Winny ネットワークにおける安全で効率的な著作権侵害監視について

株式会社クロスワープ
2007 年 1 月

目次

1. 始めに 3
2. ツール 3
   2.1. キー情報収集ツール 3
   2.2. 専用ダウンロードツール 3
3. 技術検証と結果 4
   3.1. 技術検証の方針 4
   3.2. 検証１　広域キー情報観測による絞込み 4
      3.2.1. 検証手法 4
      3.2.2. 機器構成 4
      3.2.3. 結果 4
      3.2.4. 得られた知見 5
   3.3. 検証２　特定ノードへの継続接続による追跡監視 6
      3.3.1. 検証方法 6
      3.3.2. 機器構成 6
      3.3.3. 結果 6
      3.3.4. 得られた知見 7
   3.4. 検証３　専用ダウンロードツールによる判定 8
      3.4.1. 検証方法 8
      3.4.2. 機器構成 8
      3.4.3. 結果 8
      3.4.4. 得られた知見 11
4. 監視手法の提案 12

1. 始めに

Winny ネットワークでは著作権侵害が蔓延しており、その被害が顕在化している状況にある。しかし、従来からの FTP サイト、Web サイトによる著作権侵害は主体が特定しやすくクローリング等の対策を講じやすいが、Winny ネットワークではファイル中継による匿名性の為にユーザがファイルを保持しているのかどうか特定しにくく、対策は難しいとされてきた。そこで本レポートでは、監視システムによるネットワークへの悪影響を極力防ぎつつ、ユーザがキャッシュ、アップロードに関わらずファイルを保持しているかどうかを特定する技術検証を行い、安全で効果的な著作権侵害監視を実施する手法の提案を行う。なおこのレポートは社団法人コンピュータソフトウェア著作権協会に観測ツールの提供と調査依頼を受け、株式会社クロスワープが作成した。

2. ツール

提供された Winny ネットワークの観測ツールについて以下に記述する。

2.1. キー情報収集ツール

Winny のキー情報を収集し、ファイルに出力するコマンドラインツール。Winny ネットワーク上のいずれかのノードに接続した後、拡散クエリ送信要求（コマンド 0x0A）を送信し、クエリ送信要求（0x0D）を受信する事を高速に繰り返し、Winny ネットワーク上で流通しているキー情報を網羅的に収集する。以下にキー情報として収集可能な項目を示す。

- ノードの IP アドレス、ポート番号
- BBS スレッド管理ノードの IP アドレス、ポート番号
- ファイルサイズ
- キー更新日時
- キー消滅判定タイマー（TTL）
- 被参照数
- ファイルハッシュ値
- トリップ情報

2.2. 専用ダウンロードツール

Winny ノードの IP アドレス、ポート番号、ファイルハッシュ、ファイルサイズを指定して、そのノードから指定したファイルをダウンロードするコマンドラインツール。64KB ごとのブロックに分けてファイルの取得と MD5 による破損チェックを行い、全てのブロックがダウンロードできたところで結合を行いファイルを復元する。

3. 技術検証と結果

3.1. 技術検証の方針

Winny 作者の金子氏による「Winny の技術」には、「あるノードが完全キャッシュかアップロードファイルを保持している場合、キー情報の IP アドレスは常にそのノードの IP アドレスに書き換えられる」という記述がある。また、TTL は初期に 1500 秒程度にセットされており、時間の経過で減っていくため、ある程度長期にわたって同一キー情報（同一 IP アドレスが同一ファイルを保持しているという情報）が観測された場合、高い確率でその IP アドレスのノードはキャッシュかアップロードファイルを保持していると推定できる。この考え方を実測するため、以下の技術検証を行った。

１．広域キー情報観測による絞込み
２．特定ノードへの継続接続による追跡監視
３．専用ダウンロードツールによる判定

3.2. 検証１　広域キー情報観測による絞込み

3.2.1. 検証手法

あらかじめ特定のダミーファイルを公開ノードのアップフォルダに配置して公開した上で、観測システム側で特に観測対象を限定せずに網羅的にキー情報を収集し、収集されるキー情報の中にダミーファイルと公開ノードの情報が含まれる事を確認する。公開したダミーファイルを「ファイル（α）」とする

3.2.2. 機器構成

Winny を動作させる PC (X)と Winny ネットワーク全域を観測する Server(A)、Server(B)で構成した。それぞれが独自のインターネット回線を用いている。

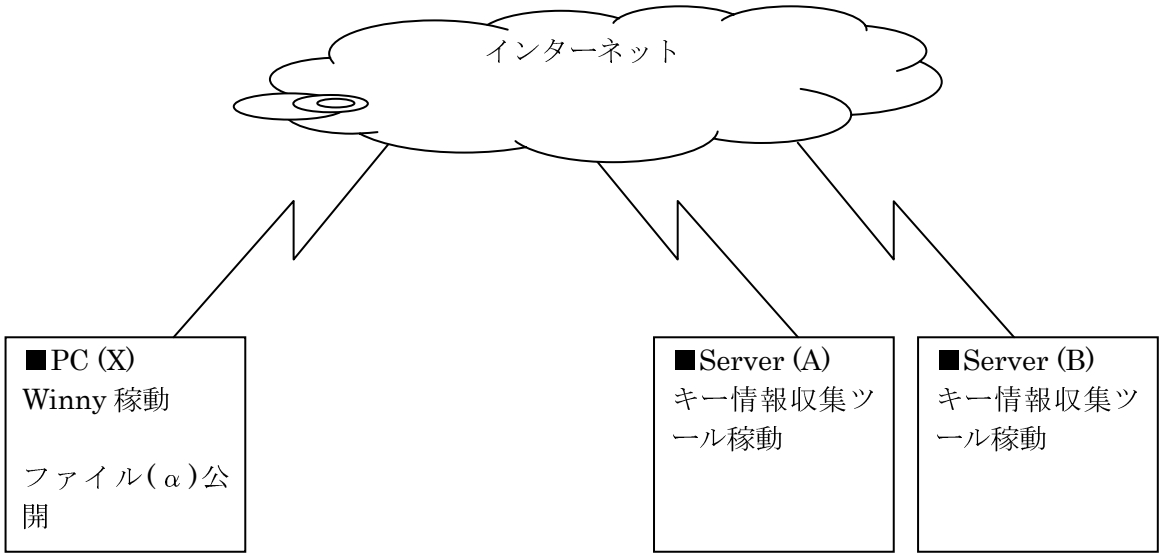

図 1　検証 1 ネットワーク構成

3.2.3. 結果

検証期間中にキー情報収集ツールによって検出されたファイル（α）のハッシュ値を含むキーは以下の通りである。

ノード	検知時刻	TTL
A	1 日目 20:12:11	323
A	1 日目 21:59:02	202
A	1 日目 22:32:29	121
A	2 日目 00:27:46	398

A	2 日目 01:49:42	114
B	2 日目 13:32:29	431
C	2 日目 22:19:57	280
C	2 日目 23:06:19	976
D	3 日目 02:09:13	343
E	3 日目 09:03:20	2
E	3 日目 21:53:35	61
E	4 日目 00:04:59	836
E	4 日目 08:34:56	689
F	4 日目 11:34:20	76
C	5 日目 01:51:20	878

このうち 3 回検出されたノード C が実際の初期公開ノード、つまり PC (X)であった。また、他にノード C が完全キャッシュを保持していた 3 個のファイルについてキーの検知状況を調査したところ、以下のような結果が得られた。

・ファイル 1 のハッシュ値を含むキー
総検知回数：746 回
検知 IP アドレス数：327 回
ノード C の IP アドレス検知回数：53 回

・ファイル 2 のハッシュ値を含むキー
総検知回数：819 回
検知 IP アドレス数：371 個
ノード C の IP アドレス検知回数：33 回

・ファイル 3 のハッシュ値を含むキー
総検知回数：351 回
検知 IP アドレス数：118 個
ノード C の IP アドレス検知回数：47 回

3.2.4. 得られた知見

上記結果より、Winny ネットワークを網羅的に観測した上で特定ファイルハッシュ値の検知回数が多いノードを抽出する事で、全検知ノードから実ファイルを保持しているノードを絞り込む手法は有効と考えられる。ただし今回の検証では初期公開ノードが時間的に遅れて見つかった事から、この手法だけでは初期公開ノードの特定は難しい。キー情報収集ツールが動作するサーバを増やして探索性能を上げることで、ある程度初期公開ノードを早い段階で見つけることも可能になると考えられる。

3.3. 検証２　特定ノードへの継続接続による追跡監視

3.3.1. 検証方法

次に、強制的に特定ノードのみに接続して観測した場合、より詳細にそのノードが保持するキャッシュを推定できると考えられる。そこで観測システムから監視対象ノードだけに接続できるようアクセス制限を行い、継続してキーの取得、集計を行った。

3.3.2. 機器構成

Winny を動作させ、実キャッシュを複数個、アップフォルダにもファイルを 2 つ持った PC (X)と PC(X)への静的ルーティングしか持たず、常にキー情報収集ツールで PX(X)に接続して情報を取得する Server(B)で構成した。それぞれが独自のインターネット回線を用いている。

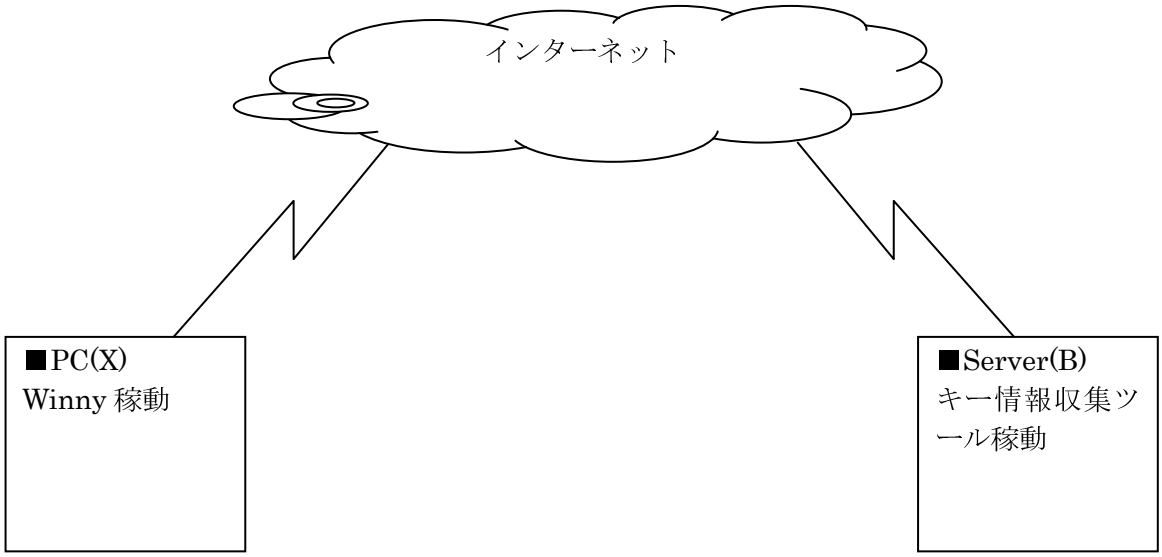

図 2　検証 2 ネットワーク構成

3.3.3. 結果

収集されたキーから PC(X)の IP アドレスを含むキーのみを抽出し、ハッシュ値を検知回数の多い順に並べたところ、顕著な偏りが見られた。

検知回数	キー数	割合
1	17171	97.28%
2	423	2.40%
3	31	0.18%
4	5	0.03%
5	2	0.01%
6	3	0.02%
7	2	0.01%
8	1	0.01%
15	1	0.01%
16	1	0.01%
17	1	0.01%
18	1	0.01%
21	1	0.01%
22	1	0.01%
25	1	0.01%
28	1	0.01%

32	2	0.01%
33	1	0.01%
34	1	0.01%
36	1	0.01%
40	1	0.01%
総数	17652	

キーが 1 回しか検知されなかったハッシュ値が 97%を占めたが、数回～数十回以上にわたってキーが観測される観測されるハッシュ値も存在し、明確に分布が分かれた。ここでキーが通算 4 回以上検知されたハッシュ値を対象にし、監視対象ノード上のキャッシュフォルダ、アップフォルダにファイルが存在するかどうかの確認を行った。

検知回数	ハッシュ値(一部置換済み)	ファイルサイズ	検証期間終了時点での実ファイルの有無
4	xxxxxf01efb19abf8e03ddb176d9318e	297725952	なし
4	xxxxx956db450323c9c065334ee6b73c	19492111	部分キャッシュが存在
4	xxxxxa0a3e2322c13a7565e3814ba4a5	439307138	部分キャッシュが存在
4	xxxxx57c8097c7ce2be9bc9248a79c9d	219865494	部分キャッシュが存在
4	xxxxx49fc476c6cf5c0e5b9289f0d11a	20078516	部分キャッシュが存在
5	xxxxxd222fd43c7c9c468ef879c51153	1670873180	なし
5	xxxxx61d503dc80130f5a0d15f7ed1b0	18834607	部分キャッシュが存在
6	xxxxx882411a2b791c27553b209094b6	5413526	完全キャッシュが存在
6	xxxxx583ca36190595af39bcc41de4f	13516701	完全キャッシュが存在
6	xxxxx18c0b17aff019fe19d4f93bd632	5082138	完全キャッシュが存在
7	xxxxx11b2fa3c432d29c73411b007b15	286586947	完全キャッシュが存在
7	xxxxxe4ff78b401d45504a699626f348	5695000	アップフォルダに存在
8	xxxxxd92af8a571c8dd4ceea987e9928	117448	完全キャッシュが存在
15	xxxxx73fb798ea756cd7518d7117b1e1	4399440	完全キャッシュが存在
16	xxxxx63f213e9885c7c259dfa578adc5	5636942	完全キャッシュが存在
17	xxxxx7c766359027c639ea5cbe0e1954	10015956	完全キャッシュが存在
18	xxxxx506dce3eb7e771c762bcf86e989	249363314	完全キャッシュが存在
21	xxxxxf9e9e4ca90c8a213d09c1475c0f	6463120	完全キャッシュが存在
22	xxxxx052a2acc6654723e93f15f4396c	6631394	完全キャッシュが存在
25	xxxxxc9e48ca5b4ededba06e768643a1	3591077	完全キャッシュが存在
28	xxxxx922e131f168417716a98b61993f	6423157	完全キャッシュが存在
32	xxxxx369ceb4cb794b197d6b2b1a15fd	7055208	完全キャッシュが存在
32	xxxxx700a4a08abe269ae3984705174b	311070724	完全キャッシュが存在
33	xxxxxe0eeb0f0ac07ee7dc7e91e85969	6631394	完全キャッシュが存在
34	xxxxxd016429b55503f58f4663fac8c7	1208320	完全キャッシュが存在
36	xxxxxb052e9d0a258dfd25f93873406c	8177625	完全キャッシュが存在
40	xxxxx5371a0b276ee009ec8157685270	6423157	完全キャッシュが存在

6 回以上検知されたキーに関しては全て、実ファイルが監視対象ノード上に完全キャッシュかアップフォルダのファイルとして存在した。逆に 4 回、5 回の場合では部分キャッシュか、キャッシュ自体が見つからない結果となった。また、キャッシュが全く見つからなかった 2 つのキーについては検知時間を調べたところ、数時間以上にわたって検知されていた。

3.3.4. 得られた知見

数日にわたって特定ノードを追跡監視した場合、特定ハッシュ値を含むキーの検知回数による実ファイル保持判定は非常に有効と考えられる。ただし検知回数と検知間隔の閾値をどの程度にするかは、観測期間及び観測対象とす

るファイルサイズにより検討の余地がある。

3.4. 検証３　専用ダウンロードツールによる判定

3.4.1. 検証方法

前述の検証手法によってほぼ特定されるとは言え、完全にそのノードがファイルを持っているとは断定できない。そこで専用ダウンロードツールを使って、さまざまな状態のキーを対象に実際に Winny ノードからファイルを取得できるかどうかを検証した。なお、キャッシュを保持しているかどうかはダミーファイルを公開している Winny の検索ウィンドウ上の情報から判断した。

3.4.2. 機器構成

Winny を動作させる PC(A)、(B)と専用ダウンロードツールを動作させる PC(C)を用意、それぞれにグローバル IP アドレスを振って１つのハブで接続した。

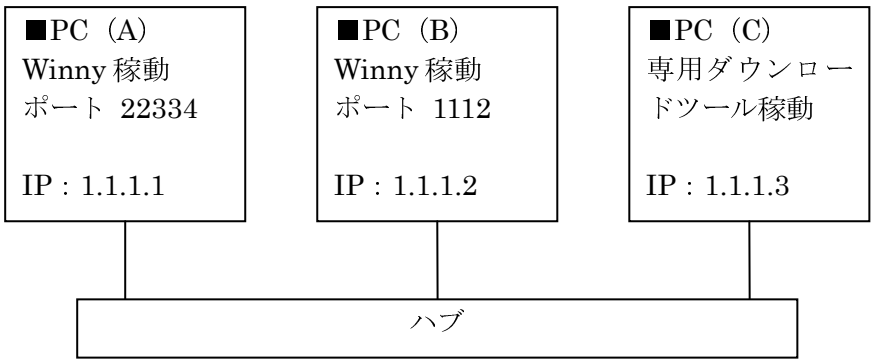

図 3　ダウンロードテスト環境

3.4.3. 結果

3.4.3.1. 試行１　アップフォルダにあるファイルのダウンロード

PC（A）上の Winny のアップフォルダにファイル（α）を置いて、PC（C）から専用ダウンロードツールを用いてダウンロードを実施した。

コマンドライン：

```
>winnydownload 1.1.1.1 22334 "" 5695000 xxxxxe4ff78b401d45504a699626f348
---[0]-----------------------
    ID=0x023a394e
    dwBlock=0x00000000
    Length=65552
---[1]-----------------------
    ID=0x023a394e
    dwBlock=0x00000001
    Length=65552
    Calc-MD5：140984D964512174A49740CF6076EDF6
    Hdr-MD5　：140984D964512174A49740CF6076EDF6
・・・
中略
・・・
---[53]-----------------------
    ID=0x182e2495
```

```
    dwBlock=0x00000035
    Length=65552
    Calc-MD5：1574C2063B91E92182BF46008E510606
    Hdr-MD5  ：DE075FAFC0EB1DC63797879BE56F780E
    MD5 failed. Try again.
・・・
中略
・・・
---[86]----------------------
    ID=0x182e2495
    dwBlock=0x00000056
    Length=58920
    Calc-MD5：8A404D541747EA347070DA07279496C9
    Hdr-MD5  ：8A404D541747EA347070DA07279496C9
CacheDecode error. r=-2
```

結果：
専用ダウンロードツールの起動直後からダウンロードが開始されたが、MD5 エラーが頻発した上で最終的にキャッシュデコードエラーが起きてしまい、ダウンロードは完了できなかった。複数回試行したところ、ブロック数は固定値だが、MD5 エラーが起きるブロックは毎回変化していた。

3.4.3.2. 試行 2　完全キャッシュのダウンロード

PC（B）上の Winny で一度 PC(A)のファイル(α)をダウンロードして完全キャッシュを生成した上で、PC（C）から専用ダウンロードツールを用いてダウンロードを実施した。

コマンドライン：

```
>winnydownload 1.1.1.2 1112 "" 5695000 xxxxxe4ff78b401d45504a699626f348
---[0]----------------------
    ID=0x6ea54f6a
    dwBlock=0x00000000
    Length=65552
---[1]----------------------
    ID=0x6ea54f6a
    dwBlock=0x00000001
    Length=65552
    Calc-MD5：140984D964512174A49740CF6076EDF6
    Hdr-MD5  ：140984D964512174A49740CF6076EDF6
・・・
中略
・・・
---[33]----------------------
    ID=0x6ea54f6a
    dwBlock=0x00000021
    Length=65552
    Calc-MD5：4E6C540EEC8ECE059FDDA135FC2186BF
    Hdr-MD5  ：570715393E4288D106423A9D0EFDC9E0
    MD5 failed. Try again.
・・・
中略
```

・・・

結果：
PC(C)での表示上、試行 1 と全く変わらない結果が得られた。

3.4.3.3. 試行 3　部分キャッシュのダウンロード
PC（A）上の Winny の検索画面から部分キャッシュとして確認できたファイル（β）を、PC(A)をあて先指定して PC（C）から専用ダウンロードツールを用いてダウンロードを実施した。

結果：
専用ダウンロードツールの起動直後からダウンロードが開始されたが、MD5 エラーが頻発した。ブロック数が途中の特定値まで行ったところで処理が止まってしまったようだった。複数回試行したところ、処理が停止するブロックは固定だが、MD5 エラーが起きるブロックは毎回変化していた。

考察：
ダウンロード自体は完全キャッシュと同様に行われるが、部分キャッシュのため、PC(A)が保持しているところまで来たところで処理が止まったように思われる。

3.4.3.4. 試行 4　アップファイルの中継
PC（B）の Winny に PC(A)をノード登録した上で PC（A）上の Winny のアップフォルダにファイル（α）を置いき、PC(B)から検索して仮想キャッシュとして画面に表示される事を確認した。同時に PC(B)のキャッシュフォルダにファイル（α）が無い事も確認したうえで PC(B)をあて先指定して PC(C)から専用ダウンロードツールを用いてファイル（α）のダウンロードを実施した。
同時に PC(C)上でパケットキャプチャを行い、通信先を観測した。

結果：
PC(C)上では専用ダウンロードツールの起動直後からダウンロードが開始されたが、MD5 エラーが頻発した上で最終的にキャッシュデコードエラーが起きてしまい、ダウンロードは完了できなかった。複数回試行したところ、ブロック数は固定値だが、MD5 エラーが起きるブロックは毎回変化していた。ここまでは試行 1 と変わらない。

パケットキャプチャダウンロードの開始から終わりまで PC(C)は PC(B)としか通信しておらず、PC(B)がファイルを中継した事は疑いようが無い。またこの過程で PC（B）上ではファイル（α）の完全キャッシュが生成された。

3.4.3.5. 試行 5　アップファイルの中継 2（ダウンロード途中停止）
試行 4 と同様の形で PC(C)からダウンロードを開始後、数ブロックが取得された時点でダウンロードを手動で停止し、PC(B)上の Winny の挙動を確認した。

結果：
PC(B)は PC(C)がダウンロードを停止した後も PC(A)からファイル（α）の取得を続け、PC(B)上にファイル(α)の完全キャッシュが生成された。

3.4.3.6. 試行 6　部分キャッシュの中継
試行 3 で使った部分キャッシュを、PC(B)に対して専用ダウンロードツール用いてダウンロード試行してみる。

結果：
ダウンロードは始まらなかった。

考察：
PC(B)の Winny からも PC(A)上の部分キャッシュの情報を持つキーは取得できなかった。この事から、PC(A)から

は部分キャッシュのキーはノード外に配布されないと推測できる。キーが無いため、PC(C)のダウンロード要求をPC(B)は PC(A)に転送せず、ダウンロードと中継が起きなかったのだと推測できる。

3.4.4. 得られた知見

専用ダウンロードツールを用いて、接続直後からファイル（の断片）が取得できる場合、高い確率でそのノードはファイルを保持している事が推定できる。ただし接続先ノードが中継ノードとしてすぐに別ノードからファイルをダウンロードできる場合も接続直後からダウンロードが始まるため、必ずしも接続先ノードが最初からファイルを保持しているとは限らない。

また、中継が起きた場合は専用ダウンロードツールを用いていても中継ノードにキャッシュが残り、ダウンロードを途中でやめたとしても中継ノードはダウンロードを続け、完全キャッシュになるまでファイルを取りきる事が確認できた。

4. 監視手法の提案

従来 Winny はリモートからファイルの保持が特定できないとされていたが、ここまでの検証により高い精度の特定ができた為、実際の監視手法として以下の方法を提案する。

①広域観測を行い、特定ファイルを保持している可能性が高いノードを絞り込む。
②絞り込まれたノードに対して継続観測を行い、キーが継続検知される事を確認する。
③ダウンロードツールで実際にダウンロードを行い、ファイルの保持を特定する。

①で大まかに実ファイルを保持しているノードを絞り込み、②で実ファイルの保持をほぼ特定して監視システムによる中継の危険性を極力排した上で③でのダウンロードで特定する形である。これら①～③をシステム的に繰り返す事で機械的に効率良く、かつ安全な監視が実現できると考えられる。

謝辞
本調査およびレポート作成は、社団法人コンピュータソフトウェア著作権協会の依頼を受け実施した。本調査を進めるにあたり、有益な助言と協力を頂いた社団法人コンピュータソフトウェア著作権協会、eEye Digital Security の鵜飼裕司氏、ならびに関係者各位に深く感謝致します。

参考文献
１．金子 勇「Winny の技術」　ASCII
２．社団法人 コンピュータソフトウェア著作権協会 他「2006 年ファイル交換ソフト利用実態調査」
http://www2.accsjp.or.jp/news/pdf/p2psurvey2006.pdf
３．鵜飼 裕司 「Inside Winny ～ Winny の解析とそのセキュリティ脅威分析」
https://sec.scs.co.jp/eeye/shiryo.html