



PacSec 2008 Conference

Inside "Winnyp"

- Winnypの内部動作とネットワーク クロールリングシステムの全貌

Fourteenforty Research Institute, Inc.

株式会社 フォティーンフォティ技術研究所

<http://www.fourteenforty.jp>

シニアソフトウェアエンジニア 石山 智祥

はじめに

- ・ Winnypとは、情報漏えいや著作権法違反などで社会問題となったWinnyを改造したP2P型ファイル交換ソフト
- ・ Winnypは、Winnyとの互換性を持ち、設定を行うことでWinnyとの通信を行うことができる
- ・ Winnyp単体では、暗号鍵生成処理がWinnyに比べて複雑になっている
- ・ 今までWinnypを解析したという報告はあがっていない
- ・ 今回は、Winnypの暗号アルゴリズムについての解析結果と、クローリングシステム(WinnypRadar)の概要を報告



Agenda

1. Winnypの内部動作
2. 匿名P2Pアプリケーションの静的解析アプローチ
3. WinnypRadar – Winnypネットワーククローラ –



```
int packet_analysis(GDDCONFIG *gddc, unsigned char *packet, unsigned long length)
{
```

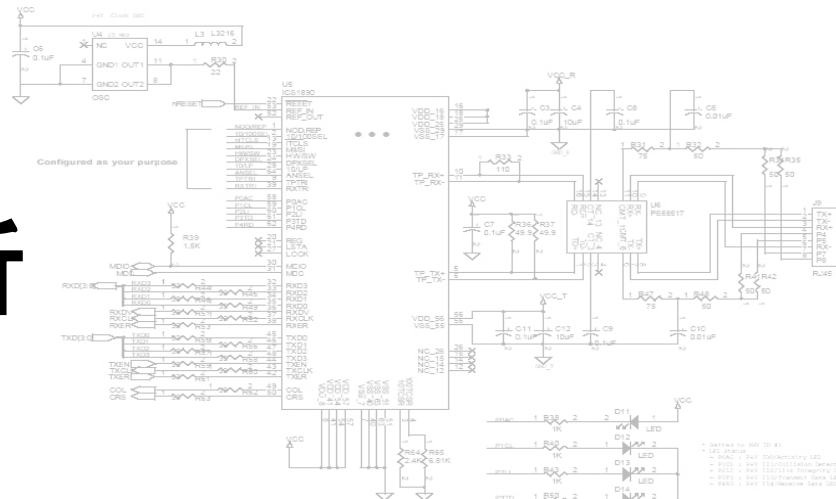
```
struct ip      *ip_header; /* IP header */
struct tcphdr *tcp_header; /* TCP header */
char          *tcp_data; /* TCP data */
struct in_addr addr; /* IP address */
char          sourceIP[16]; /* Source IP address */
char          destIP[16]; /* Destination IP address */
unsigned short sourcePort; /* Source Port */
unsigned short destPort; /* Destination Port */
unsigned long len_data; /* Length of data part */
unsigned long iph_len; /* Length of IP header */
unsigned long tcp_len; /* Length of TCP header */
unsigned long sequence; /* Expected sequence */
int           port; /* Indexnumber of port list */
char         dir; /* Direction */
unsigned char log; /* Log level */
CONN_LIST *conn_list; /* Connection list */
static char datestr[512]; /* Buffer to store datetime */
time_t      time_t;
struct tm   *timep=NULL;
char        *timep=NULL;
char        *c;
```

```
/* Get pointer of IP header and check length of IP */
if (length-SIZE_OF_ETHHDR < MINSIZE_IP+MINSIZE_TCP) return(0);
ip_header = (struct ip *) (packet+SIZE_OF_ETHHDR);
if (ip_header->ip_v!=IPPROTO_TCP)
|| ip_header->ip_v!=4) return(0);
iph_len = ((unsigned long) (ip_header->ip_hl))*4;
if (iph_len<MINSIZE_IP) return(0);
if ((unsigned long)ntohs(ip_header->ip_len) < MINSIZE_IP+MINSIZE_TCP)
return(0);
if ((unsigned long)ntohs(ip_header->ip_len) > length-SIZE_OF_ETHHDR) {
return(0);
}
```

```
/* Get pointer of TCP header and check length of TCP */
tcp_header = (struct tcphdr *) ((char *) ip_header+iph_len);
tcp_len = ((unsigned long) (tcp_header->th_off))*4;
tcp_data = (char *) tcp_header+tcp_len;
if (tcp_len<MINSIZE_TCP) return(0);
```

```
/* Get other parameter in TCP/IP header */
if (((long)ntohs(ip_header->ip_len)-(long)iph_len-(long)tcp_len)<0)
return(0);
len_data = ((unsigned long)ntohs(ip_header->ip_len)
-iph_len-tcp_len);
sourcePort = ntohs(tcp_header->th_sport);
destPort = ntohs(tcp_header->th_dport);
memcpy(&addr, &(ip_header->ip_sro), sizeof(struct in_addr));
strcpy(sourceIP, (char *) inet_ntoa(addr));
memcpy(&addr, &(ip_header->ip_dst), sizeof(struct in_addr));
strcpy(destIP, (char *) inet_ntoa(addr));
if (!strcmp(sourceIP, destIP)) return(0);
```

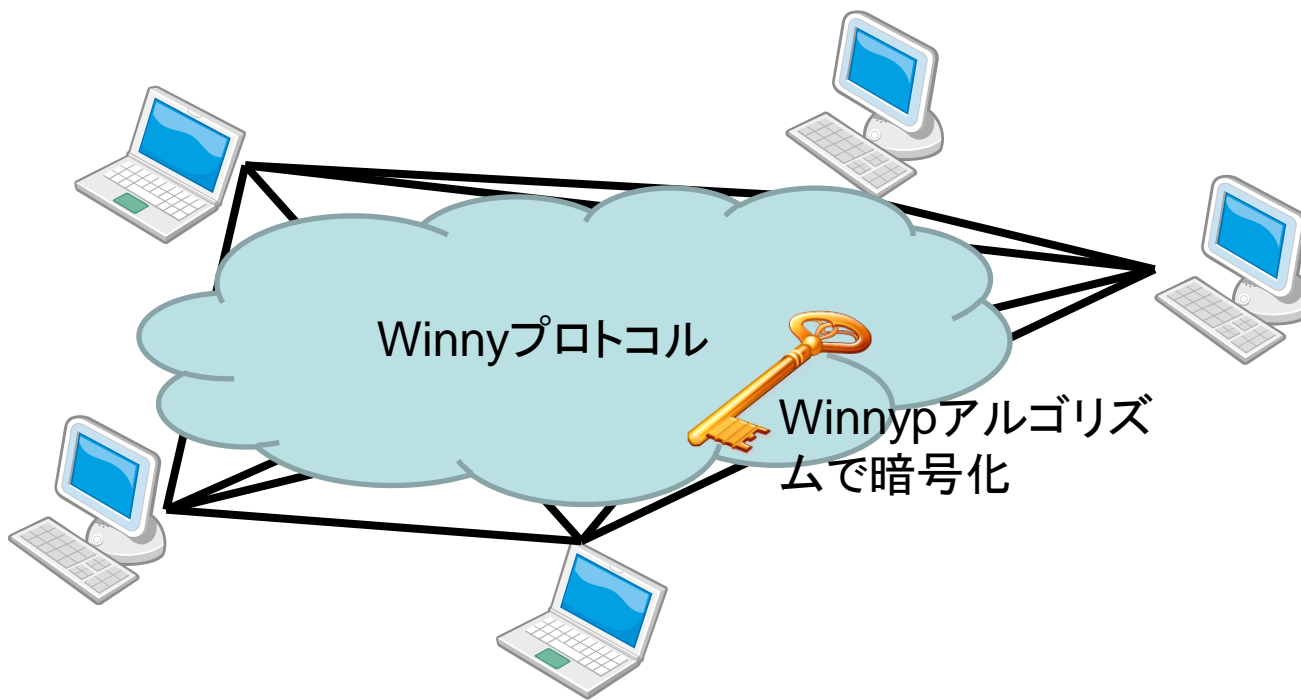
1. Winnypの静的解析



00001B70	FF 15 F0 11 00 01 E9 C0 03 00 00 E8 70 1C 00 00	銀...開...
00001B80	33 F6 56 E8 B1 FC FF FF 85 C0 0F 84 B7 03 00 00	3* 離...+...
00001B90	56 6A 02 FF 35 6C 80 00 01 FF 35 D0 87 00 01 FF	Vj...51...5...
00001BA0	15 CC 11 00 01 85 C0 75 D 68 10 10 00 00 FF 35	7...+u.h...5
00001BB0	50 80 00 01 FF 35 44 80 00 01 FF 35 D0 87 00 01	P...5D...5...
00001BC0	FF 15 04 12 00 01 FF 35 D0 87 00 01 FF 15 2C 12	...5...
00001BD0	00 01 FF 35 D4 88 00 01 FF 15 58 10 00 01 E9 64	...5t...X...誠
00001BE0	03 00 00 83 FE 1A 77 47 0F 84 59 03 00 00 83 FE	...wg...山...
00001BF0	11 0F 85 16 01 00 00 33 F6 39 35 E8 87 00 01 74	...3.95開...t
00001C00	22 88 3D 28 12 00 01 56 FF D7 56 FF D7 68 00 10	...(-...V.5V.5h...
00001C10	00 00 FF 35 50 80 00 01 FF 35 88 80 00 01 E9 7D	...5P...5*...霧
00001C20	02 00 00 6A 01 E8 0F FC FF E9 1A 03 00 00 88	...j...急
00001C30	70 14 88 11 01 00 00 3B F0 0F 87 88 00 00 00 3B	...ク...南...
00001C40	F0 0F 84 16 02 00 00 83 FE 1C 0F 85 B0 00 00 00	...
00001C50	33 F6 39 75 10 74 2F A1 EC 87 00 01 88 0D F0 87	3.9u.t/...*
00001C60	00 01 3B C6 75 08 3B CE 0F 84 D9 02 00 00 8B 3D	...+...珠...=
00001C70	14 12 00 01 51 50 68 B1 00 00 00 FF 35 D4 87 00	...QP7...5t...
00001C80	01 E9 56 01 00 00 8B 3D 14 12 00 01 68 F0 87 00	...翻...+...h...
00001C90	01 68 EC 87 00 01 68 B0 00 00 00 FF 35 D4 87 00	...h*...h...5t...
00001CA0	01 FF D7 A1 EC 87 00 01 88 0D F0 87 00 01 3B C1	...う*...+...+...
00001CB0	75 11 89 35 EC 87 00 01 89 35 F0 87 00 01 E9 84	u...5*...5*...驗
00001CC0	02 00 00 51 50 67 0F 01 00 00 8B CE B8 12 01 00	...QP...動...
00001CD0	00 2B C8 0F 84 3C 02 00 00 83 E9 04 0F 84 29 02	...+...<...>...
00001CE0	00 00 49 0F 84 F9 01 00 00 81 E9 1C 01 00 00 0F	...+...<...>...
00001CF0	84 E0 01 00 00 81 E9 E6 00 00 00 0F 84 3D 01 00	...+...<...>...
00001D00	00 81 E9 E8 7C 00 00 0F 84 02 01 00 00 3B 35 C0	*開...5t...
00001D10	88 00 01 0F 85 EE 00 00 88 45 14 8B 48 0C 88	...+...<...>...銀稀驚
00001D20	C1 8B D1 F7 D0 C1 EA 02 83 E0 01 83 E2 01 F6 C1	鏡*子*...*

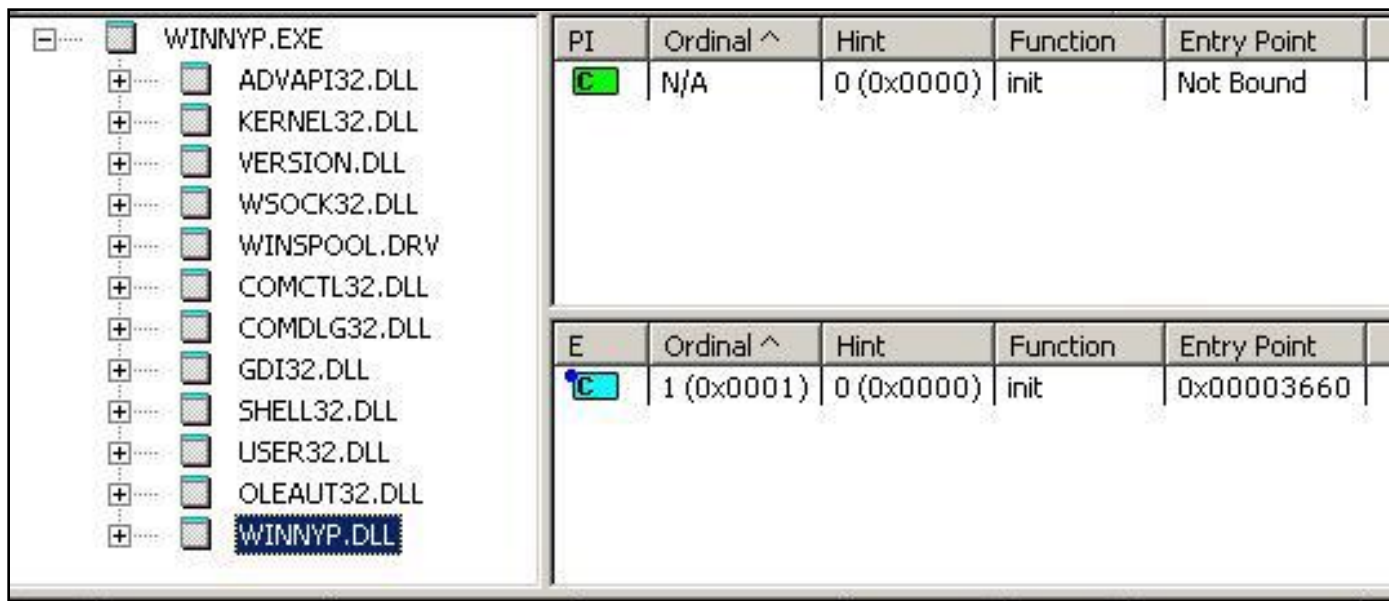
Winnypの動作概要


- ・ WinnypはWinny.exeを改造することによって作成されたP2Pファイル共有ソフト
- ・ Winnyと互換性を持ち、Winnyプロトコルを使用しているが、パケットの暗号アルゴリズムに独自のアルゴリズムを使用している




Winnypの動作概要

- Winnypは、Winnyの実行ファイルに対して、独自に作成したWinnyp.dllを読み込ませることにより動作する
- 修正したファイルからは、Winnyp.dllのinit関数のみを呼び出すようになっている

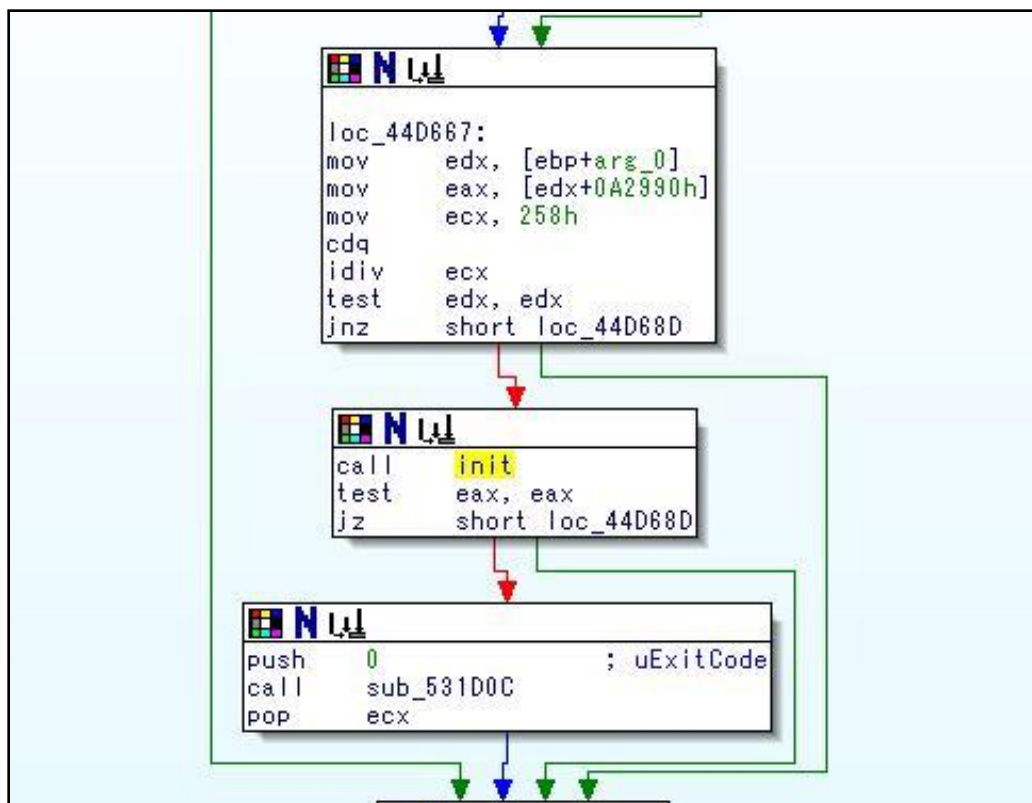


PI	Ordinal ^	Hint	Function	Entry Point
	N/A	0 (0x0000)	init	Not Bound

E	Ordinal ^	Hint	Function	Entry Point
	1 (0x0001)	0 (0x0000)	init	0x00003660

Winnypの動作概要

- Winnyp.exeからWinnyp.dllのinit関数をコールするために、Winnyに実装されていた関数が削除されている



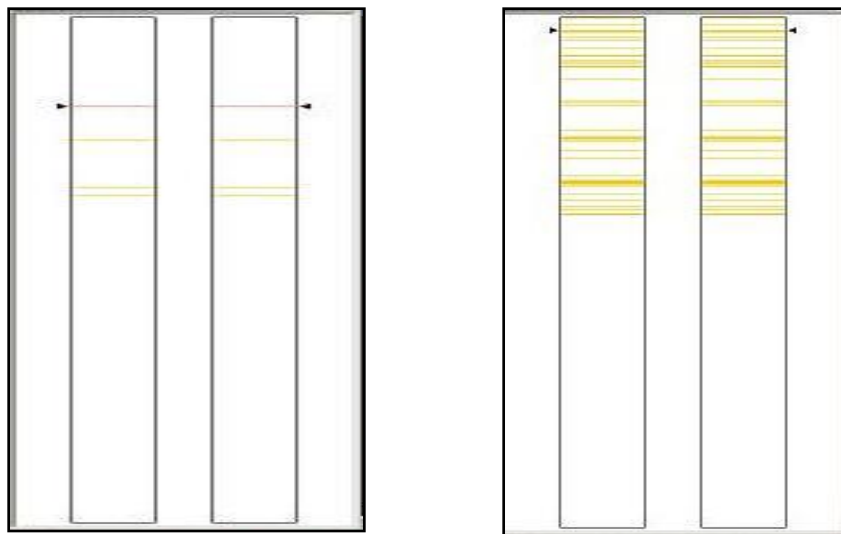


Winnypの初期化処理

- ・ Winnyp専用の設定ファイルを読み込み (disper.ini)
- ・ 暗号鍵生成処理で使用するパラメータの生成 (固定値が生成)
- ・ パケット送信時に使用するパラメータ生成
- ・ Winnyp.exeのコード領域へのパッチ処理

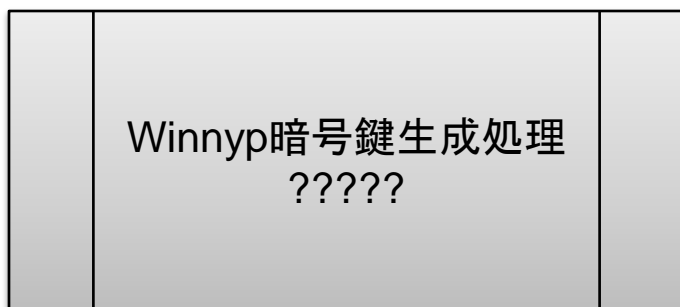
Winnypの初期化処理

- ・ Winnyp.exeへのパッチ処理では、約200箇所を書き換え処理を実行
- ・ 書き換え処理の大半は、参照する文字列の変更
例) Noderef.txt → Noderefp.txt など

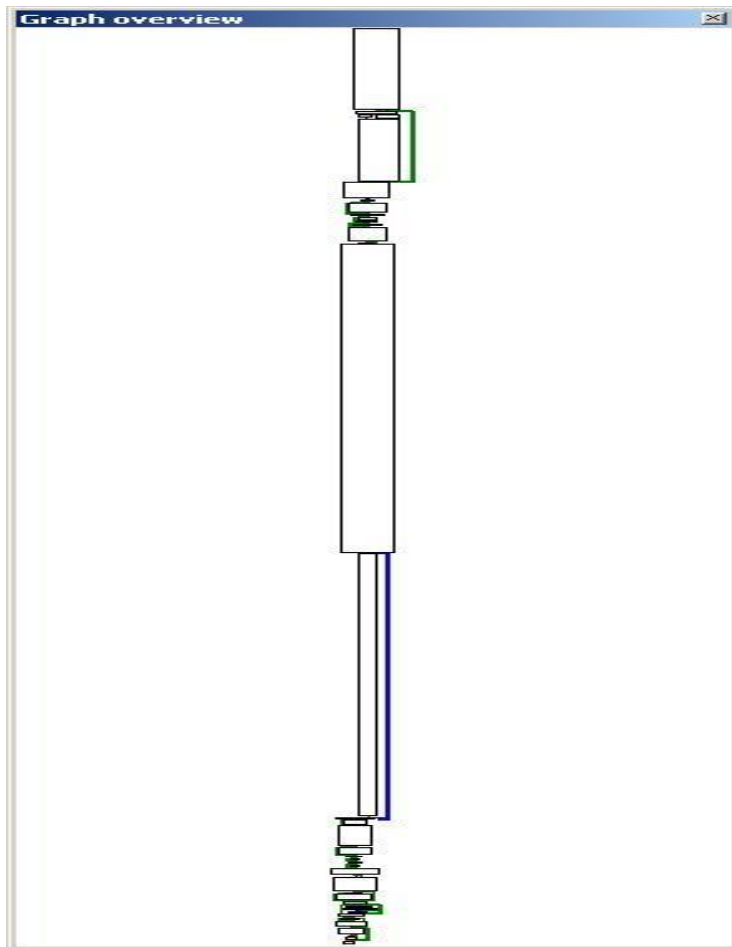


Winnypの暗号鍵生成処理

- ・ Winnyでは、暗号鍵生成処理が簡単だったため(RC4の初期化处理)、解析を行うことで簡単に暗号鍵生成処理を解明することができた
- ・ Winnypでは、パケットの暗号アルゴリズムとしてRC4を採用しているが、暗号鍵生成処理に複数の暗号アルゴリズムを使用し解析が困難になっている

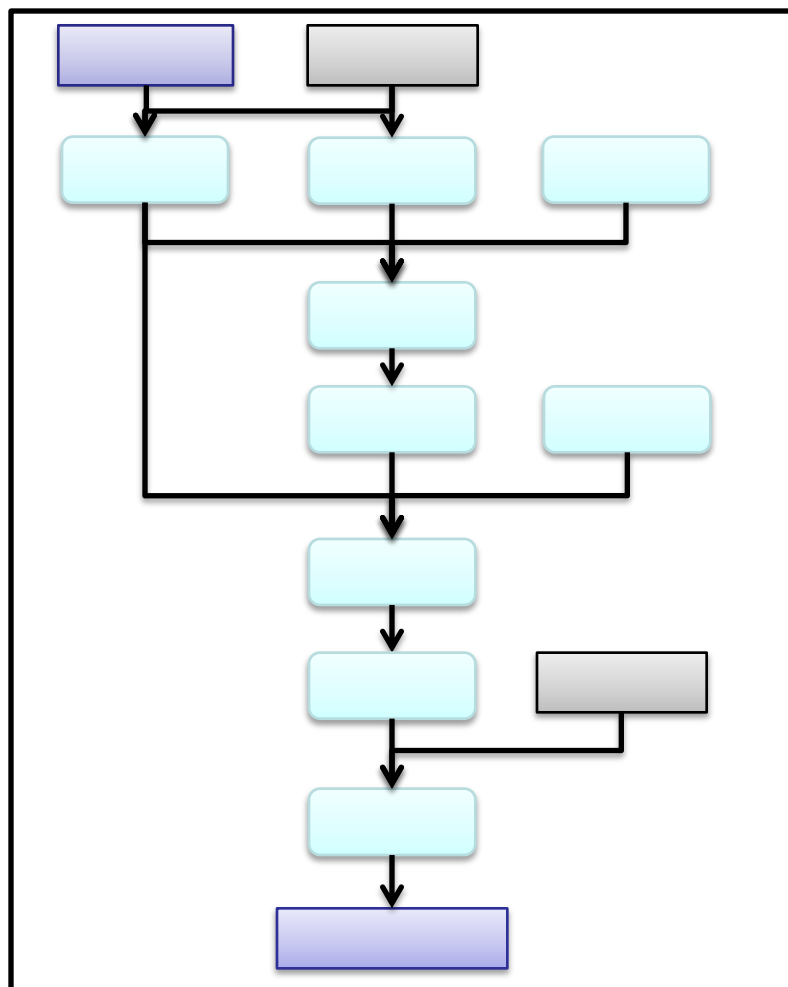


Winnypの暗号鍵生成処理 — 全体像 —

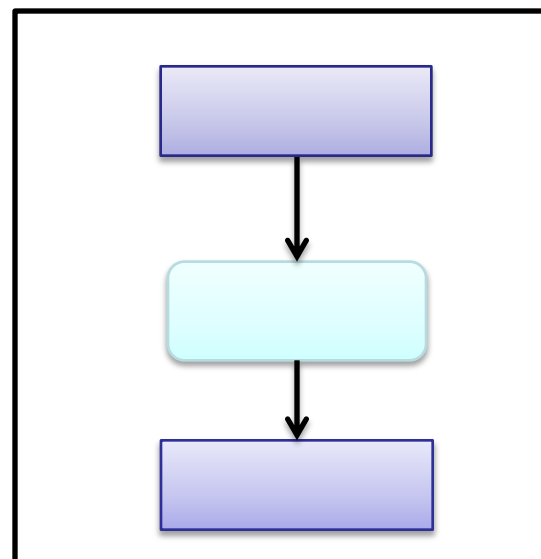


- ・ Winnypの暗号鍵生成処理メインルーチン
- ・ 長い処理が複数続いていて複雑になっている

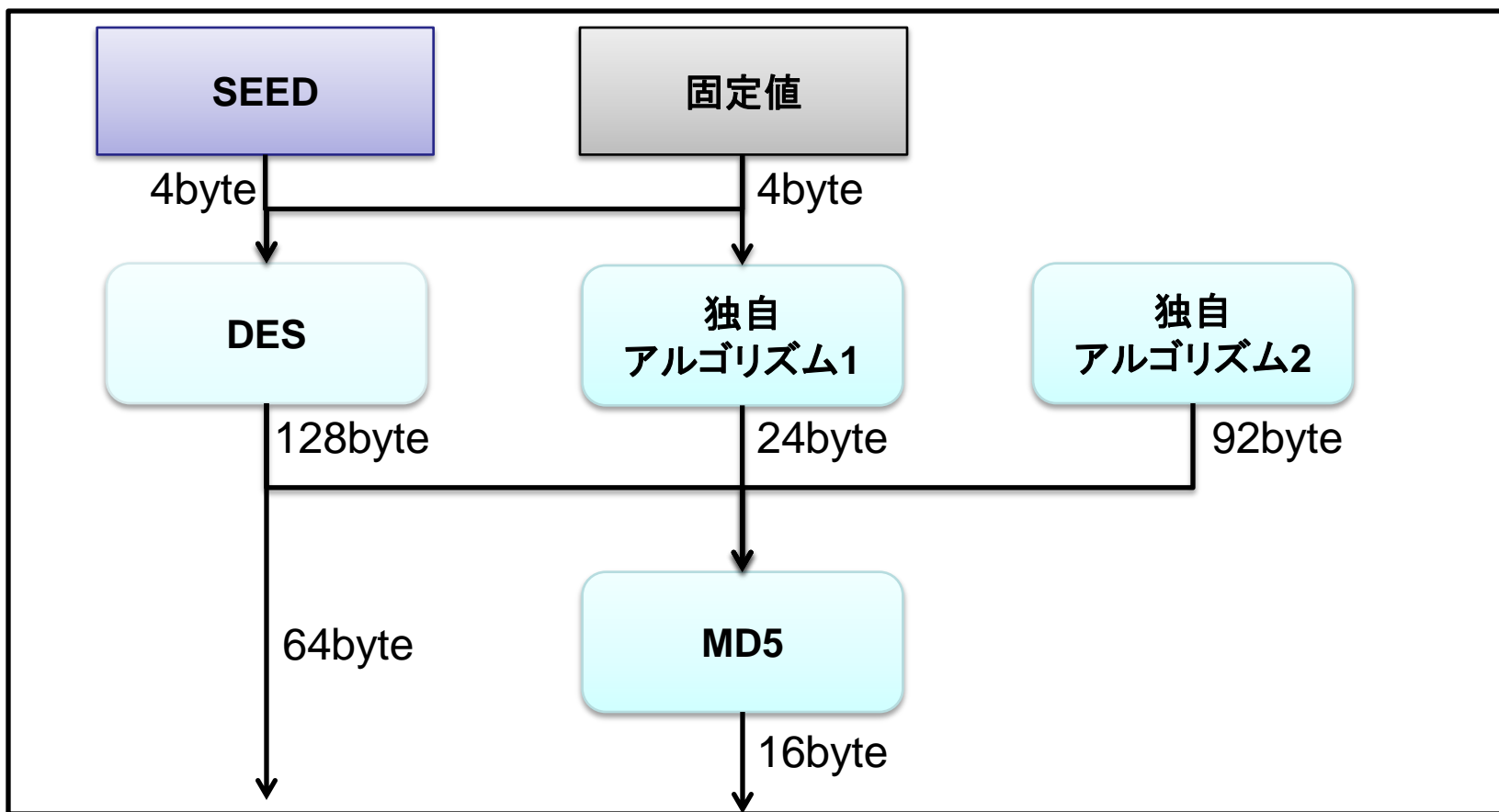
Winnypの暗号鍵生成処理 — 全体像 —



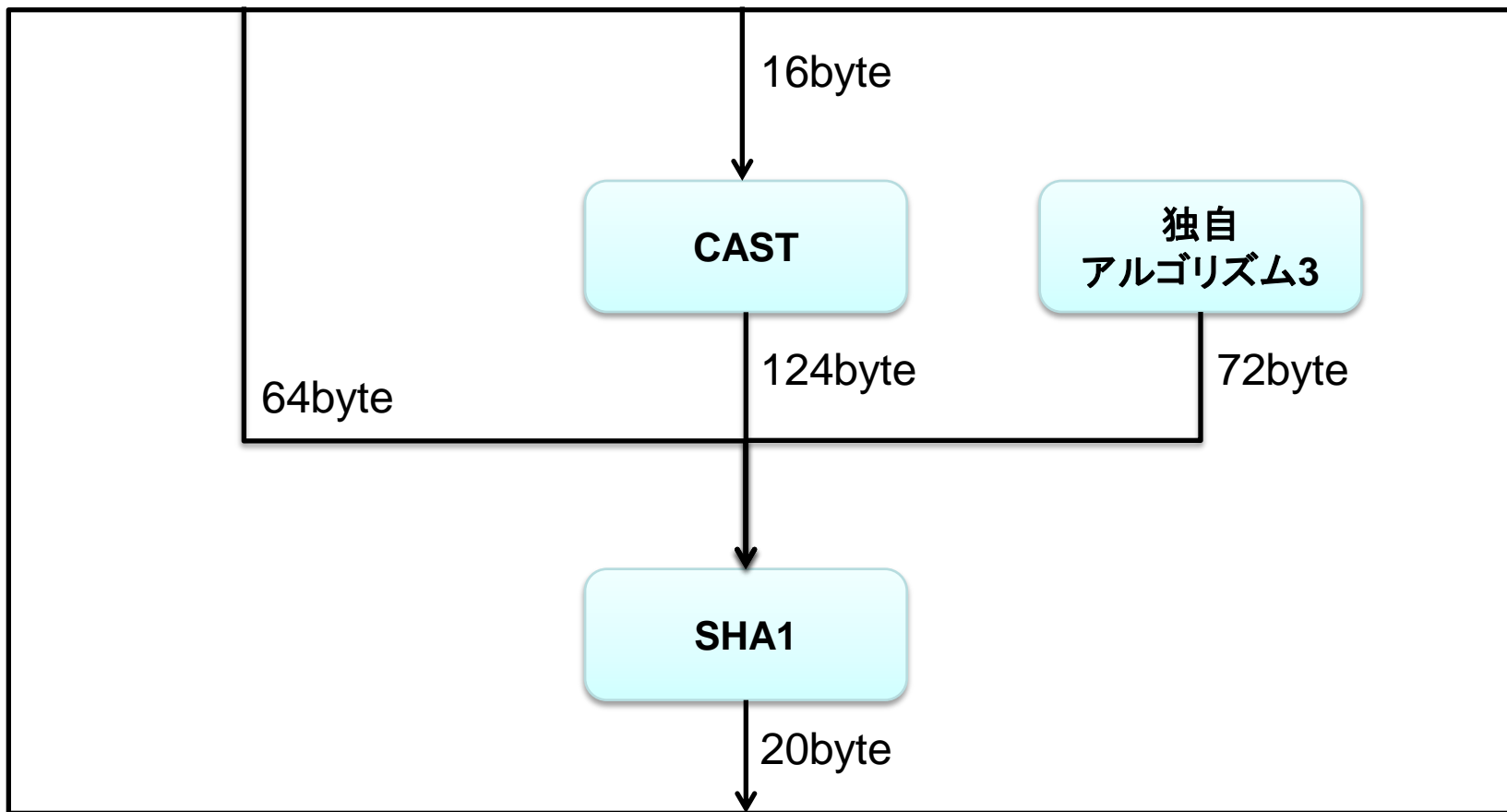
- ・ Winnypの暗号鍵生成処理を処理ブロックごとに流れを図式化
- ・ Winnyの場合の暗号鍵生成処理に比べて複雑になっている



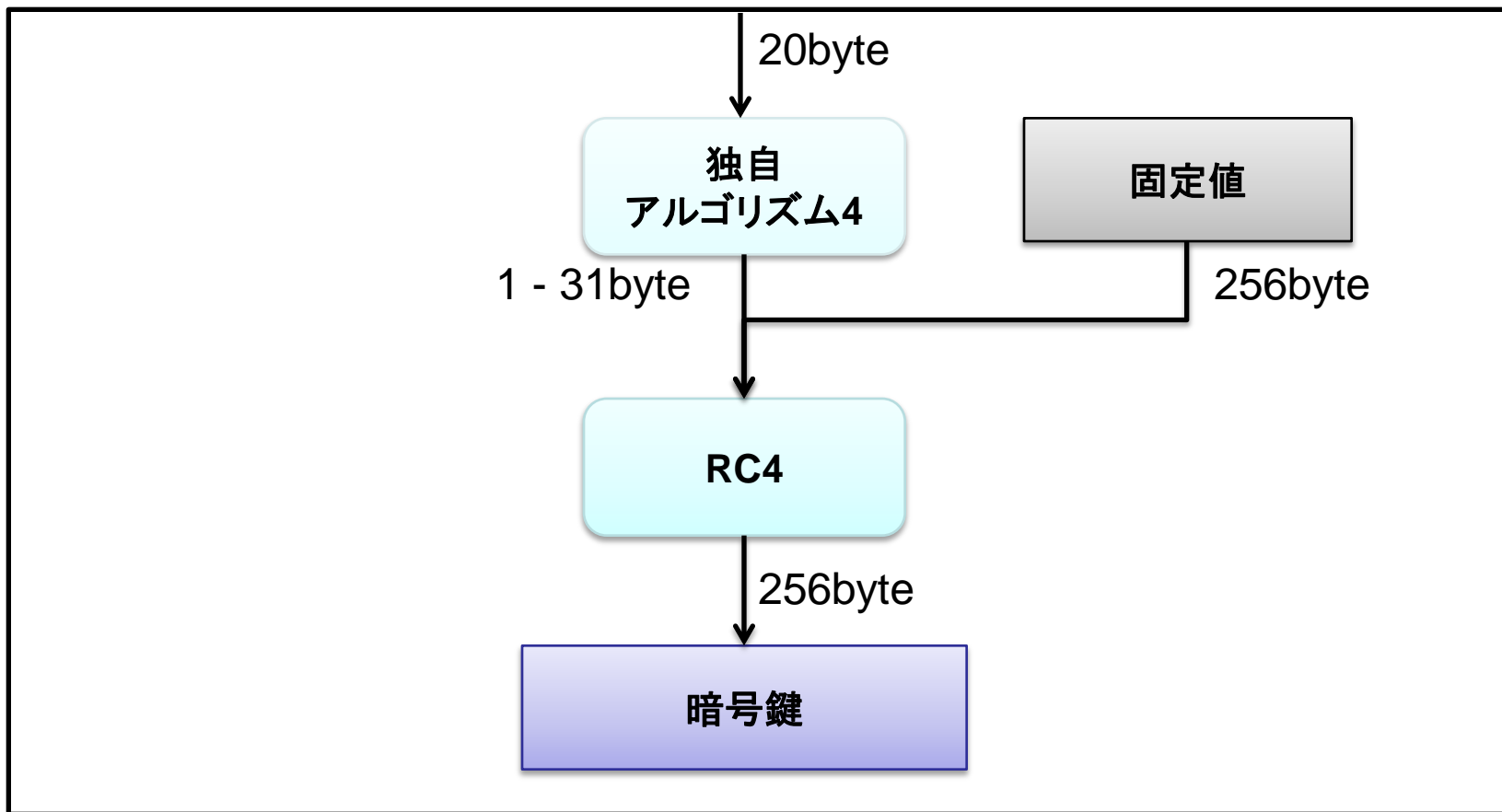
Winnypの暗号鍵生成処理 — Stage 1 —



Winnypの暗号鍵生成処理 — Stage 2 —

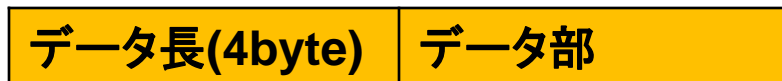


Winnypの暗号鍵生成処理 — Stage 3 —



Winnypの packets 送信処理

- ・ Winnypの packets 送信処理では、初期化時に生成したデータを使用して Winny packets の後にダミーデータを付加する
- ・ 暗号鍵の生成アルゴリズムは、Winnypの設定ファイルにより選択される (Winny用なのかWinnyp用なのか)
- ・ ダミーデータを付加するのは、接続確立, 切断時に送信されるいくつかの コマンド packets のみ





Winnyの packets 送信処理

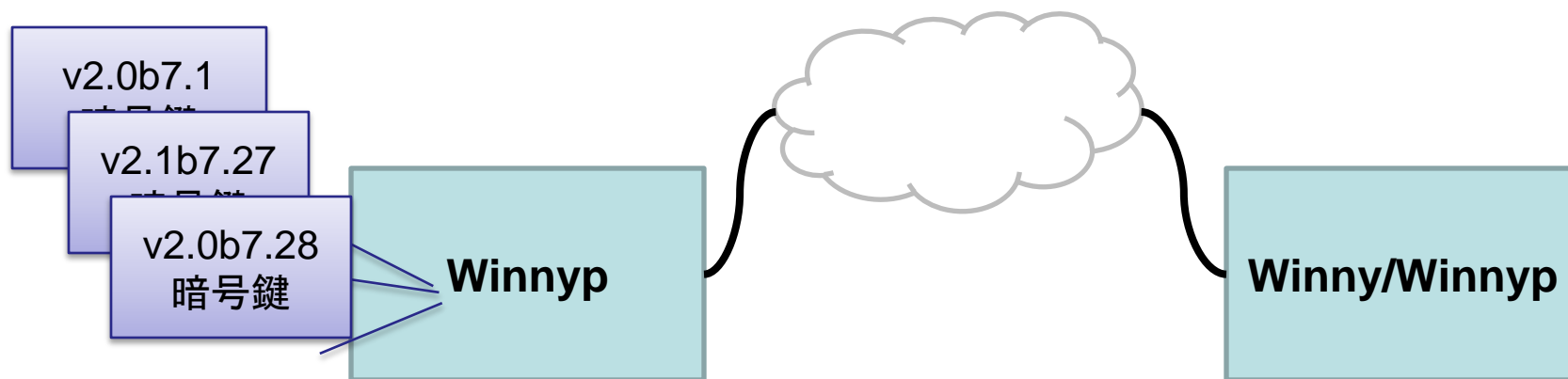
- Winnyのダミーデータが付加された初期 packets

```

0030 fd b8 92 44 00 00 f7 90 ea b4 a4 17 4e b6 6a 12 ...D... ..N,j.
0040 34 7e eb 1f 35 70 38 cd e8 74 d6 4b 48 22 13 37 4~..5p8. .t.KH".7
0050 8b 5a 2c 8a 3a 06 08 6c 4a 89 8a 81 3b 7e 9a 44 .Z,..:..l J...;~.D
0060 74 c2 65 a6 6f 31 50 f9 24 81 ef 58 29 ff c6 be t.e.o1P. $.X)...
0070 5e 92 ba 8f 13 cc df 86 95 9c 8d fa 19 df 93 f7 ^.....
0080 35 1d 7d 67 ee f3 02 49 0a bf 25 a2 78 6a 7d 4f 5.}g...I ...%.xj}O
0090 20 0f 78 36 47 af 64 a9 e2 de fc 90 7d e1 37 6e .x6G.d. ....}.7n
00a0 3b 42 20 59 24 67 bf 21 d9 e8 26 e6 ad b0 96 93 ;B Y$g.! ..&.....
00b0 08 1e ca cd cf 08 21 28 09 5e ac e6 4d 60 6b 1c .....!( ^..M`k.
00c0 c2 78 e1 e2 bc 95 32 1c 02 3c b5 01 ee 1d b8 8a .x....2. <.....
00d0 b5 07 fb 3f 93 c4 3c 79 0b c2 a7 56 94 a6 9d 51 ...?...<y ...V...Q
00e0 c4 8b a3 f8 f6 44 a8 14 14 df 12 54 ac 47 24 99 .....D... ..T.G$.
00f0 0a e1 3e f7 4d b8 cb 0b ae 00 09 18 db 25 3b c2 ...>.M... ..%;.
0100 5c 4d de 6b d0 01 00 3f 3e 6d 0d 51 89 2a c0 c2 \M.k...? >m.Q.*..
0110 13 44 4d 36 73 f9 a4 45 68 bf ad 42 04 42 51 97 .DM6s..E h..B.BQ.
0120 cd 11 8a 84 6b 2b 43 39 76 a8 e0 3d e1 3a 67 00 ....k+C9 v...=:g.
0130 59 39 35 ed 88 84 e1 ec 2a 18 9b f4 38 a7 b4 31 Y95..... *.8..1
0140 5d 98 30 35 a3 fb 90 b7 12 77 75 93 f6 20 8e 6f ].05.... .wu... o
0150 96 97 79 14 f4 a9 df 09 f1 f5 66 77 f4 15 55 6c ..y..... ..fw..U]
0160 26 e3 b3 12 b0 33 43 0a 10 b0 22 8f b9 ba 2c 26 &....3C. ...".&
0170 f2 4e 42 aa 38 95 49 07 f3 d6 72 f8 db e3 5c b4 .NB.8.I. ..r...\.
0180 f1 98 73 fc c6 ea cf 61 ec 33 7d 31 69 85 2f ac .s....a .3}1i./
0190 50 6f 27 ec 3a 36 10 b8 bb 95 fa e6 ea 80 c8 6b Po'.:6.. ....k
01a0 f6 51 24 9f a7 f6 65 06 6b 61 14 84 3a 25 9b ec .Q$...e. ka...%..
01b0 8f 30 4f f1 d3 5e a1 4b cd ac a0 cc af c3 90 49 .00..^,K .....I
01c0 e3 24 86 07 02 1d 36 e7 cb 8c dd 8a ee a0 a9 9b .$....6. ....
01d0 8f 06 24 17 b3 ea b0 66 f1 39 0f b2 6c 30 85 0a ..$....f .9..70..
01e0 99 d8 80 bd 4f bd 92 eb cb ee 5e a5 93 f5 67 04 ....O... ..^....g.
01f0 0a 89 b3 c1 e7 5f 04 23 5c b1 8d aa db 02 25 45 .....# \.....%E
0200 f6 34 a2 e9 4d a5 bf 69 af 30 8d e9 89 27 9d e6 .4..M..i .0...'.
0210 a1 03 ab b9 74 3b 8d 6e ba 08 9c 77 3a d1 3d d9 ....t;:n ...w:=.
    
```

Winnypの packets 受信処理

- Winnypの packets 受信処理では、初期 packets 受信時に、3つの暗号鍵を生成する (Winny v2.0b7.1, Winnyp v2.1b7.27, Winnyp v2.1b7.28)
- それぞれの暗号鍵を使用して接続ノードのバージョンを特定
- これらの処理により、複数バージョンのノードとの接続が可能



Winnypの接続ノード特定処理

- ・ 特定処理では、受信パケットの先頭5byteを復号する
- ・ 復号したデータに対していくつかのチェックを行う

チェック1	長さの部分が0以上
チェック2	長さの部分が131,072以下
チェック3	長さの部分+4が受信パケット長以下
チェック4	コマンド番号が100以下

- ・ チェックがすべて成功した場合、接続ノードの特定ができたと判定する
- ・ 失敗した場合は、別の暗号鍵を使用して再度復号してチェックする



```
int packet_analysis(GDDCONFIG *gddc, unsigned char *packet, unsigned long length)
```

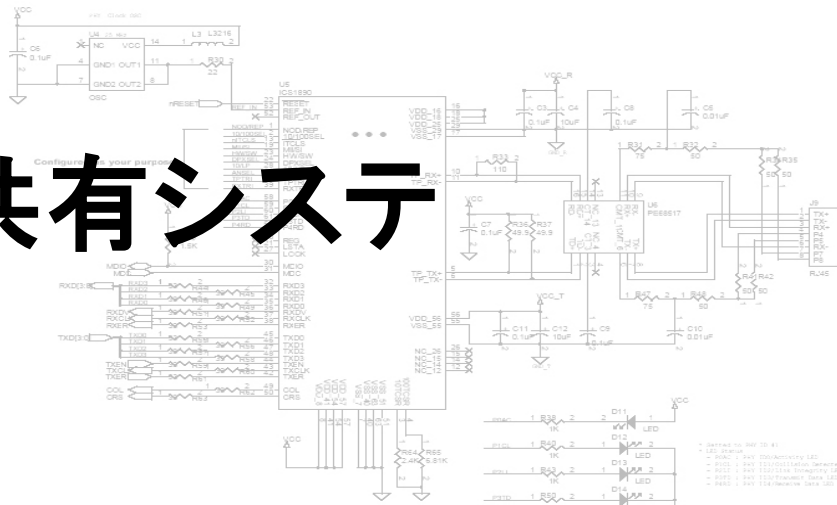
```
struct ip      *ip_header; /* IP header */
struct tcphdr *tcp_header; /* TCP header */
char          *tcp_data; /* TCP data */
struct in_addr addr; /* IP address */
char          sourceIP[16]; /* Source IP address */
char          destIP[16]; /* Destination IP address */
unsigned short sourcePort; /* Source Port */
unsigned short destPort; /* Destination Port */
...
/* Log type */
CONN_LIST *bcl;
static char dates[10];
time_t time;
struct tm *timep=NULL;
char *timesp=NULL;
char *c;
```

2. 匿名P2Pファイル共有システムの静的解析

```
/* Get pointer of IP header and check length of IP */
if (length-SIZE_OF_ETHHDR < MINSIZE_IP+MINSIZE_TCP) return(0);
ip_header = (struct ip *) (packet+SIZE_OF_ETHHDR);
if (ip_header->ip_p!=IPPROTO_TCP)
|| ip_header->ip_v!=4) return(0);
iph_len = ((unsigned long) (ip_header->ip_hl))*4;
if (iph_len<MINSIZE_IP) return(0);
if ((unsigned long)ntohs(ip_header->ip_len) < MINSIZE_IP+MINSIZE_TCP)
return(0);
if ((unsigned long)ntohs(ip_header->ip_len) > length-SIZE_OF_ETHHDR) {
return(0);
}
```

```
/* Get pointer of TCP header and check length of TCP */
tcp_header = (struct tcphdr *) ((char *) ip_header+iph_len);
tcp_len = ((unsigned long) (tcp_header->th_off))*4;
tcp_data = (char *) tcp_header+tcp_len;
if (tcp_len<MINSIZE_TCP) return(0);
```

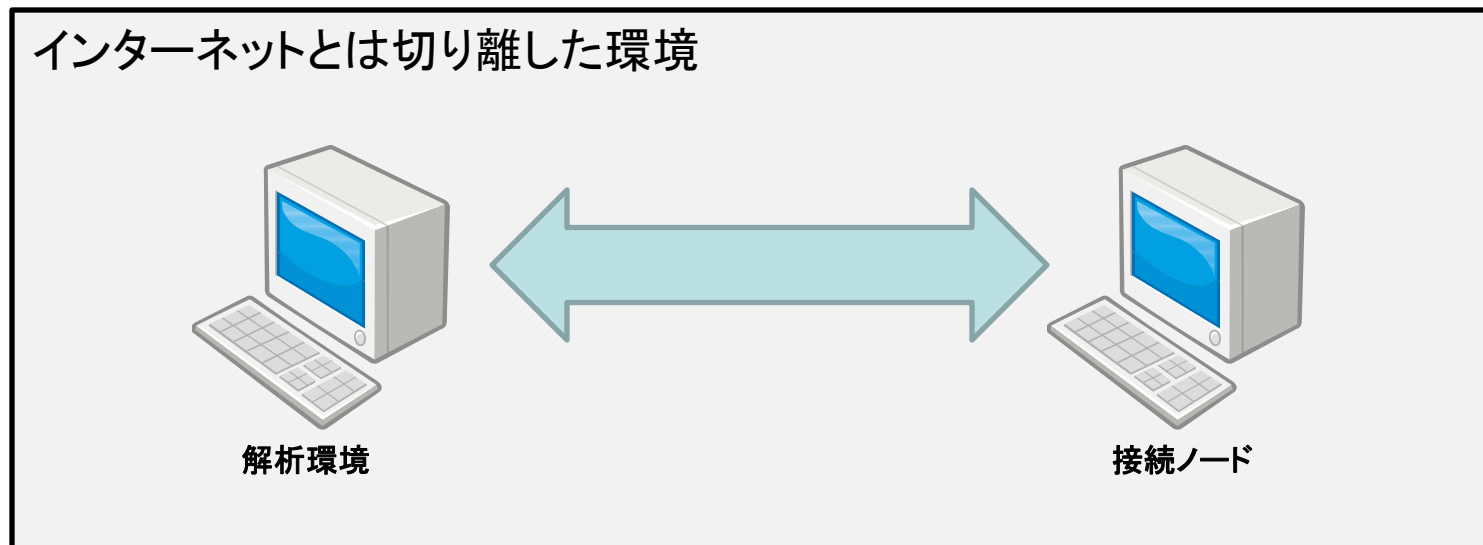
```
/* Get other parameter in TCP/IP header */
if (((long)ntohs(ip_header->ip_len)-(long)iph_len-(long)tcp_len)<0)
return(0);
len_data = ((unsigned long)ntohs(ip_header->ip_len)
-iph_len-tcp_len);
sourcePort = ntohs(tcp_header->th_sport);
destPort = ntohs(tcp_header->th_dport);
memcpy(&addr, &(ip_header->ip_sro), sizeof(struct in_addr));
strcpy(sourceIP, (char *) inet_ntoa(addr));
memcpy(&addr, &(ip_header->ip_dst), sizeof(struct in_addr));
strcpy(destIP, (char *) inet_ntoa(addr));
if (!strcmp(sourceIP, destIP)) return(0);
```



00001B70	FF 15 F0 11 00 01 E9 C0 03 00 00 E8 70 1C 00 00	...
00001B80	33 F6 56 E8 B1 FC FF FF 85 C0 0F 84 B7 03 00 00	3 * 離 ...
00001B90	56 6A 02 FF 35 6C 80 00 01 FF 35 D0 87 00 01 FF	Vj . 51 ...
00001BA0	15 CC 11 00 01 85 C0 75 D 68 10 10 00 00 FF 35	. 7 ... u . h ...
00001BB0	50 80 00 01 FF 35 44 80 00 01 FF 35 D0 87 00 01	P ... 5D ...
00001BC0	FF 15 04 12 00 01 FF 35 D0 87 00 01 FF 15 2C 12	... 53 ...
00001BD0	00 01 FF 35 D4 88 00 01 FF 15 58 10 00 01 E9 64	... 5t ... X ...
00001BE0	03 00 00 83 FE 1A 77 47 0F 84 59 03 00 00 83 FE	... wG . 山 ...
00001BF0	11 0F 85 16 01 00 00 33 F6 39 35 E8 87 00 01 74	... 3.95 ...
00001C00	22 88 3D 28 12 00 01 56 FF D7 56 FF D7 68 00 10	... (= ... V . 5V . 5h ...
00001C10	00 00 FF 35 50 80 00 01 FF 35 88 80 00 01 E9 7D	... 5P ... 5 ...
00001C20	02 00 00 6A 01 E8 0F FC FF E9 1A 03 00 00 88	... j ...
00001C30	7D 14 88 11 01 00 00 3B F0 0F 87 88 00 00 00 3B	... 夕 ...
00001C40	F0 0F 84 16 02 00 00 83 FE 1C 0F 85 B0 00 00 00	...
00001C50	33 F6 39 75 10 74 2F A1 EC 87 00 01 88 0D F0 87	3.9u.t/ ...
00001C60	00 01 3B C6 75 08 3B CE 0F 84 D9 02 00 00 8B 3D	...
00001C70	14 12 00 01 51 50 68 B1 00 00 00 FF 35 D4 87 00	...
00001C80	01 E9 56 01 00 00 8B 3D 14 12 00 01 68 F0 87 00	...
00001C90	01 68 EC 87 00 01 68 B0 00 00 00 FF 35 D4 87 00	...
00001CA0	01 FF D7 A1 EC 87 00 01 88 0D F0 87 00 01 3B C1	...
00001CB0	75 11 89 35 EC 87 00 01 89 35 F0 87 00 01 E9 84	...
00001CC0	02 00 00 51 50 68 07 01 00 00 8B CE B8 12 01 00	...
00001CD0	00 2B C8 0F 84 3C 02 00 00 83 E9 04 0F 84 29 02	...
00001CE0	00 00 49 0F 84 F9 01 00 00 81 E9 1C 01 00 00 0F	...
00001CF0	84 E0 01 00 00 81 E9 E6 00 00 00 0F 84 3D 01 00	...
00001D00	00 81 E9 E8 7C 00 00 0F 84 02 01 00 00 3B 35 5C	...
00001D10	88 00 01 0F 85 EE 00 00 88 45 14 8B 48 0C 8B	...
00001D20	C1 8B D1 F7 D0 C1 EA 02 83 E0 01 83 E2 01 F6 C1	...

解析環境の構築

- ・ 通常のネットワークにP2Pアプリケーションを接続した場合、コネクションが多く、解析が困難
- ・ そのため、1対1で通信を行うような環境を構築



デバッグ対策、難読化の回避

- ・ 匿名P2Pアプリケーションには、匿名性を高めるためデバッグ対策や難読化が施されている
- ・ これらを回避するため、デバッグで起動させるのではなく、P2Pアプリケーション起動後にデバッグでアタッチさせる
- ・ 通信処理を解析する場合は、初期化処理の解析はある程度飛ばしても問題ないため、この方法での解析が可能



通信処理の解析

- ・ 実行ファイルがIDA Proで読み込むことができないので、通信処理を特定することが困難
- ・ APIに対してブレークポイントを設定し、スタックをトレースすることで、通信処理を行っている個所を特定する
- ・ 同様の方法で、ファイルアクセス箇所等の特定も可能

暗号アルゴリズムの推測

- ・ 暗号アルゴリズムのアセンブリコードを解析しても、アルゴリズムの特定は困難
- ・ 暗号アルゴリズム内で使用されている特定の数値を用いてコードサーチエンジンを使用

```

arg_0      = dword ptr 4
mov     eax, [esp+arg_0]
xor     ecx, ecx
mov     dword ptr [eax], 67452301h
mov     dword ptr [eax+4], 0EFCDAB89h
mov     dword ptr [eax+8], 98BADCFEh
mov     dword ptr [eax+0Ch], 10325478h
    
```

Google Labs
ソースコード検索オプション

検索

ソースコード

同一の結果 11 件

```

206: /* SHA1 initialization constants */
context->state[0] = 0x67452301;
context->state[1] = 0xEFCDAB89;
context->state[2] = 0x98BADCFE;
    
```

をさらに

同一の結果 4 件

```

231: {
    this->state[0] = 0x67452301;
    this->state[1] = 0xEFCDAB89;
    this->state[2] = 0x98BADCFE;
    
```

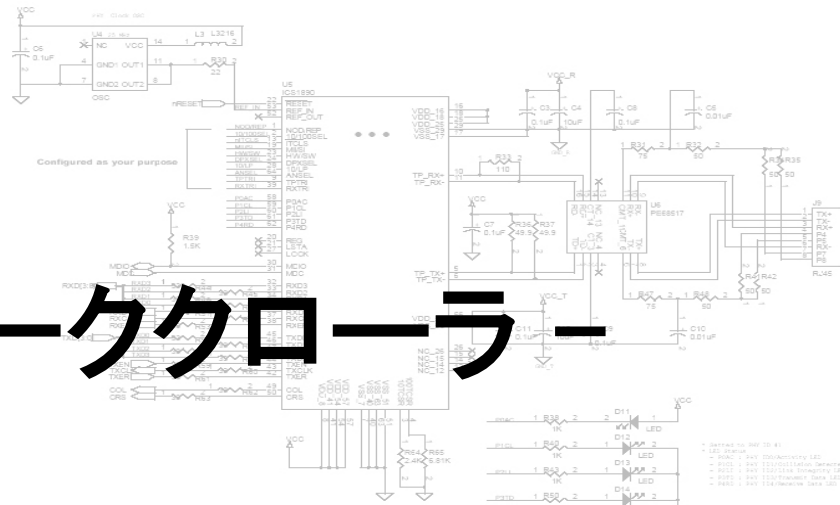



```
int packet_analysis(GDDCONFIG *gddc, unsigned char *packet, unsigned long length)
```

```
struct ip      *ip_header; /* IP header */
struct tcphdr *tcp_header; /* TCP header */
char          *tcp_data; /* TCP data */
struct in_addr addr; /* IP address */
char          sourceIP[16]; /* Source IP address */
char          destIP[16]; /* Destination IP address */
unsigned short sourcePort; /* Source Port */
unsigned short destPort; /* Destination Port */
...
/* Get pointer of IP header and check length of IP */
if (length-SIZE_OF_ETHHDR < MINSIZE_IP+MINSIZE_TCP) return(0);
ip_header = (struct ip *) (packet+SIZE_OF_ETHHDR);
if (ip_header->ip_p!=IPPROTO_TCP)
|| ip_header->ip_v!=4) return(0);
iph_len = ((unsigned long)(ip_header->ip_hl))*4;
if (iph_len<MINSIZE_IP) return(0);
if ((unsigned long)ntohs(ip_header->ip_len) < MINSIZE_IP+MINSIZE_TCP)
return(0);
if ((unsigned long)ntohs(ip_header->ip_len) > length-SIZE_OF_ETHHDR) {
return(0);
}
/* Get pointer of TCP header and check length of TCP */
tcp_header = (struct tcphdr *) ((char *) ip_header+iph_len);
tcp_len = ((unsigned long)(tcp_header->th_off))*4;
tcp_data = (char *) tcp_header+tcp_len;
if (tcp_len<MINSIZE_TCP) return(0);
/* Get other parameter in TCP/IP header */
if (((long)ntohs(ip_header->ip_len)-(long)iph_len-(long)tcp_len)<0)
return(0);
len_data = (unsigned long)ntohs(ip_header->ip_len)
-iph_len-tcp_len;
sourcePort = ntohs(tcp_header->th_sport);
destPort = ntohs(tcp_header->th_dport);
memcpy(&addr, &(ip_header->ip_sro), sizeof(struct in_addr));
strcpy(sourceIP, (char *) inet_ntoa(addr));
memcpy(&addr, &(ip_header->ip_dst), sizeof(struct in_addr));
strcpy(destIP, (char *) inet_ntoa(addr));
if (!strcmp(sourceIP, destIP)) return(0);
```

3. WinnypRadar

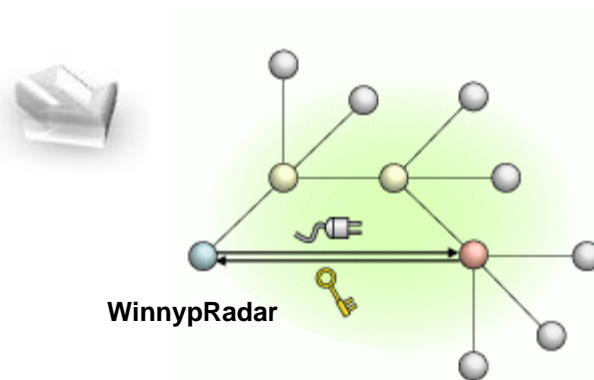
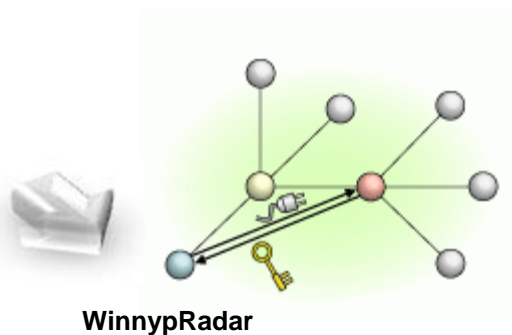
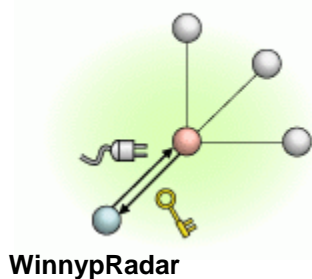
- Winnypネットワーククローラ -



00001B70	FF 15 F0 11 00 01 E9 C0 03 00 00 E8 70 1C 00 00	銀...開...
00001B80	33 F6 56 E8 B1 FC FF FF 85 C0 0F 84 B7 03 00 00	3*誰...+...
00001B90	56 6A 02 FF 35 6C 80 00 01 FF 35 D0 87 00 01 FF	Vj...51...5...
00001BA0	15 CC 11 00 01 85 C0 75 D 68 10 10 00 0F FF 35	7...+u.h...5
00001BB0	50 80 00 01 FF 35 44 80 00 01 FF 35 D0 87 00 01	P...5D...5...
00001BC0	FF 15 04 12 00 01 FF 35 D0 87 00 01 FF 15 2C 12	...5...
00001BD0	00 01 FF 35 D4 88 00 01 FF 15 58 10 00 01 E9 64	...5t...X...誠
00001BE0	03 00 00 83 FE 1A 77 47 0F 84 59 03 00 00 83 FE	...wg.U...誠
00001BF0	11 0F 85 16 01 00 00 33 F6 39 35 E8 87 00 01 74	...3.95開...t
00001C00	22 88 3D 28 12 00 01 56 FF D7 56 FF D7 68 00 10	...(.V.V.5h...
00001C10	00 00 FF 35 50 80 00 01 FF 35 88 80 00 01 E9 7D	...5P...5*...暮
00001C20	02 00 00 6A 01 E8 0F FC FF E9 1A 03 00 00 88	...j...急
00001C30	70 14 88 11 01 00 00 3B F0 0F 87 88 00 00 00 3B	...ク...衛...
00001C40	F0 0F 84 16 02 00 00 83 FE 1C 0F 85 B0 00 00 00	...
00001C50	33 F6 39 75 10 74 2F A1 EC 87 00 01 88 0D F0 87	3.9u.t/...*
00001C60	00 01 3B C6 75 08 3B CE 0F 84 D9 02 00 00 8B 3D	...u.u.株...=
00001C70	14 12 00 01 51 50 68 B1 00 00 00 FF 35 D4 87 00	...QPh7...5t...
00001C80	01 E9 56 01 00 00 88 3D 14 12 00 01 68 F0 87 00	...儲...h...
00001C90	01 68 EC 87 00 01 68 B0 00 00 00 FF 35 D4 87 00	...h...h...5t...
00001CA0	01 FF D7 A1 EC 87 00 01 88 0D F0 87 00 01 3B C1	...う...*...:付
00001CB0	75 11 89 35 EC 87 00 01 89 35 F0 87 00 01 E9 84	u...5*...5*...験
00001CC0	02 00 00 51 50 E9 07 01 00 00 8B CE B8 12 01 00	...QP...動...
00001CD0	00 2B C8 0F 84 3C 02 00 00 83 E9 04 0F 84 29 02	...+<...<...)
00001CE0	00 00 49 0F 84 F9 01 00 00 81 E9 1C 01 00 00 0F	...l...*
00001CF0	84 E0 01 00 00 81 E9 E6 00 00 00 0F 84 3D 01 00	...*...=...
00001D00	00 81 E9 E8 7C 00 00 0F 84 02 01 00 00 3B 35 5C	*開...:5t...
00001D10	88 00 01 0F 85 EE 00 00 88 45 14 8B 48 0C 8B銀稀驚
00001D20	C1 8B D1 F7 D0 C1 EA 02 83 E0 01 83 E2 01 F6 C1	鏡*子...*

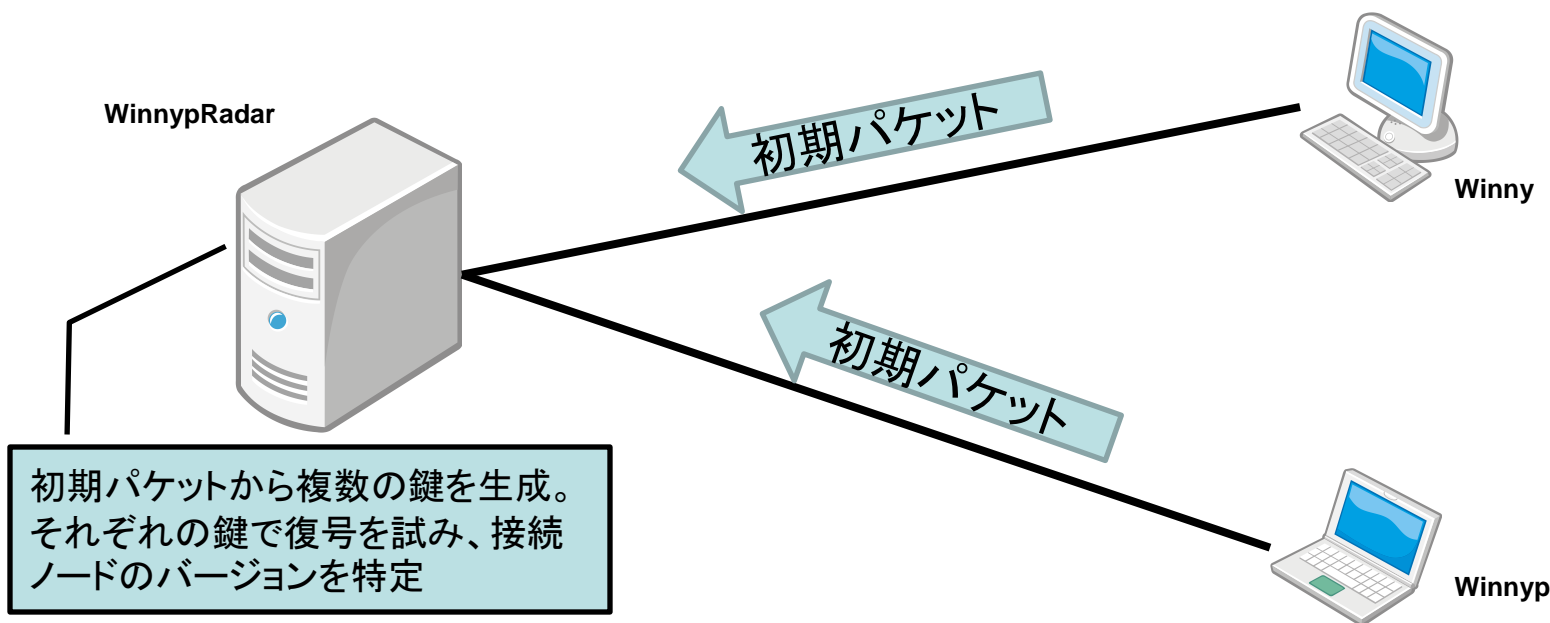
WinnypRadar

- ・ Winnypノードのひとつとして、Winnypネットワークに接続
- ・ Winnypプロトコルを使用して、接続したノードからキー情報を収集
- ・ 本クローラでは、Winnyノード/Winnypノードの両方に接続することが可能



WinnyノードとWinnypノードの判別

- ・ Winnypネットワークには、Winnyとの互換性があるため、Winnyノードが含まれている可能性がある
- ・ WinnypRadarでは、接続したノードの初期パケットを元に接続ノードのバージョンを特定している



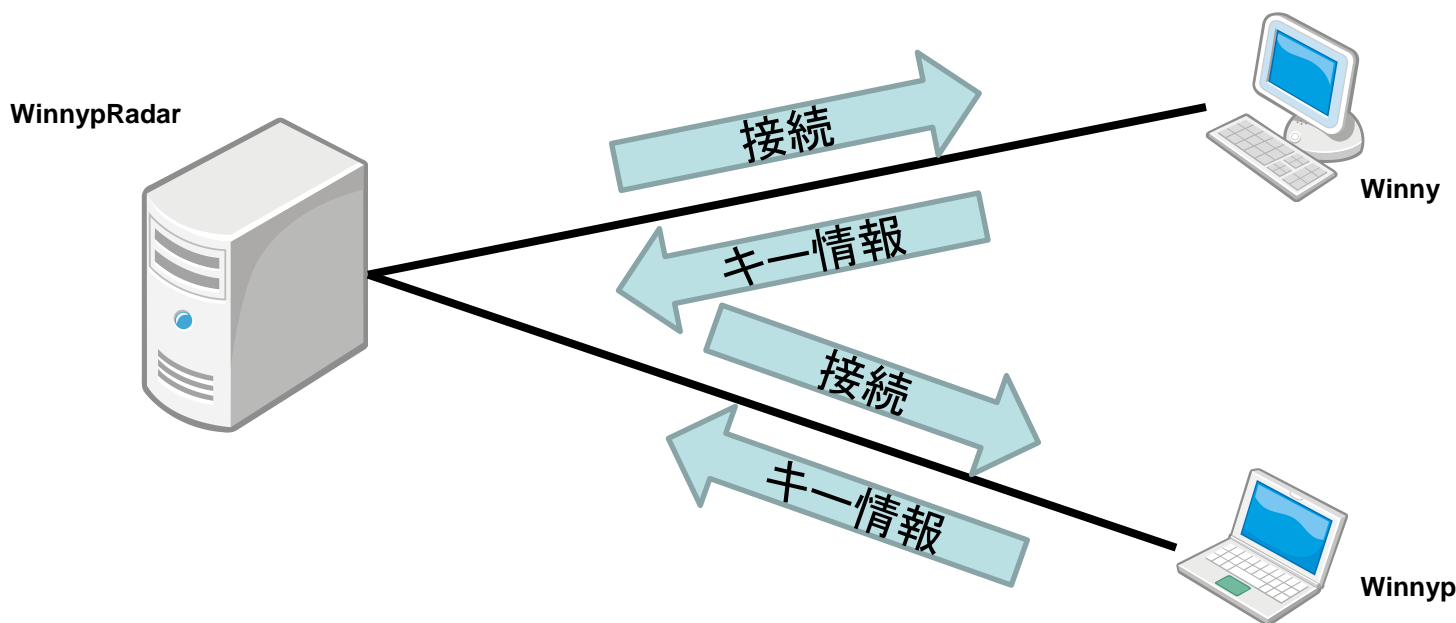
収集情報

- ・ 収集したキー情報には、公開しているファイルと公開元のIPアドレスが含まれる
- ・ これらの情報を収集し、どのIPアドレスがどんなファイルを公開しているかを調査することが可能

IPアドレス
ポート番号
ファイルサイズ(バイト数)
ファイルタイムスタンプ
ファイル名
ハッシュ
...

クローリング

- ・ キー情報の中から、公開元のIPアドレスを取得し、新しい接続先とする
- ・ 新しい接続先に接続し、キー情報を取得する
- ・ これらの動作を繰り返し、ネットワーク内をクローリングする



観測結果

- ・ Winnyネットワーク内のWinnypノードの割合
 - Winnypクローラが接続する際に、接続ノードがWinnypかどうかを判定し、記録
 - 全接続ノード数からWinnypの割合を算出
- ・ WinnypRadarを使用し、1日間計測した結果

ノード数	19. 8万ノード (Port0を除く)
Winnypノードの割合	8% (1. 6万ノード)

※P2P研究会、クロスワープ社の調査結果より

http://www.scat.or.jp/stnf/contents/p2p/p2p080910_4.pdf

まとめ

- ・ 今回Winnyp 2.1b7.28の暗号鍵生成処理とパケット送受信処理についての解析を行った
- ・ Winnypでは、解析を困難にさせるため暗号鍵生成処理が複雑となっている
- ・ 解析結果をもとにWinnypネットワーククローラ WinnypRadarを開発
- ・ WinnypRadarを使用することにより、今まで検出できなかったWinnypノードに関する調査が可能になった

今後の課題

- ・ 今回、十分なノード調査期間が取れなかったため、長期的にノード調査を行う必要がある
- ・ 今回の調査では、Winnyネットワークに接続可能なWinnypノードを調査したが、Winnypのみでの接続を行うノードの調査を行うことでWinnypノードのより正確な数が計測できると思われる

ありがとうございました



Fourteenforty Research Institute, Inc.

株式会社 フォティーンフォティ技術研究所

<http://www.fourteenforty.jp>

シニアソフトウェアエンジニア 石山智祥

ishiyama@fourteenforty.jp