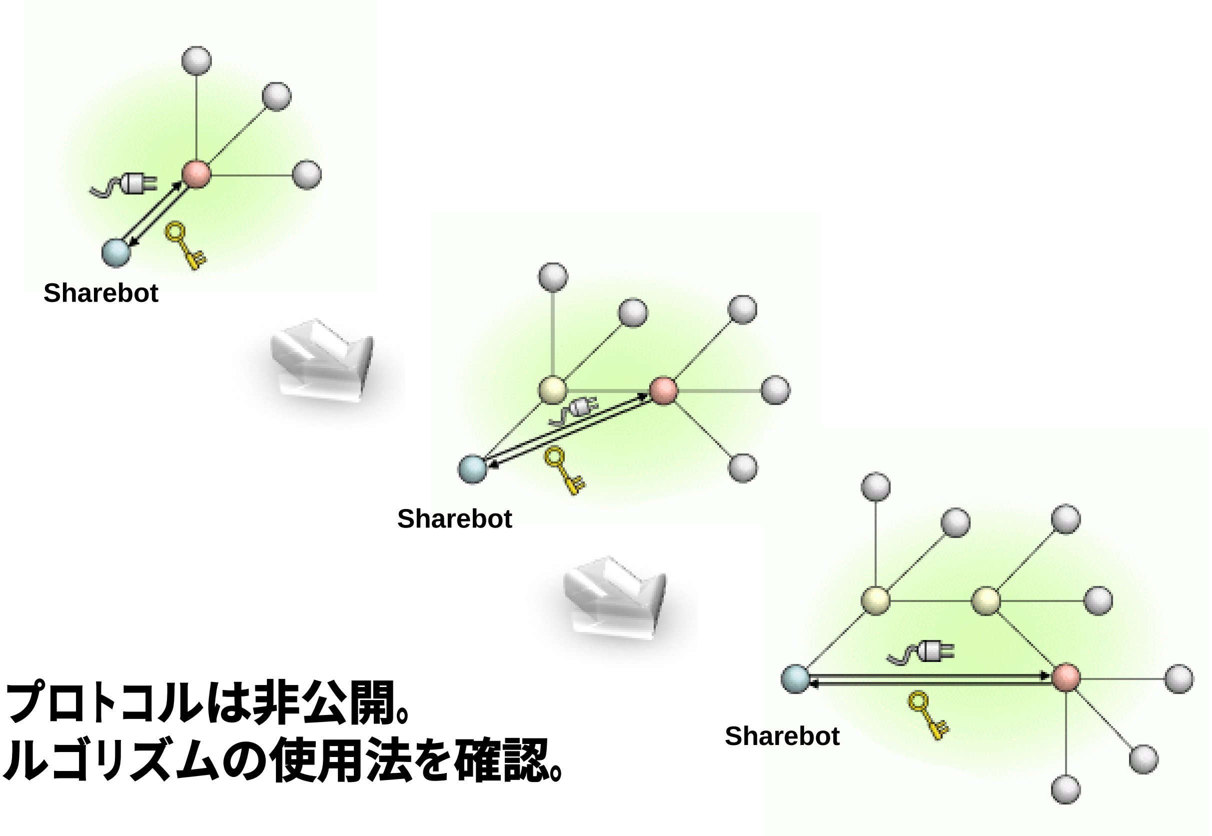
- ピュアP2Pであるため、不特定多数からの能動的な通信を受け付ける
  全ノードは他ノード情報を保持している

- ノードが保持している他ノード情報一覧を取得
  次々と接続を繰り返していく事で、ほぼ全てのノードをクロール可

- ファイルの所在を示すキー情報を同時に収集・蓄積
  指定されたファイルを所持しているノードの情報を列挙可能

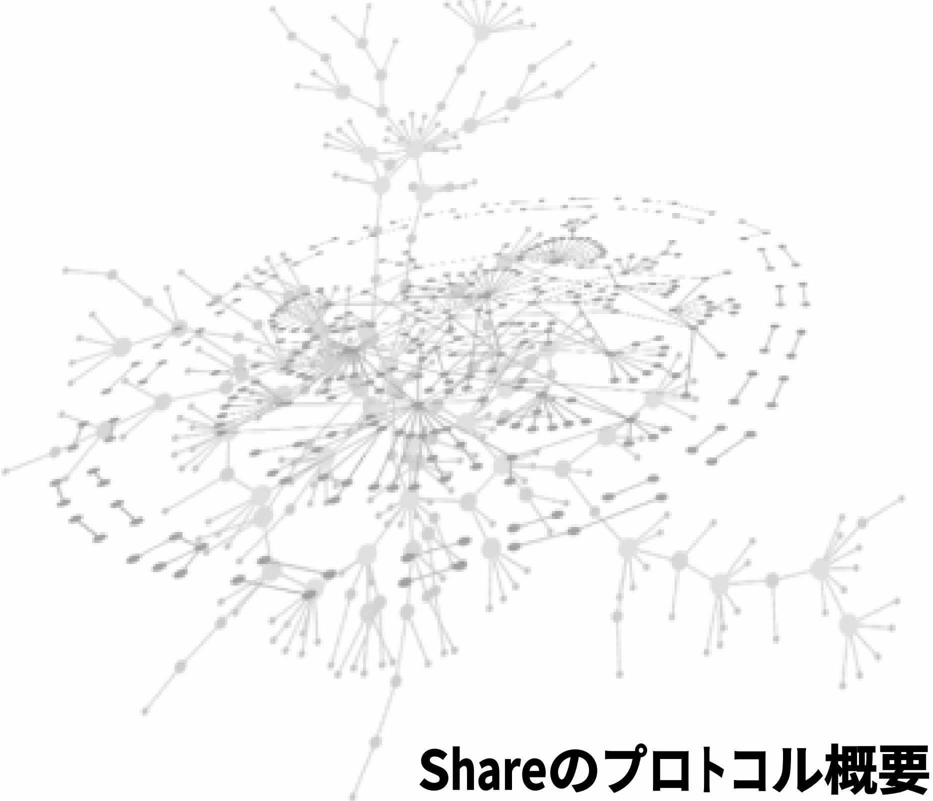
Shareネットワーククローラ

- Shareネットワーク内に存在するノードの一つとして振る舞う。
- 相手を次々と変えながら接続し、キーを収集。
- 相手のノードとは、Shareプロトコルに則って一対一で通信。



- Shareパケットは暗号化。プロトコルは非公開。
- プロトコルを解析。暗号アルゴリズムの使用法を確認。

解析 Analysis



Shareのプロトコル概要
暗号アルゴリズム
クローラー開発のアプローチ
プロトコル、コマンドの詳細

Shareプロトコルの概要

- Share EX2のプロトコルはTCPベース。
- ユーザーが設定する任意のポートで接続を聴取。
- ユーザーは、自らのIPアドレスとポート番号の対に特殊なエンコードを施した「ノード情報」をwebなどで公開、他者の接続を促す。
- ネットワークに存在しているノードが接続を受け付けると、接続元に他ノード情報を提供。ネットワークが拡大。
- パケットは暗号化されている。パケットをキャプチャなどで、簡単にプロトコルの概要を把握したり、やりとりしている情報を確認したりする事は不可。

Shareで利用されている暗号アルゴリズム概要

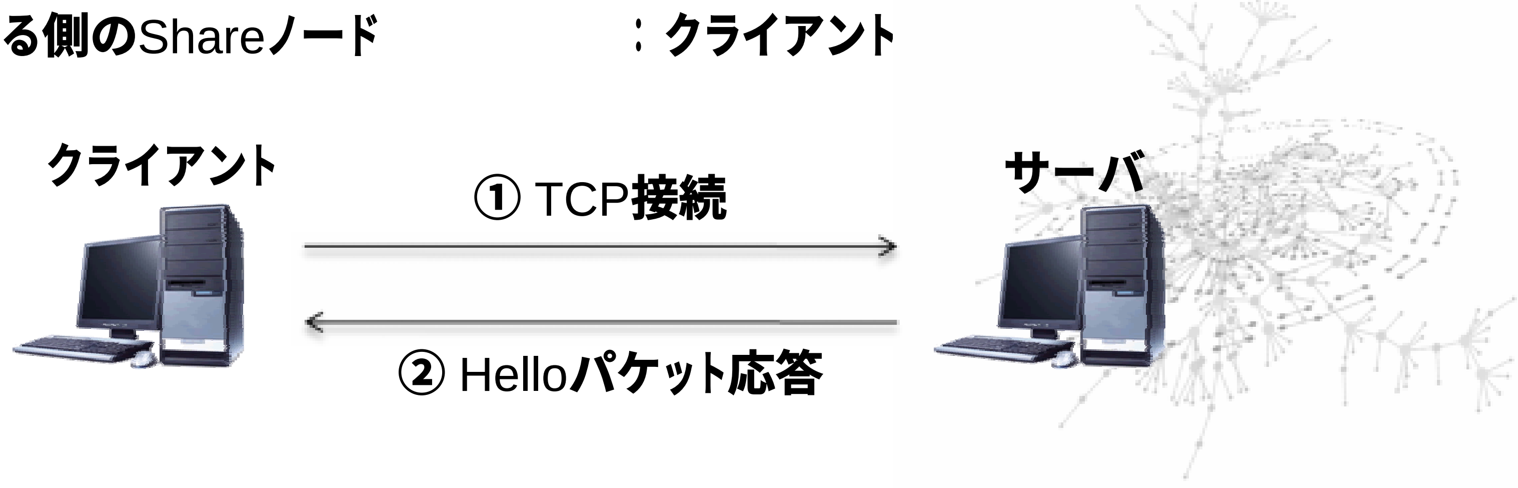
- 1024ビットRSAとRC6を併用
- ハッシュアルゴリズムはCRC-32とSHA-1を利用
- 初期パケット以外は公開鍵暗号がベースになっている
- トラフィックを傍受し、やりとりしているファイルの情報を把握する事は非常に困難。
- キーペアもシステム起動時に毎回ランダムに生成。
  本人同定のリスクも比較的少ない。
- 各暗号アルゴリズムは適切に利用・実装されている。

クローラ開発のアプローチ

1. 利用されている暗号アルゴリズムとその適用法を知る
   - 暗号化は、あくまで第三者による盗聴を防ぐためのもの。
   - 対象ノードと一対一通信ができるため、暗号解析は不要。
   - 利用法を知るだけで良い。
2. プロトコル解析
   - 暗号化前のパケット、復号後のパケットを取得して解析
   - コードを追跡
3. ノードに接続し、キー情報を取得するまでのプロトコルを実装
4. クローラー開発

第1ステージ - Helloパケット

【定義】
接続を受ける側のShareノード　：サーバ
接続する側のShareノード　：クライアント



クライアント

① TCP接続

② Helloパケット応答

サーバ

サーバーは、聴取しているTCPポートに接続があると、特定のパターンを持つパケットをクライアントに返信する

⇨ 「Helloパケット」とする

- Helloパケットを含め、全てのパケットは暗号化されている。
- Helloパケットの暗号キーはポート番号から一意に生成されたもの。

キー生成手順(1)

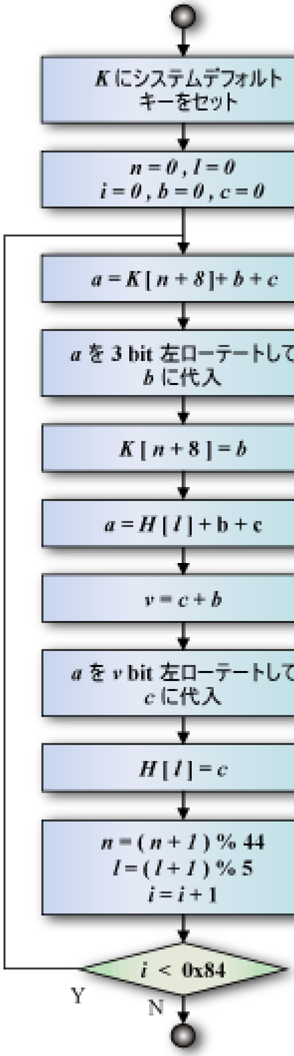
- 送信パケットは、最終的に必ずRC6で暗号化される。

- RC6暗号化ルーチンには、RC6キーと元データを入力。
ビットローテートやXOR演算を繰り返しながら元データを暗号化。

- RC6キーを保持するバッファは256バイト。
第1ステージでは:

  (1) 対象ノードのポート番号から20バイトのSHA-1ハッシュを生成
  (2) 得られた20バイトのSHA-1ハッシュを元にRC6キーを生成

キー生成手順(2)



- **ポート番号から**SHA-1**ハッシュを得る**

```
CryptCreateHash(hProv, CALG_SHA1, 0, 0, &hHash);
CryptHashData(hHash, (UCHAR *)&wPort, 2, 0);
memset(pHashData,0,sizeof(HashData));
dwLenHash=0x14;
CryptGetHashParam(hHash,HP_HASHVAL,pHashData,&dwLenHash,0);
```

- SHA-1**ハッシュから**RC6**キーを計算**

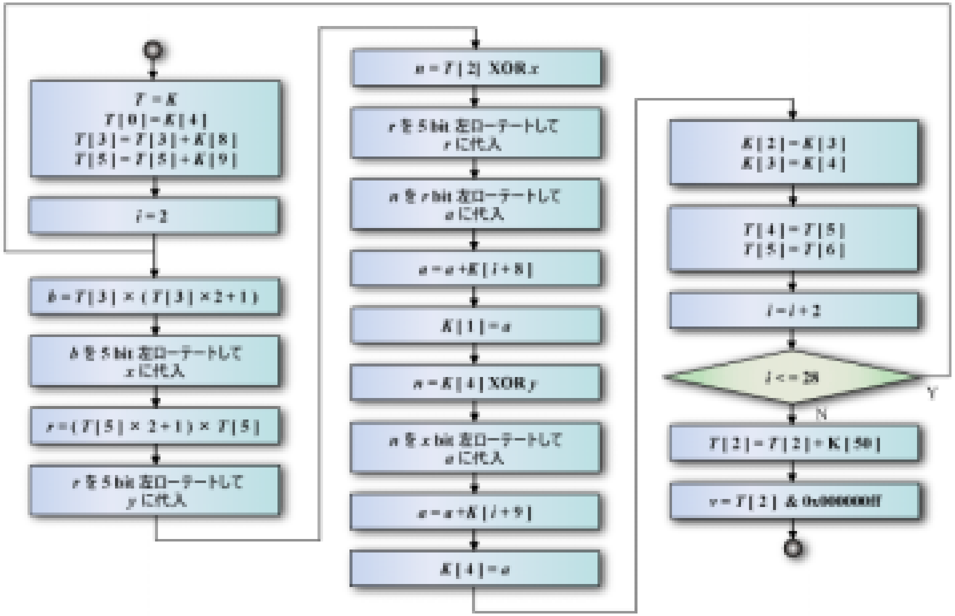
K : **キー**

H : 20**バイト**SHA-1**ハッシュ値**

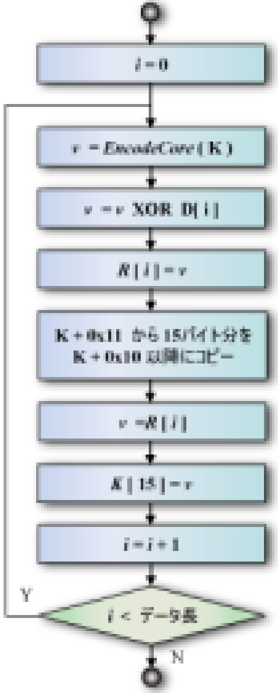
システムデフォルトキー :
先頭32**バイトは**0、32**バイト以降は**RC6**マジックテーブル**

暗号化と復号

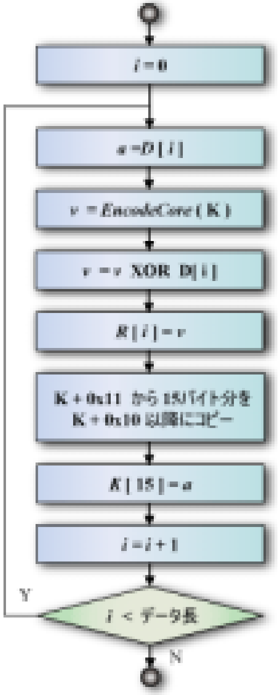
共通演算部



暗号化



復号



入力 : **キー** K (DWORD**型配列**)
出力 : **一意に算出される値**v (DWORD)**型**

D : **入力データ**
K : **キー**
R : **出力データ**

Helloパケットの復号

- Helloパケットは4バイト
- サーバから4バイト受信し、RC6復号関数に入力する

復号後のHelloパケット = 02 00 00 00 = 0x00000002

- textセクションでハードコードされた32ビット即値

- 対象の全てのTCPポートに接続し、送られた先頭4バイトのパケットを復号すれば、対象でShareが動作しているか否かを判定可能

- パケットモニタリングやゲートウエイでShareの通信を検知する事も可

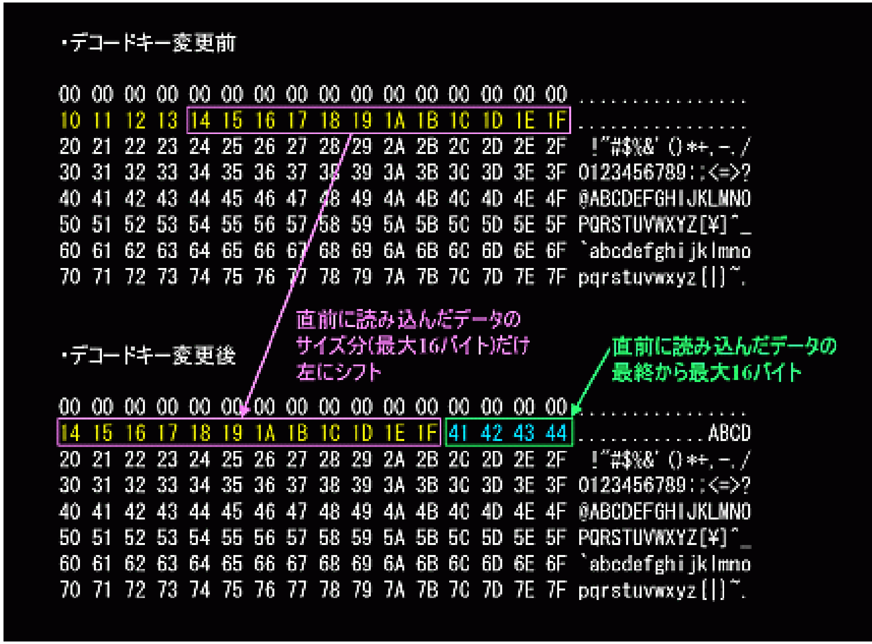
復号キーの変化

1. 受信したパケットの最後16バイト(復号前の生データ)をNewKeyにセット、n = 16 とする。
   16バイトに満たない場合は、読み込んだデータの全てをNewKeyにセットし、そのサイズを n とする。

   例 : n=4, NewKey = 0x41,0x42,0x43,0x44

2. 復号キーの 16 バイト目から
   16バイト分を左に nバイトシフト

3. NewKeyを コピー



・デコードキー変更前

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F  !"#$%&'()*+,-./
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0123456789:;<=>?
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F @ABCDEFGHIJKLMNO
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F PQRSTUVWXYZ[¥]^_
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F `abcdefghijklmno
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F pqrstuvwxyz{|}~.
```

直前に読み込んだデータの
サイズ分(最大16バイト)だけ
左にシフト

直前に読み込んだデータの
最終から最大16バイト

・デコードキー変更後

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 41 42 43 44 ............ABCD
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F  !"#$%&'()*+,-./
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0123456789:;<=>?
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F @ABCDEFGHIJKLMNO
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F PQRSTUVWXYZ[¥]^_
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F `abcdefghijklmno
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F pqrstuvwxyz{|}~.
```

サーバ公開鍵の受け取り

- サーバは、Helloパケットに次いで自身の公開鍵とそのデータ長を送信

- データ長は0x0084バイト固定。
  サーバは、Helloパケット直後にWORD値として送信。
  クライアントは、Helloパケットと同様の手順で復号 (0x84, 0x00)

- サーバは、0x84バイトのサーバー公開鍵(1024ビットRSA)を送信。
  ただし鍵ヘッダがない。

適切な鍵ヘッダ = 06 02 00 00 00 a4 00 00 52 53 41 31 00 04 00 00

06	PUBLICKEYSTRUC構造体のbTypeパラメータ。0x06は、公開鍵ブロブである事を示す
02	PUBLICKEYSTRUC構造体のbVersionパラメータ。ブロブフォーマットのバージョン
00 00	PUBLICKEYSTRUC構造体の予約フィールド(2バイト)
00 a4 00 00	PUBLICKEYSTRUC構造体のaiKeyAlgパラメータ。CALG_RSA_KEYX(RSA公開鍵)を示す
52 53 41 31	RSAPUBKEY構造体のmagicパラメータ。RSA1
00 04	RSAPUBKEY構造体のbitlen パラメータ。0x0400 = 1024ビットである事を示す

Microsoft CSPに公開鍵をインポート

```
CryptImportKey(hProv, ServerRSAKey, 0x94, 0, 0, &hKey);
```

第2ステージ - クライアント暗号化キーの送信 (1)

- クライアントは自らの暗号化キーをサーバに送信

  - 今まで利用してきたポート番号ベースの暗号化キーではない
  - 以降のデータ交換で利用するための新しい第2ステージ用暗号化キー

- 送信される第2暗号化キー

  - RC6キーではなく、その元となっている20バイト長の暗号化キー
  - サーバー側では同様のRC6キーを作成できる
  - 20バイト長の第2暗号化キーはランダムに決定

- 第1ステージ
- 第2ステージ

第2ステージ - クライアント暗号化キーの送信 (2)

- **データ長0x0080を送信(WORD値)**

  - **データ長を暗号化して送信**
  データ長の暗号化に利用される暗号化キーは、
  サーバポート番号から生成されたものをそのまま利用
  (**暗号化キーの変化のないものを利用**)

- **第2暗号化キーを送信(0x80バイト)**

  - 20**バイトの第**2**暗号化キーを生成**(**ランダム**)
  - **サーバー公開鍵で第**2**暗号化キーを暗号化** - CryptEncrypt()
  暗号化されたデータのサイズは0x80**バイト。**
  - **サーバ公開鍵で暗号化された第**2**暗号化キーを、更に第**1**暗号化キーで暗号化**
  - **サーバに送信**

自ノード情報の送信

自ノード情報には･･･

- バージョン
- 最大送信速度
- 最大受信速度
- IPアドレス
- ポート番号
- 起動時刻、など

```
00 00 00 00 09 00 53 00 69 00 67 00 6e 00 61 00    ......S.i.g.n.
74 00 75 00 72 00 65 00 04 00 43 41 54 00 0a 00    t.u.r.e...CAT.
56 00 65 00 72 00 73 00 69 00 6f 00 6e 00 4e 00    V.e.r.s.i.o.n.
75 00 6d 00 04 00 63 00 00 00 0a 00 56 00 65 00    u.m...c.....V.
72 00 73 00 69 00 6f 00 6e 00 53 00 74 00 72 00    r.s.i.o.n.S.t.
08 00 45 58 32 00 00 00 00 00 08 00 56 00 65 00    ..EX2.......V.
72 00 49 00 6d 00 61 00 67 00 65 00 c4 00 63 00    r.I.m.a.g.e.ﾄ.
00 00 45 00 58 00 32 00 00 00 00 00 00 00 00 00    ..E.X.2.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ..............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ..............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ..............
00 00 5a 3e 5a 9a db bc 1b 30 75 6e 3c f1 40 ad    ..Z>Z.ﾛｼ.0un<.
80 d4 b1 65 31 4a d3 2f 6c bb bc c7 7f 70 d3 57    .ﾔｱe1Jﾓ/lｻｼﾇ.p
2c 20 c2 e8 e7 15 50 fa 19 25 a2 de 73 f9 89 5b    ,, ﾂ...P..%｢ﾞs.
74 b0 52 18 6e 94 91 ab bf fa 17 48 82 00 43 26    tｰR.n..ｫｿ..H..
86 b5 14 c4 85 32 06 98 9b db ab 49 83 66 41 da    .ｵ.ﾄ.2...ﾛｫI.f
42 6c fd 0b f2 95 3e e4 81 1d 31 ff c0 ac 94 a8    Bl....>...1.ﾀｬ.
0e 26 a1 fd bc 5f eb e6 f1 fd 91 8e 1e 21 90 a9    .&｡.ｼ_.......!.
fc c0 87 15 92 44 63 50 87 f3 2e eb 72 e0 97 c5    .ﾀ...DcP....r.
1f 38 05 00 53 00 70 00 65 00 65 00 64 00 04 00    .8..S.p.e.e.d.
00 f4 01 00 06 00 53 00 70 00 65 00 65 00 64 00    ......S.p.e.e.
32 00 04 00 00 f4 01 00 09 00 44 00 6f 00 77 00    2.........D.o.
6e 00 53 00 70 00 65 00 65 00 64 00 04 00 00 00    n.S.p.e.e.d...
00 00 07 00 55 00 70 00 53 00 70 00 65 00 65 00    ....U.p.S.p.e.
64 00 04 00 00 00 00 00 0a 00 4d 00 69 00 6e 00    d.........M.i.
56 00 65 00 72 00 73 00 69 00 6f 00 6e 00 04 00    V.e.r.s.i.o.n.
60 00 00 00 09 00 4c 00 6f 00 63 00 61 00 6c 00    `.....L.o.c.a.
41 00 64 00 64 00 72 00 06 00 00 00 00 00 5c 11    A.d.d.r.......
08 00 42 00 6f 00 6f 00 74 00 54 00 69 00 6d 00    ..B.o.o.t.T.i.
65 00 08 00 38 3d 1e 58 8d 31 c7 01 03 00 4c 00    e...8=.X.1ﾇ...
44 00 50 00 14 00 43 10 e3 dc e9 6b 30 7d 17 c6    D.P...C..ﾜ.k0}
d5 c5 0a 2c 95 ff da d1 df b3 08 00 49 00 6e 00    ﾕﾅ.,..ﾚﾑﾟｳ..I.
66 00 6f 00 46 00 6c 00 61 00 67 00 04 00 01 00    f.o.F.l.a.g...
00 00 07 00 44 00 62 00 67 00 49 00 6e 00 66 00    ....D.b.g.I.n.
6f 00 58 00 00 00 00 00 00 00 00 00 00 00 00 00    o.X...........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ..............
00 00 00 00 8c 9c f8 02 00 00 00 00 10 27 00 00    .............'
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ..............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ..............
2c 24 01 00 00 00 00 00 00 79 09 00 04 00 54 00    ,$.......y....
69 00 6d 00 65 00 08 00 a2 7e 9b 84 8d 31 c7 01    i.m.e...｢~...1
0a 00 52 00 65 00 6d 00 6f 00 74 00 65 00 41 00    ..R.e.m.o.t.e.
64 00 64 00 72 00 06 00 c0 a8 00 03 5c 11 07 00    d.d.r...ﾀｨ..¥..
43 00 6f 00 6d 00 6d 00 61 00 6e 00 64 00 04 00    C.o.m.m.a.n.d.
00 00 00 00 0c 00 50 00 6f 00 72 00 74 00 43 00    ......P.o.r.t.
68 00 65 00 63 00 6b 00 53 00 69 00 67 00 10 00    h.e.c.k.S.i.g.
0e fe 47 8b f7 7f 3f 22 3f 2a c3 d0 c4 b7 e1 9f    ..G...?"?*ﾃﾐﾄｷ.
09 00 50 00 6f 00 72 00 74 00 43 00 68 00 65 00    ..P.o.r.t.C.h.
63 00 6b 00 04 00 01 00 00 00                      c.k.......
```

(例) 最大送信速度

```
1f 38 05 00 53 00 70 00 65 00 65 00 64 00 04 00 .8..S.p.e.e.d...
00 f4 01 00 06 00 53 00 70 00 65 00 65 00 64 00 ......S.p.e.e.d.
32 00 04 00 00 f4 01 00 09 00 44 00 6f 00 77 00 2.........D.o.w.
6e 00 53 00 70 00 65 00 65 00 64 00 04 00 00 00 n.S.p.e.e.d.....
```

0x01f4 = 500

自ノード情報に含まれるコードハッシュ

- 自ノード情報の先頭4バイトは、自身のShareアプリケーションのコードイメージハッシュ値(CRC32)

- 他ノードから送られた自ノード情報に含まれるコードハッシュ値が、自身のコードイメージハッシュ値と異なる場合、接続を切る

- クラック版のShareアプリケーションを既存のShareネットワークに参加できなくさせるのが目的？

その他のクラック、解析対策

- Shareは定期的に自身のコードイメージハッシュをチェックしている
- イメージ改造やint3ブレイクポインタ設置はチェックルーチンに引っかかる
  → アプリケーションの強制終了

クラスタ情報の送信

フィールド名	バイト数	値
サイズ	1バイト	クラスタデータの合計バイト数(nバイト) / 2の値
クラスタデータ	nバイト	クラスタ名(ワイドキャラクタ)。複数指定する場合は、NULLの後に次のクラスタ名を指定
ターミネータ	4バイト	61 00 00 00

(例)

```
0x20,

0x6D,0x00,0x6F,0x00,0x63,0x00,0x68,0x00,0x69,0x00,0x79,0x00,0x61,0x00,0x00,0x00,
0x75,0x00,0x68,0x00,0x6F,0x00,0x68,0x00,0x6F,0x00,0x68,0x00,0x6F,0x00,0x00,0x00,
0x75,0x00,0x6B,0x00,0x65,0x00,0x6B,0x00,0x65,0x00,0x6B,0x00,0x65,0x00,0x00,0x00,
0x63,0x00,0x6F,0x00,0x6D,0x00,0x6D,0x00,0x6F,0x00,0x6E,0x00,0x00,0x00,0x00,0x00,

0x61,0x00,0x00,0x00
```

クラスタ名 = mochiya, uhohoho, ukekeke, common

開始要求

- **サーバを、各種情報を送信するモードに移行させる (開始要求)**

  0xff, 0x05, 0x06

- **サーバは、開始要求を受信後、以下の情報を順次送信する**

(1) **サーバ自ノード情報**
(2) **サーバクラスタ情報**
(3) **コマンド送信開始**
(4) **キー情報やクエリなどの各種コマンド**

サーバー自ノード情報とクラスタ情報

- **サーバー自ノード情報の受信前に、**WORD**値のサイズ情報を受信**
- **サーバー自ノード情報は、クライアント自ノード情報同様、**
  最大送信速度、最大受信速度、起動時刻等などが含まれる
- **つぎに、クラスタ情報が送られて来るまで待機**
  1**バイトづつ読み込んで復号、**0x01 **を受け取るまで繰り返す**
- **先頭**2**バイトが**WORD**値のサイズ情報**
  サーバクラスタデータのフォーマットは、クライアントと同様

コマンド

- **コマンド送信開始が来るまで待機**

  - 1**バイトづつ読み込んで復号、**0x01, 0x05, 0x06 **が来るまで待機**

  - **以降コマンド処理ループに入る**

- **コマンド番号とペイロード**

  - 1**バイト受信して復号**(**コマンド番号**)
    キー拡散、ノーマルクエリ、ハッシュクエリといった合計9**個のコマンド**

  - **後に続くペイロードのサイズは、コマンド番号やペイロード内のサイズフィールド値によって変化**

コマンドとペイロードサイズ

1. **コマンド番号 0**

8バイト : サイズフィールド
nバイト : データフィールド

- n = サイズフィールドの先頭2バイト目から合計2バイトのWORD値

2. **コマンド番号 2、3、4**

9バイト : サイズフィールド
r*6バイト : ノード情報フィールド。
nバイト : データフィールド

- r = ノードの個数。サイズフィールドの先頭から合計2バイトのWORD値。
- 各ノード情報フィールドの先頭4バイトはIPアドレス、続く2バイトはポート番号。
- n = サイズフィールドの先頭5バイト目から合計2バイトのWORD値。

3. **コマンド番号 5、6、7、8**

ペイロード無し

キー拡散コマンド

キー拡散コマンド(コマンド番号0)
ノード情報やファイル名を含むキーを処理

コマンド番号 0

8**バイト**　: **サイズフィールド**
n**バイト**　: **データフィールド**

n = **サイズフィールドの先頭**2**バイト目から合計**2**バイトの**WORD**値**

- **データフィールドは圧縮されている**
- **データフィールドの先頭から**4**バイト目以降を展開**

キー情報の展開

Shareプロセスのアドレス0x004C6618に展開ルーチンが存在

- 出力バッファのサイズをスタックにPUSH
- ecx = 出力バッファのアドレス
- eax = 入力データのアドレス
- edx = 入力データのサイズ

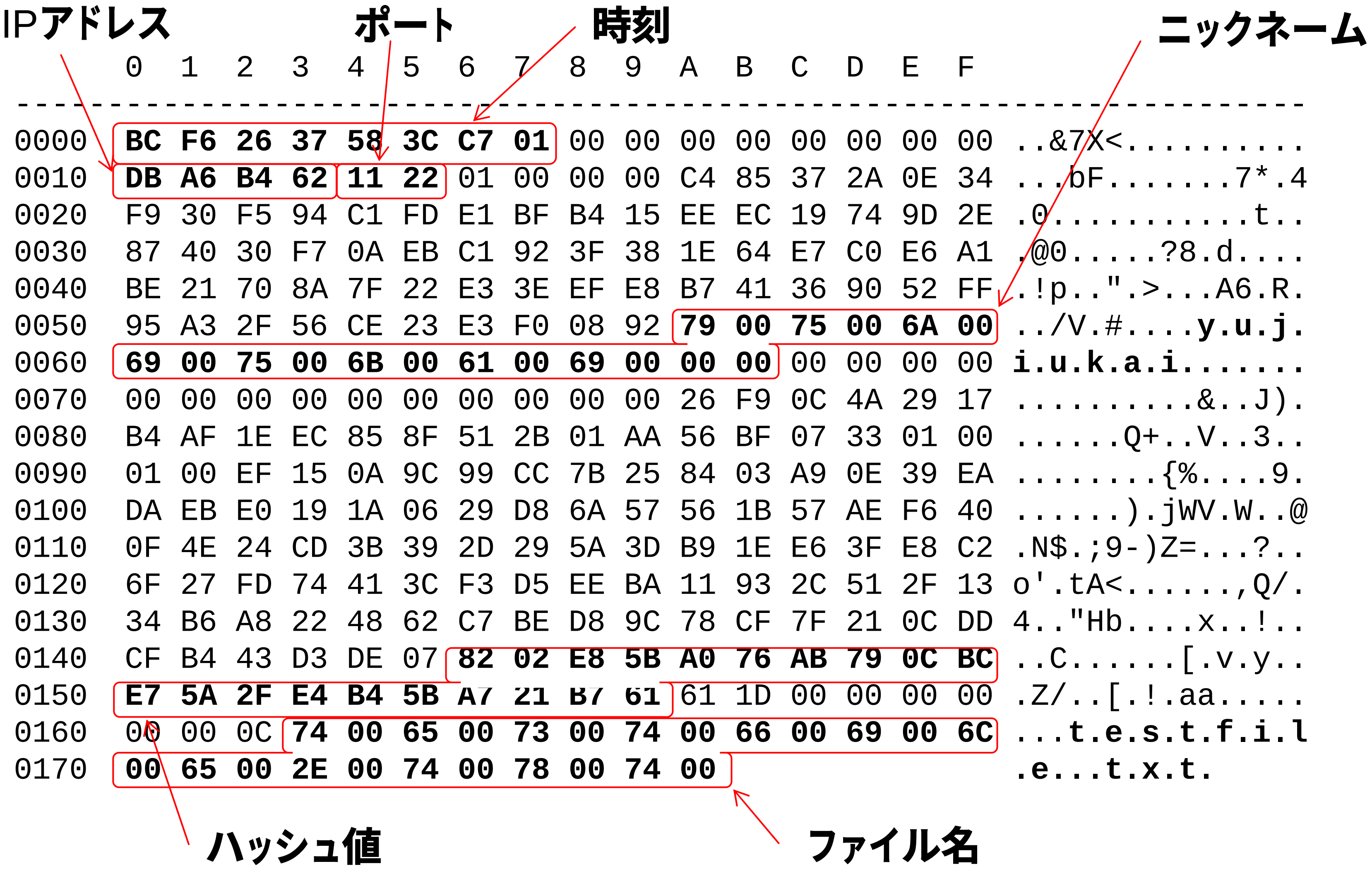
展開ルーチンを呼ぶと、ecxが示すバッファにキー情報が展開される

```
__asm{
        mov     eax,dword ptr [dwLenOutbuf]
        push    eax
        mov     ecx,dword ptr [pOutbuf]
        mov     edx,dword ptr [dwLenInputBuf]
        lea     eax,dword ptr [testdata]
        mov     edi,DecodeKeyInfo
        call    edi
        mov     dword ptr [dwErr],eax
}
```

Shareイメージのロード

- ShareイメージをロードするメモリエリアをVirtualAlloc
- ReBaseImage() でベースアドレス変更
- PEヘッダをパースして必要な情報を収集
- セクションを適切に構成しつつロード
- インポートテーブルを構成
- キー情報展開APIなどのエントリアドレスを算出

展開されたキー情報



IPアドレス

ポート

時刻

ニックネーム

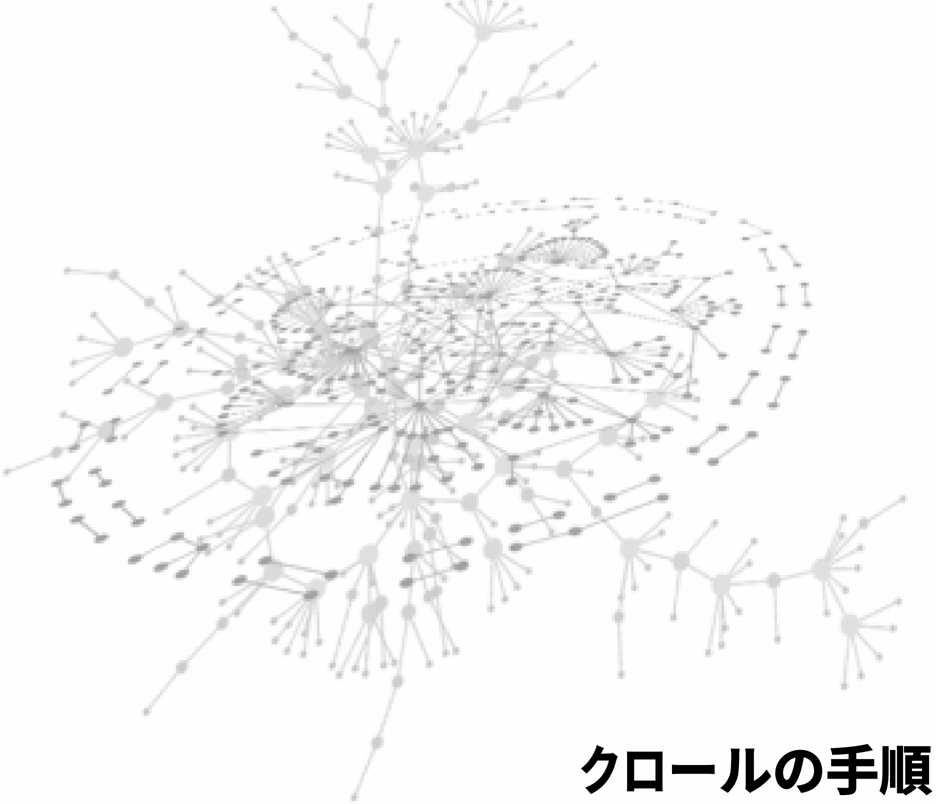
```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
-----------------------------------------------------------
0000  BC F6 26 37 58 3C C7 01 00 00 00 00 00 00 00 00 ..&7X<..........
0010  DB A6 B4 62 11 22 01 00 00 00 C4 85 37 2A 0E 34 ...bF.......7*.4
0020  F9 30 F5 94 C1 FD E1 BF B4 15 EE EC 19 74 9D 2E .0...........t..
0030  87 40 30 F7 0A EB C1 92 3F 38 1E 64 E7 C0 E6 A1 .@0.....?8.d....
0040  BE 21 70 8A 7F 22 E3 3E EF E8 B7 41 36 90 52 FF .!p..".>...A6.R.
0050  95 A3 2F 56 CE 23 E3 F0 08 92 79 00 75 00 6A 00 ../V.#....y.u.j.
0060  69 00 75 00 6B 00 61 00 69 00 00 00 00 00 00 00 i.u.k.a.i.......
0070  00 00 00 00 00 00 00 00 00 00 26 F9 0C 4A 29 17 ..........&..J).
0080  B4 AF 1E EC 85 8F 51 2B 01 AA 56 BF 07 33 01 00 ......Q+..V..3..
0090  01 00 EF 15 0A 9C 99 CC 7B 25 84 03 A9 0E 39 EA ........{%....9.
0100  DA EB E0 19 1A 06 29 D8 6A 57 56 1B 57 AE F6 40 ......).jWV.W..@
0110  0F 4E 24 CD 3B 39 2D 29 5A 3D B9 1E E6 3F E8 C2 .N$.;9-)Z=...?..
0120  6F 27 FD 74 41 3C F3 D5 EE BA 11 93 2C 51 2F 13 o'.tA<......,Q/.
0130  34 B6 A8 22 48 62 C7 BE D8 9C 78 CF 7F 21 0C DD 4.."Hb....x..!..
0140  CF B4 43 D3 DE 07 82 02 E8 5B A0 76 AB 79 0C BC ..C......[.v.y..
0150  E7 5A 2F E4 B4 5B A7 21 B7 61 61 1D 00 00 00 00 .Z/..[.!.aa.....
0160  00 00 0C 74 00 65 00 73 00 74 00 66 00 69 00 6C ...t.e.s.t.f.i.l
0170  00 65 00 2E 00 74 00 78 00 74 00                .e...t.x.t.
```

ハッシュ値

ファイル名

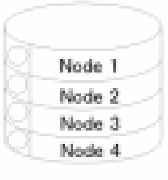
実装

Implementation

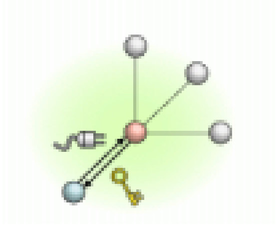


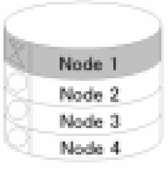
クロールの手順
Retina Sharebotの特徴
ハードウエア環境
セットアップ法
利用法

クロールの手順

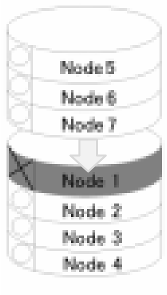


1. - **ノードDBに初期ノードをセット**
   - **全て未処理フラグをセット**





2. - **ノードDBの先頭から未処理フラグがセットされたノードを検索**
   - **未処理フラグがセットされたノードに接続し、キー情報を得る**
   - **対象ノードは処理中のフラグをセット**



3. - **得られたキー情報から他ノード情報を取り出す。**
   - **ノードDBの先頭に得られた新ノードを格納(重複チェック)**
   - **対処ノードは処理済みのフラグをセット**
   - **新たなノードには未処理フラグをセット**

Retina Sharebotの特徴

- Sharebotは、Shareネットワーク内の一つのノードとして動作するクローラー
- ShareネットワークをクロールしながらIPアドレスやファイル情報を収集
- 作成されたDBを検索することで、指定ファイルを保持するIPアドレスを特定可能

- Windows 2000/XP/2003上で動作
- マルチスレッドで高速クロール
- 簡単に操作できるGUI
- 日本語化
- クロール結果や検索結果のエクスポート
- 個人には無償提供
  (法人の場合はeEye顧客にのみ無償提供)

eEye Sharebot Version: 0, 1, 11, 12

THE SOFTWARE IS PROVIDED ON A STRICTLY "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY WARRANTY THAT THE SOFTWARE IS FREE OF DEFECTS, VIRUS FREE, ABLE TO OPERATE ON AN UNINTERRUPTED BASIS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING.

Coded by Yuji Ukai <yukai@eeye.com>

Copyright (C) 1998-2007 eEye Digital Security

ハードウエア環境

- Windows® 2000 SP4, Windows XP SP2 **もしくは** Windows 2003 SP1
- Intel® Pentium 4 1GHz CPU **もしくは同等の**CPU
- Internet Explorer version 5.0 **もしくはそれ以上のバージョン**
- **メモリ**512MB, 1024 MB**以上推奨**
- **ハードディスク**1GB **もしくはそれ以上**
- **画面解像度**1024 x 768

※ 全てのノード情報とキー情報をメモリに格納。ノードとキーの重複チェックを高速化。このため、多くのメモリエリアを必要が必要

※ ノードとキーはファイルにも出力されるため、多くのディスク容量が必要

動作環境の注意点

- **各自Share.exeを入手**

  SharebotはShare EX2 のAPIを内部で呼び出す
  (Shareは実行されない)
  Share EX2を別途入手

- **NAT下での動作**

  NAT下のプライベートアドレス環境でも動作する
  Sharebotが外部から接続を受けることは無い
  ポートフォワーディングの設定等は不要
  ただし、設定次第でNATテーブルが溢れる可能性があり

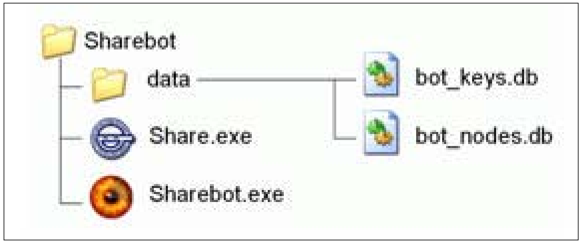
- **不完全な外向きのTCPコネクション数の制限**

  Windows XP SP2 、Windows 2003 SP1以降を
  利用する場合、Biot併用を推奨

ディレクトリ構造

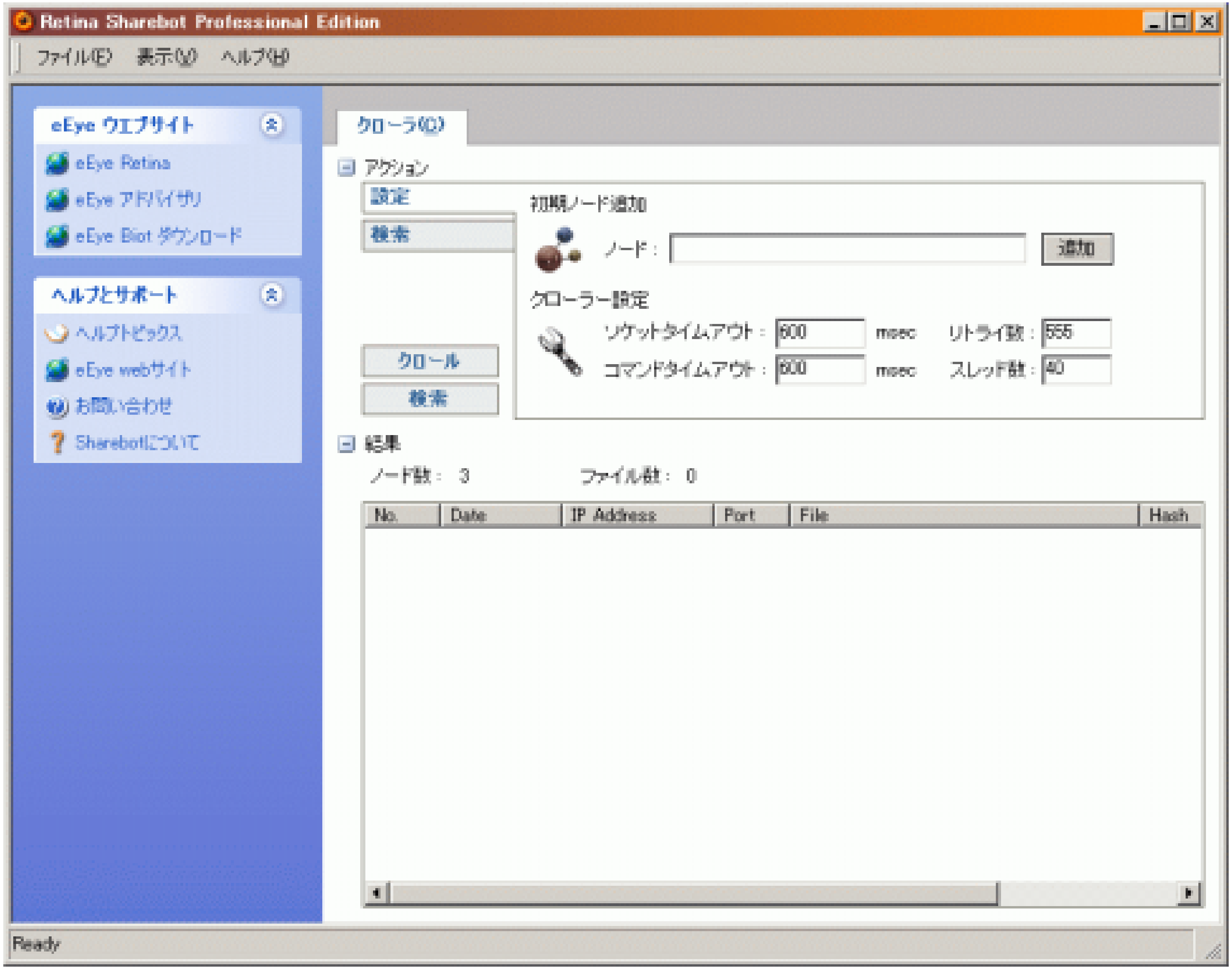
Share EX2の実行ファイルイメージ (Share.exe) を
Sharebotのインストールディレクトリにコピー

(「ダウンロード share ex2」をキーワードにgoogle検索)



Sharebot.exe	Sharebot 実行ファイル
Share.exe	Share EX2 実行ファイル
bot_keys.db	キーデータベース
bot_nodes.db	ノードデータベース

Retina Sharebotの起動



ソケットタイムアウト : 600 msec

リトライ数 : 555

コマンドタイムアウト : 600 msec

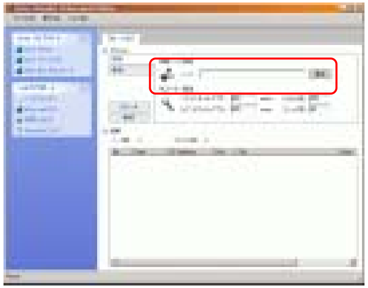
スレッド数 : 40

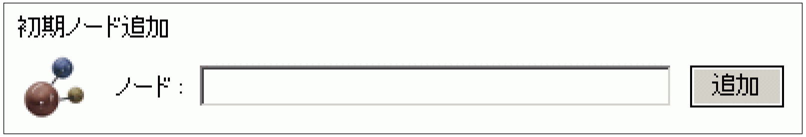
ノード数 : 3

ファイル数 : 0

初期ノードを追加する

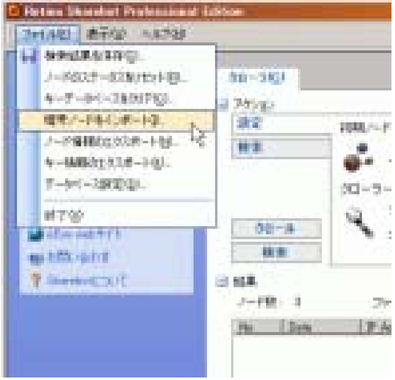
- 初期ノード追加





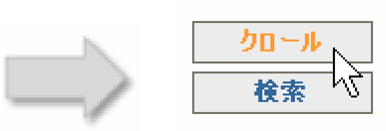
- IPアドレス:ポート形式
  (例) 111.22.33.4:5678
- Share暗号ノード形式
  (例) goH/gL+UCKOyjfm9UA2tct5Q・・・・

- 暗号化された初期ノードリストをインポート



nodes.txtファイルを読み込み
「share ex2 初期ノード」をキーワードにgoogle検索

クロール開始



⊟ 結果

ノード数： 76374　ファイル数： 10291

クロールが進むと、
ノード数、ファイル数が増加

Biot**が起動していない場合**

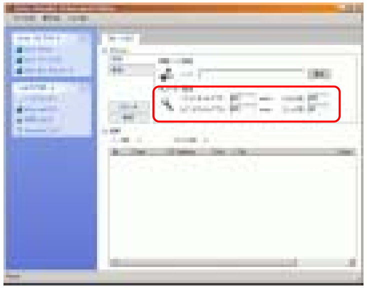
Biotが起動していません

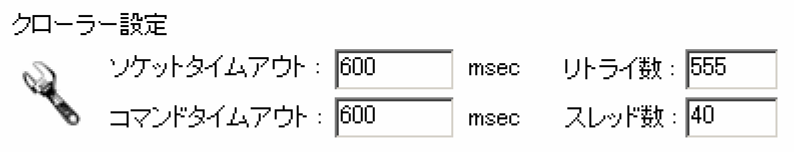
Biotが起動していません。

Windows XP SP2、およびWindows 2003 SP1 以降のシステムでは
不完全な外向きのTCP同時接続数が制限されています。
Biotを起動する事でこの制限を回避し、高速にクロールできます。

☐ 今後確認を行わない

パラメータの設定





クローラー設定

ソケットタイムアウト： 600 msec リトライ数： 555

コマンドタイムアウト： 600 msec スレッド数： 40

- **ソケットタイムアウト**
  TCP**の接続、送受信のタイムアウト値**

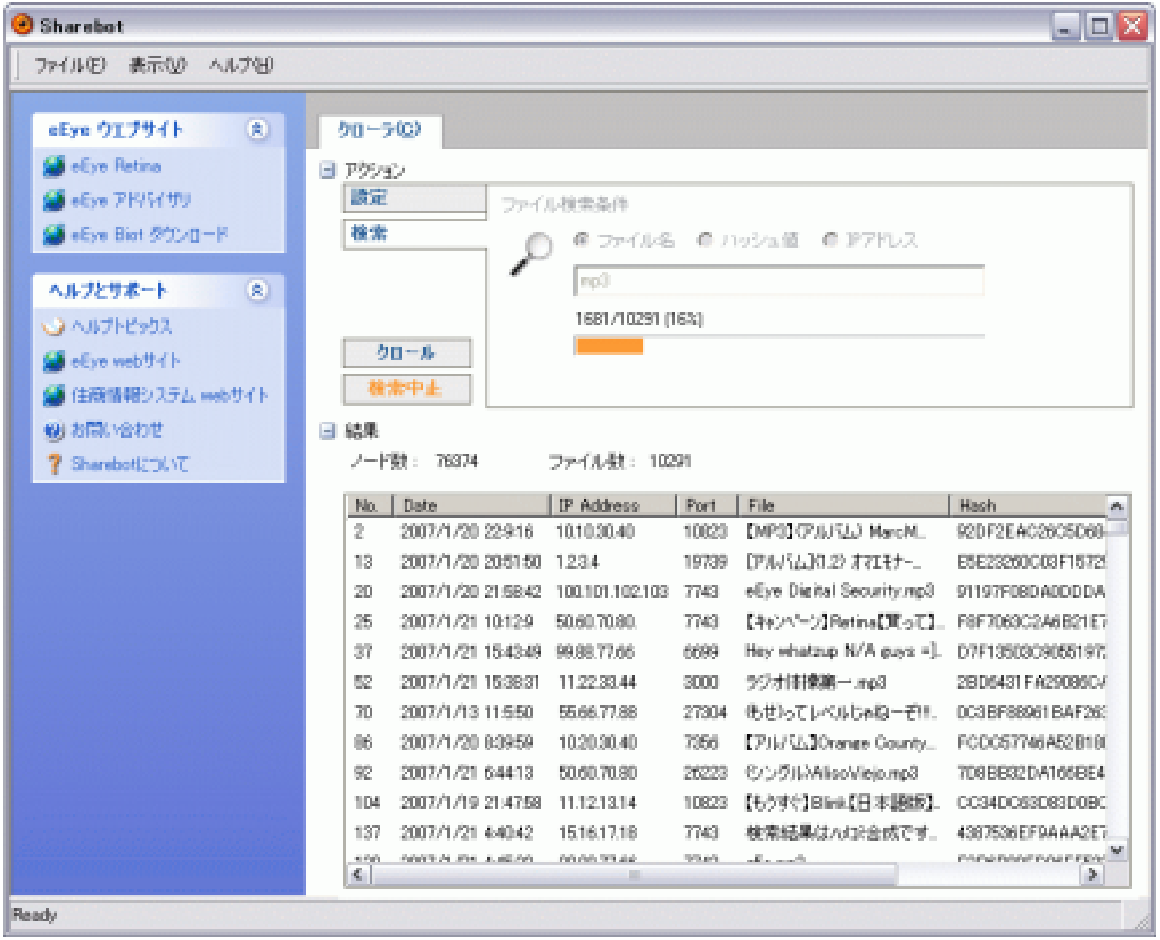
- **コマンドタイムアウト**
  Share**ノードからのコマンド受信のタイムアウト値**

- **リトライ数**
  接続失敗、コマンド受信失敗の際のリトライ数
  256**以上で無制限リトライ**

- **スレッド数**
  最大スレッド数
  スレッド数を増やすことによりネットワークを高速にクロール(NAT**テーブルの限界に注意**)
  Personal**は最大**40、Professional**は無制限**

キーデータベース検索

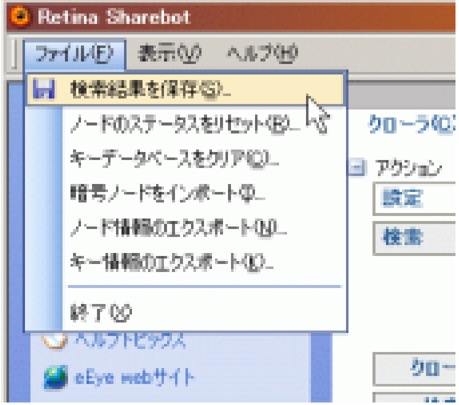
検索



ノード数： 76374　ファイル数： 10291

No.	Date	IP Address	Port	File	Hash
2	2007/1/20 22:9:16	10.10.30.40	10023	【MP3】(アルバム) MarcM...	92DF2EAC26C5D6
13	2007/1/20 20:51:50	1.2.3.4	19739	[アルバム](1.2) オマエモナー...	E5E23260C03F1572
20	2007/1/20 21:58:42	100.101.102.103	7743	eEye Digital Security.mp3	91197F08DA0DDDA
25	2007/1/21 10:1:29	50.60.70.80	7743	【キャンペーン】Retina【買って】...	F8F7063C2A6B21E7
37	2007/1/21 15:43:49	99.88.77.66	6699	Hey whatsup N/A guys =]...	D7F13503C9055197
52	2007/1/21 15:38:31	11.22.33.44	3000	ラジオ[特集]一.mp3	2BD6431FA29086C
70	2007/1/13 11:5:50	55.66.77.88	27304	(もせりってレベルじゃねーぞ!!...	0C3BF88961BAF26
86	2007/1/20 8:39:59	10.20.30.40	7356	【アルバム】Orange County...	FCDC57746A52B1
92	2007/1/21 6:44:13	50.60.70.80	26223	(シングル)AlisoViejo.mp3	7D8BB32DA166BE4
104	2007/1/19 21:47:58	11.12.13.14	10823	【もうすぐ】Blink【日本語版】...	CC34DC63D83D0BC
137	2007/1/21 4:40:42	15.16.17.18	7743	検索結果はハメ合成です...	4387536EF9AAA2E7

検索結果の保存



CSV, XML **形式に対応**

Sharebot Professional版のデータベース

- **高速化のため、キー情報とノード情報をメモリに格納**
  格納できるキー数とノード数は40**万個に制限**

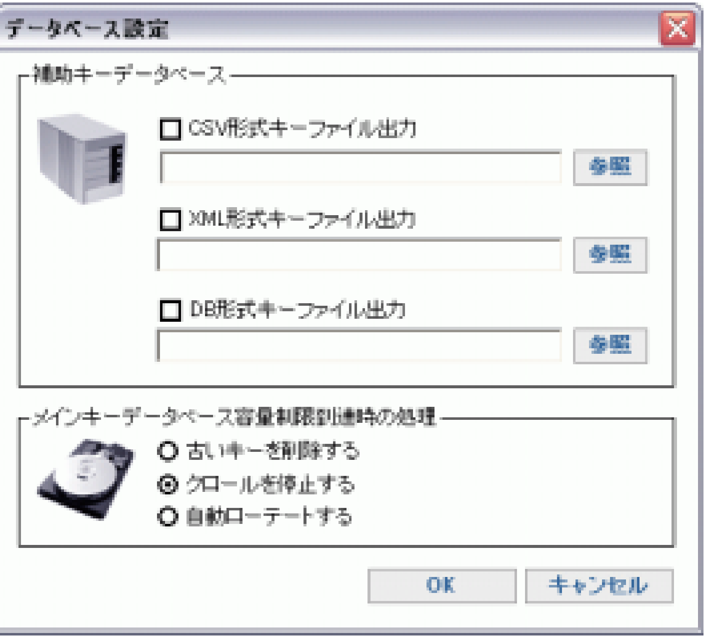
- Personal**版**
  \- **キー数が上限値に達するとダイアログ表示。クロール停止**
  \- **再開する場合はキーデータベースを手動でバックアップ、クリア**

  長期間の無人連続運用ができない
  大規模運用にも不向き

- Professional**版**
  \- **無制限にキーを保存する補助データベースを作成可能(重複チェックは無い)**
  \- **メインのキーデータベースも、自動ローテートや古いキーの自動削除が可能**

  長期間の無人連続運用が可能

Professional版のデータベース設定



データベース設定

補助キーデータベース

□ CSV形式キーファイル出力　[参照]

□ XML形式キーファイル出力　[参照]

□ DB形式キーファイル出力　[参照]

メインキーデータベース容量制限到達時の処理

○ 古いキーを削除する
● クロールを停止する
○ 自動ローテートする

[OK] [キャンセル]

- **補助キーデータベース**

  - **容量制限などはない**
  - CSV、XML、DB(**独自**)**形式で作成可**
  - **重複チェックが無いため注意**

- **メインキーデータベース**

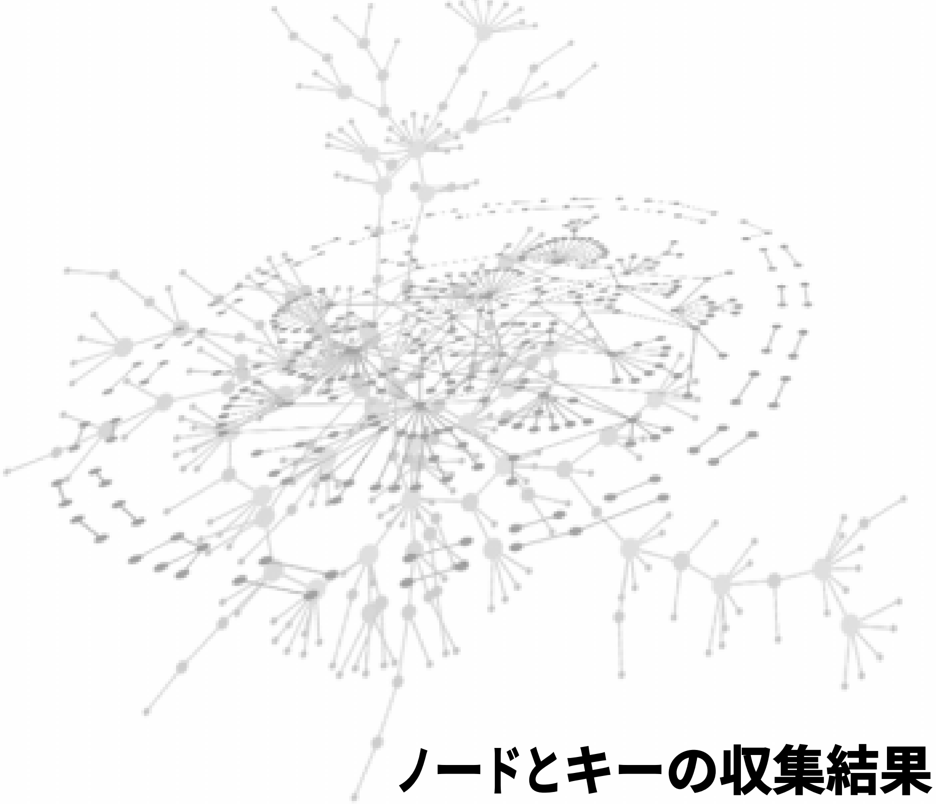
  - **古いキーを削除する**
    40**万到達時、下位**10%**を自動削除**(**推奨**)

  - **クロールを停止する**
    Personal**版と同じ設定**(**デフォルト**)

  - **自動ローテートする**
    タイムスタンプ付きのファイルにバックアップ

結果

Performance



ノードとキーの収集結果
今後の課題
さいごに

ノード情報、キー情報の収集結果

2007年2月21日～3月13日にかけ、10台のRetina Sharebotを用いてクローリングを実施

	Share	参考値 Winny
ノード数	約14万ノード	約35万ノード
オリジナルファイル数	約40万ファイル	約470万ファイル

ノード数、オリジナルファイル数は1日あたりの値。

Winnyについては、同様なクローリング手法により、
2006年10月6日～10月15日にかけ実施した観測結果。

出典：P2Pファイル交換ソフトウェア環境を対象とした観測に関する一考察 (SCIS2007)

今後の課題

- Share のより詳細な解析
- 実験目的の長期観測体制の整備
- 大規模運用のための機能強化 (SQL対応など)
- ダウンロード機能 (証拠保全用)
- エンジン部のプラグイン化 (他のP2PやShare EX3への対応)

さいごに

- **ファイルを削除できない**P2P**ファイル交換共有システムは危険**
  ファイル削除機構が実装されるか、あるいは、淘汰されるべき

- **クローラー** vs **同種の新**P2P**ファイル交換共有システム**
  「イタチゴッコ」になりかねないが、現状を放置できない

- **法的アプローチ**
  一刻も早いウイルス作成罪の新設が望まれる

eEye Digital Security **は、日本の情報漏えい問題において、**
公共性を重視したソリューション提供を今後も継続する

Thank you for your attention !

Question ?

Contact : Yuji Ukai <yukai@eeye.com>