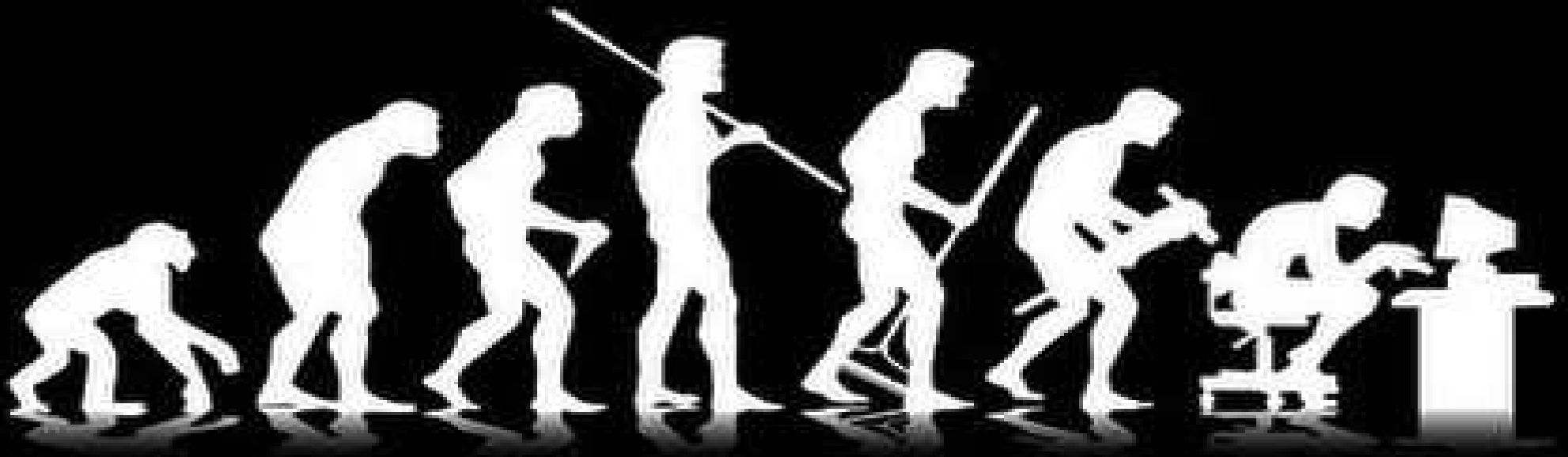
종의 기원



생존 경쟁에 있어서 유리한 종족의 보존에 대하여

발표자 소개

- 성명 : 조효제 (29세), 전문연구요원
- 닉네임 : peterpan, KIMKAPSOO (임시)
- 소속 : (주)하우리
- 관심 분야 : 러블리즈 케이, 구구단 세정
- 주 연구분야 : 악성코드분석 외길인생

- Blog : sfkino.tistory.com
- Facebook : https://www.facebook.com/hyoje.jo.3

- 꿈 : 꿈은 없고 그냥 놀고 싶습니다.
- 하고 싶은 말 : 2D는 좋아하지만 오덕은 아닙니다.

이 발표에서는..

- 발표계기
  - 최종 행위적 관점에서의 분석내용은 많으나 치열한 악성코드 내부 상황을 기술한 발표가 많지 않음
  - 악성코드발전을 위해 노력한 제작자들의 **노고를 치하**하기 위함

- 이 발표에서 다루어 지는 내용
  - 작성자가 경험한 악성코드들의 자가보호 기법
  - 얕디 얕은 진화론의 일부

- 이 발표에서 다루지 않는 내용
  - 악성코드 분석론
  - 악성코드의 최종 행위

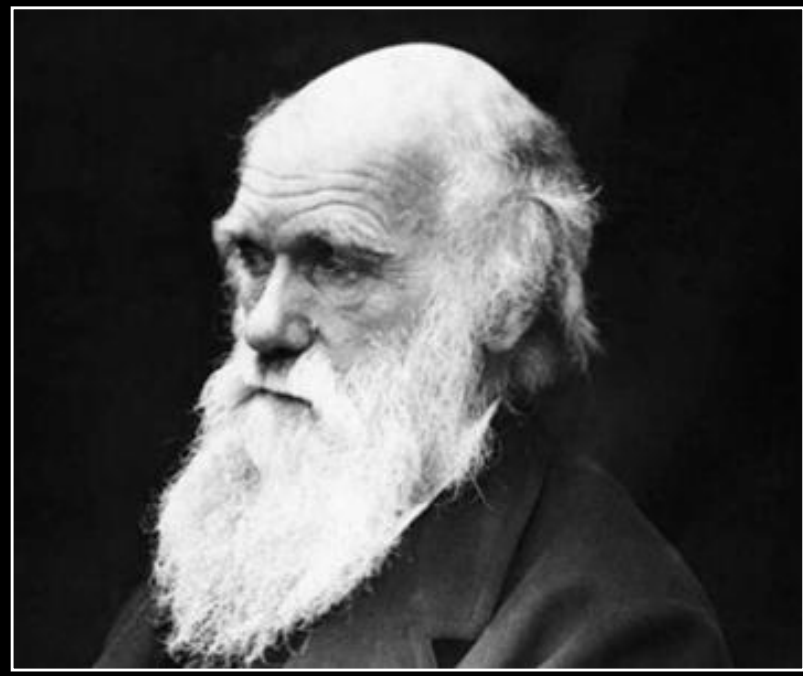
종의 기원 (On the Origin of Species)

생존 경쟁에 있어서 유리한 종족의 보존에 대하여

1859년 11월 24일 출판

자연선택을 통한 생물들의 진화를 이론적으로 제시

현대의 진화생물학의 토대가 됨



Part 1. 자연상태에서 발생하는 변이

어떤 부분이라 하더라도 그것이 완성된 상태로 갑자기 태어났다고 하는 것은 어떤 복잡한 기계가 완성된 상태로 발명되는 것과 같은 것으로 있을 수 없는 일이라고 생각한다.

종의 기원 4장 자연상태에서 발생하는 변이 中 61p

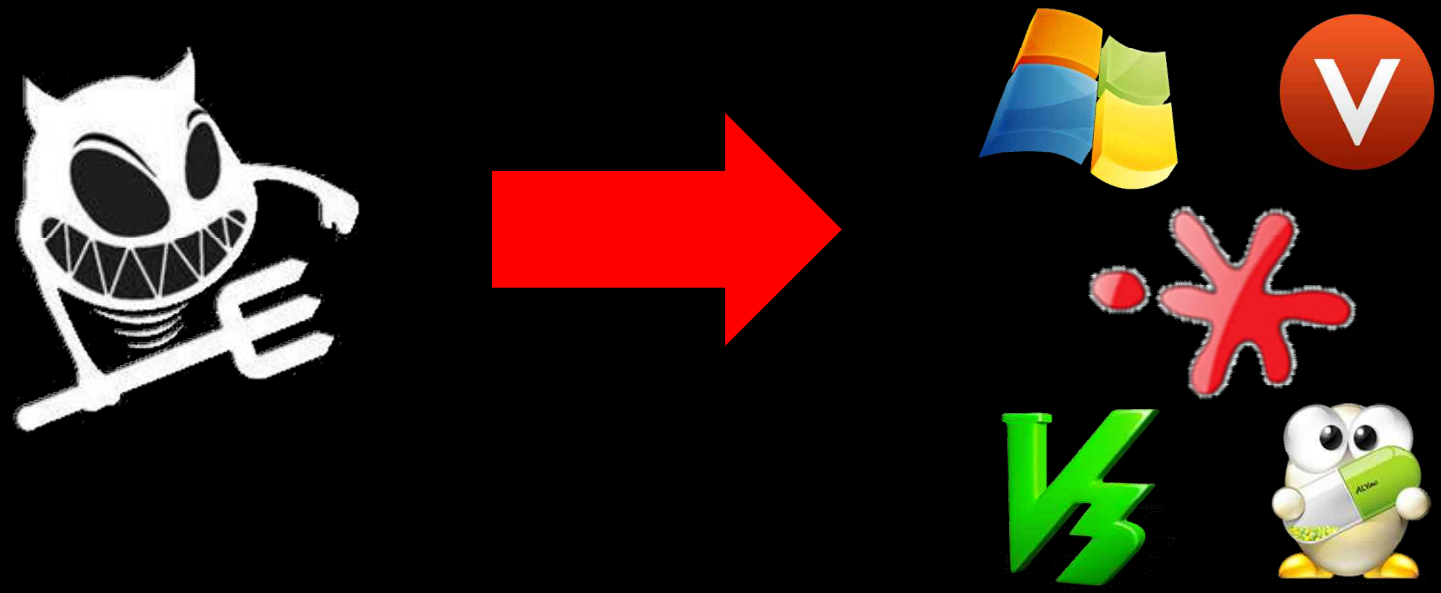
Part 1. Self Protection

Part 1. 환경적 요소

Part 1. 자체적 요소

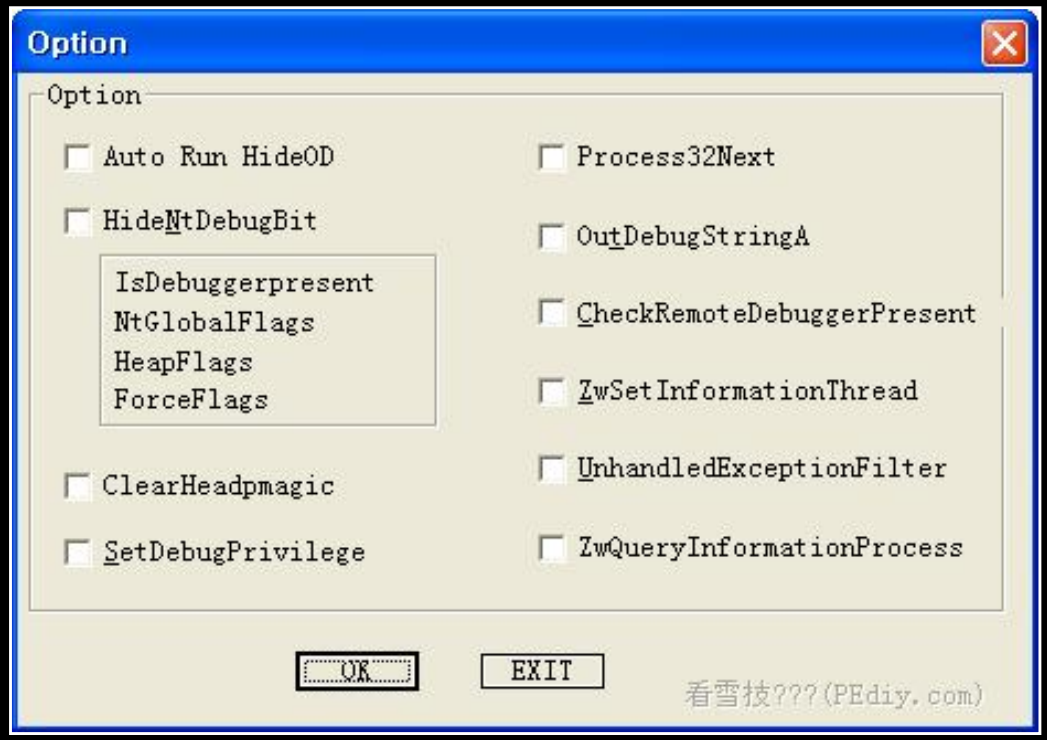
10000000	00001000	tflrq		PE header
10001000	0000B000	tflrq	.text	code
1000C000	00004000	tflrq	.rdata	
10010000	00008000	tflrq	.data	data
10018000	00016000	tflrq	.dasf0	
1002E000	0001C000	tflrq	.dasf1	SFX,imports
1004A000	00001000	tflrq	.reloc	relocations
1004B000	00004000	tflrq	.rsrc	resources
3F230000	00001000	wininet		PE header
3F231000	000B1000	wininet	.text	code,import
3F2E2000	00007000	wininet	.data	
3F2E9000	00027000	wininet	.rsrc	resources
3F310000	00007000	wininet	.reloc	relocations
3F8D0000	00001000	iertutil		PE header
3F8D1000	001CE000	iertutil	.text	code,import
3FA9F000	00005000	iertutil	.data	
3FAA4000	00001000	iertutil	.rsrc	resources
3FAA5000	00017000	iertutil	.reloc	relocations

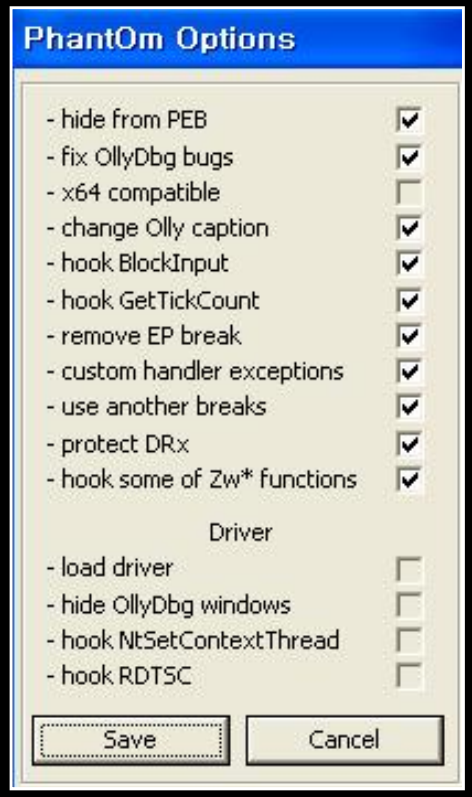
Part 1. 악성코드 VS …

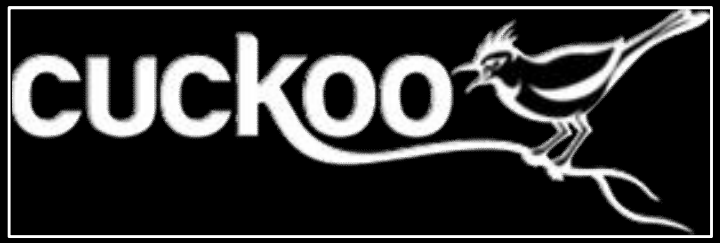


```
if ( !v38 && (StrStrIA(&First, "alyac") || StrStrIA(&First, "ahnlab") || StrStrIA(&First, "v3lite")) )
  v55 = 1;
```

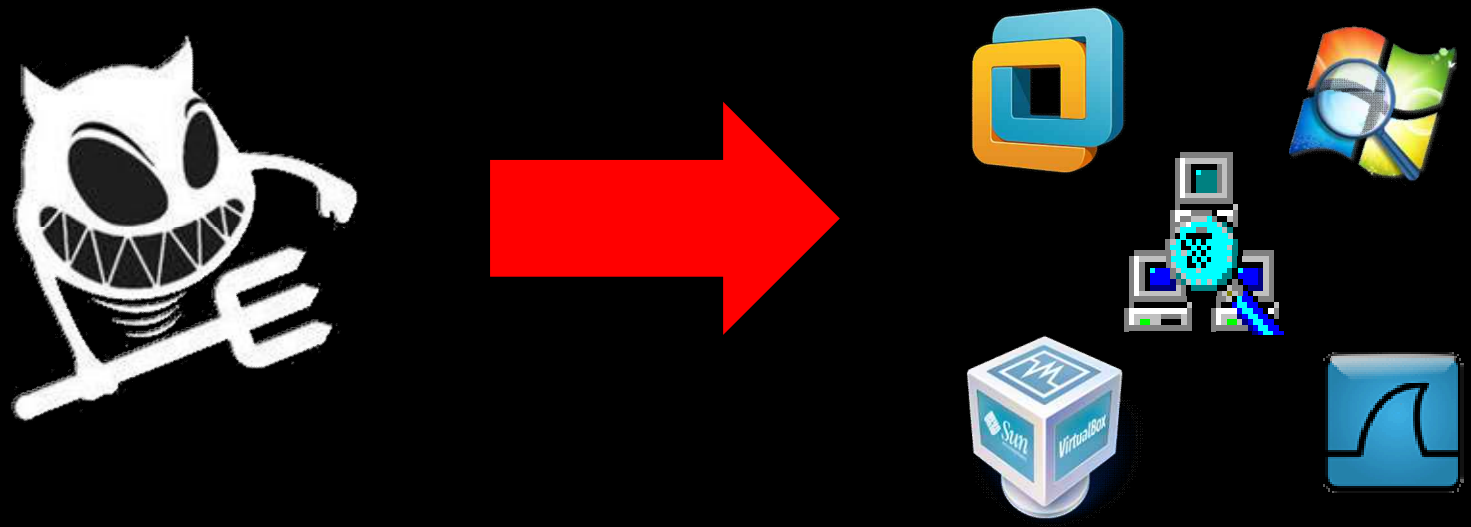
Part 1. 환경의 변화


Option
Option
Auto Run HideOD
HideNtDebugBit
IsDebuggerpresent
NtGlobalFlags
HeapFlags
ForceFlags
ClearHeadpmagic
SetDebugPrivilege
Process32Next
OutDebugStringA
CheckRemoteDebuggerPresent
ZwSetInformationThread
UnhandledExceptionFilter
ZwQueryInformationProcess
OK
EXIT
看雪技???(PEdiy.com)


PhantOm Options
- hide from PEB
- fix OllyDbg bugs
- x64 compatible
- change Olly caption
- hook BlockInput
- hook GetTickCount
- remove EP break
- custom handler exceptions
- use another breaks
- protect DRx
- hook some of Zw* functions
Driver
- load driver
- hide OllyDbg windows
- hook NtSetContextThread
- hook RDTSC
Save
Cancel


cuckoo

Part 1. 악성코드 VS …



Part 2. 생존경쟁

아무리 경미한 변이라도 유용한 점이 있으면 보존되는
이 원리를 인간선택 능력과 구분하기 위해 나는
자연선택 이라는 용어로 부르기로 했다.

종의 기원 3장 생존경쟁 中 80p

환경적 요소

Part 2. 가상머신

- 가상머신 시그니쳐
  - Hardware Info
  - Software Info

```
if ( !j_GetAdaptersInfo(&AdapterInfo, &SizePointer) )
{
  v1 = &AdapterInfo;
  while ( 1 )
  {
    v2 = v1->Address[0];
    if ( !v2 && v1->Address[1] == 0xC )        // Check VMWare
      break;
    if ( v2 == 8 && !v1->Address[1] )          // Check VirtualBox
      break;
```

```
CALL to GetFileAttributesA from 00C70B02
FileName = "C:\WINDOWS\system32\drivers\vmmouse.sys"
CALL to GetFileAttributesA from 00C70B17
FileName = "C:\WINDOWS\system32\drivers\vmhgfs.sys"
CALL to GetFileAttributesA from 00C70D67
FileName = "C:\WINDOWS\system32\drivers\VBoxMouse.sys"
```

Part 2. 가상머신

- 가상머신 시그니쳐
  - Hardware Info
  - Software Info

```
CALL to RegOpenKeyExA from 00C70CA0
hKey = HKEY_LOCAL_MACHINE
Subkey = "SOFTWARE\Oracle\VirtualBox Guest Additions"
Reserved = 0
Access = KEY_READ
pHandle = 0012F3B8
CALL to RegOpenKeyExA from 00C70CD8
hKey = HKEY_LOCAL_MACHINE
Subkey = "HARDWARE\Description\System"
Reserved = 0
Access = KEY_READ
pHandle = 0012F3B8
    CALL to strlen from 00C70C2B
    s = "INTEL   - 6040000"
```

```
CALL to RegOpenKeyExA from 00C706AE
hKey = HKEY_LOCAL_MACHINE
Subkey = "SYSTEM\CurrentControlSet\Services\Disk\Enum"
Reserved = 0
Access = KEY_READ
pHandle = 0012F42C
CALL to RegOpenKeyExA from 00C70A4B
hKey = HKEY_LOCAL_MACHINE
Subkey = "HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target Id 0\Logical Unit Id 0"
Reserved = 0
Access = KEY_READ
pHandle = 0012F3B8
CALL to RegOpenKeyExA from 00C70AEC
hKey = HKEY_LOCAL_MACHINE
Subkey = "SOFTWARE\VMware, Inc.\VMware Tools"
Reserved = 0
Access = KEY_READ
pHandle = 0012F3B8
```

Part 2. 가상머신

```
do
{
  wsprintfW(&FileName, L"\\\\.\\PhysicalDrive%d", Index);
  hFile = CreateFileW(&FileName, 0, 3u, 0, 3u, 0, 0);
  if ( hFile != -1 )
  {
    retn = DeviceIoControl(hFile, 0x70000u, 0, 0, &Cylinders, 0x18u, &BytesReturned, 0);
    CloseHandle(hFile);                         // IOCTL_DISK_GET_DRIVE_GEOMETRY
    if ( retn )
      Total += BytesPerSector * SectorsPerTrack * TrackPerCylinder * Cylinders;
  }                                             // SectorsPerTrack * BytesPerSector * TracksPerCylinder+ Cylinders
  ++Index;
}
while ( Index < 3 );
return SHIDWORD(Total) >= 0x19;
```

Part 2. 가상머신

```
ProcessList_41A9DC dd offset aWnxbsf    ; DATA XREF: EnumFu
                                        ; "WNXBSF"
                   dd offset aOfunpo    ; "OFUNPO"
                   dd offset aXjsftibsl ; "XJSFTIBSL"
                   dd offset aBvupsvot  ; "BVUPSVOT"
                   dd offset aQspdfyq   ; "QSPDFYQ"
                   dd offset aQspdnpo   ; "QSPDNPO"
                   dd offset aUdqwjfx   ; "UDQWJFX"
                   dd offset aXjobmztjt ; "XJOBMZTJT"
                   dd offset aQtfyftwd  ; "QTFYFTWD"
                   dd offset aQfuppmt   ; "QFUPPMT"
                   dd offset aOfusu     ; "OFUSU"
                   dd offset aWnuppmte  ; "WNUPPMTE"
                   dd offset aTzofshzd  ; "TZOFSHZD"
                   dd offset aPmmzech   ; "PMMZECH"
                   dd offset aEvnqdbq   ; "EVNQDBQ"
                   dd offset aJebh      ; "JEBH"
                   dd offset aGjmfnpo   ; "GJMFNPO"
                   dd offset aSfhtipu   ; "SFHTIPU"
                   dd offset aTcjftwd   ; "TCJFTWD"
                   dd offset aXjoech    ; "XJOECH"
                   dd offset aQfje      ; "QFJE"
                   dd offset aTztbobmzFs ; "TZTBOBMZ[FS"
                   dd offset aBqjnpojups ; "BQJNPOJUPS"
                   dd offset aXjotzt    ; "XJOTZT"
                   dd offset aUqbvupdpoofdu ; "UQBVUPDPOOFDU"
                   dd offset aQspdfyq75 ; "QSPDFYQ75"
```

```
if ( !GetWindowTextLengthW(hWnd) )
  goto LABEL_8;
GetWindowTextW(hWnd, &String, 260);
v2 = wcslen(&String);
if ( v2 > 0 )
{
  v3 = &String;
  v4 = v2;
  do
  {
    *v3 = towupper(*v3) + 1;
    ++v3;
    --v4;
  }
  while ( v4 );
}
v5 = 0;
v6 = (const wchar_t **)ProcessList_41A9DC;
while ( !wcsstr(&String, *v6) )
{
  ++v6;
  ++v5;
  if ( (signed int)v6 >= (signed int)L"QSPDFYQ75" )
    goto LABEL_8;
}
```

```
Vmware   VMWARE   NETMON   WIRESHARK
AUTORUNS          PROCEXP PROCMON TCPVIEW
WINALYSIS         PSEXESVC         PETOOLS NETRT
VMTOOLSD          SYNERGYC         OLLYDBG DUMPCAP
IDAG     FILEMON REGSHOT SBIESVC
WINDBG   PEID     SYSANALYZER      APIMONITOR
WINSYS   TPAUTOCONNECT    PROCEXP64
```

Part 2. 가상머신

```
v0 = CreateToolhelp32Snapshot(2u, 0);
if ( v0 && (pe.dwSize = 556, Process32FirstW(v0, &pe)) )
{
  while ( !FileNameToHash_4023A0(pe.szExeFile) )
  {
    if ( !Process32NextW(v0, &pe) )
      goto LABEL_5;
  }
  result = 1;
}
```

```
                ProcessNameHash_41A96C dd 41B088h       ;
"9196D1767133E3B982DB0413C1C3961C9E41547"; a9196d1767133e3
"BCDD97B853A5C1404CBC8CCAAFB852798FA155B"; aBcdd97b853a5c1
"A70BEB5E80BEA810C5D9E5D628535855F117C9B"; aA70beb5e80bea8
"E888B9450692098516FBA5FA3911FD54283E94E"; aE888b945069209
"2C1386CF0B32A1BE37743C3AA6BEBEAE60C16A2"; a2c1386cf0b32a1
"689C4006F25B274BCBFF9312A4ED41F4DF99637"; a689c4006f25b27
"1E95024A9E83DA0DD177830E7360B9A3B9A917C"; a1e95024a9e83da
"C8E6C2C5871C2B9CB623B5B2A509AE2C3D80750"; aC8e6c2c5871c2b
"59EF909F37AF59C9B896F017D9C4F1AE90DA6B3"; a59ef909f37af59
"D40F7E6E00A2553961DB29E81D779D0D9382E4E"; aD40f7e6e00a255
"294CEF81F5063689FD042C3C29961166A20A03B"; a294cef81f50636
"0F186CA71C6A5645414408F5A172D0F3456DCA6"; a0f186ca71c6a56
"CA0DC035C9E7F5993DC58DAA0F94F1317F3697C"; aCa0dc035c9e7f5
"E540E9E856A1ED505A77A8C1C01529DD4357C84"; aE540e9e856a1ed
"769C67DC336D803B874484F174D3DE7E61C584E"; a769c67dc336d80
"FBCB498D106EAC56A997B987AE8BC3D158C9AF5"; aFbcb498d106eac
"C302B0E2356830B9CB7546712EEBBF8E7188778"; aC302b0e2356830
"2CBA872DE5B776EB5A8D44C656235D3F9027CC6"; a2cba872de5b776
"FF5F9332E19BFAB99F544481A13186703E77613"; aFf5f9332e19bfa
"01F927DDFAF92FDD6DFECCB405B006E35C23C30"; a01f927ddfaf92f
"831B8EF8E52E1A8A4473B1E11DFB328CC8D8685"; a831b8ef8e52e1a
"59B18D2A1D6A62D60F15796B9FF97E204DC2CDE"; a59b18d2a1d6a62
"60533F54548A0007966 03DBE6D334A461229FE0"; a60533f54548a00
"ADDFE18E9AE6581B3E6E3F8F11E93A6EBAC46DB"; aAddfe18e9ae658
"64D3207A264DD5C78DD4A28B4808D8077C7FC6C"; a64d3207a264dd5
"5271A6FDE87C430F386C8AF91A803A1ACC1A1CB"; a5271a6fde87c43
```

Part 2. 샌드박스

- Sandbox (Using DLL)
  - sandboxie : sbiedll.dll
  - SunBelt Sandbox: dir_watch.dll, api_log.dll
  - Window: SandboxiecontrolWndClass

```
CALL to GetModuleHandleA from 00C71610
pModule = "sbiedll.dll"
CALL to GetModuleHandleA from 00C7162F
pModule = "api_log.dll"
CALL to GetModuleHandleA from 00C7164B
pModule = "dir_watch.dll"
```

```
CALL to GetModuleFileNameA
hModule = NULL
PathBuffer = 0012F228
BufSize = 1F4 (500.)
CALL to strstr from 00C713FE
s1 = "C:\CODEENGN.EXE"
s2 = "\SAMPLE"
CALL to strstr from 00C7141C
s1 = "C:\CODEENGN.EXE"
s2 = "\VIRUS"
CALL to strstr from 00C71436
s1 = "C:\CODEENGN.EXE"
s2 = "SANDBOX"
```

```
CALL to GetUserNameA
Buffer = 0012F350
pBufCount = 0012F418
CALL to strstr from 00C71355
s1 = "ADMINISTRATOR"
s2 = "SANDBOX"
CALL to strstr from 00C71370
s1 = "ADMINISTRATOR"
s2 = "VIRUS"
CALL to strstr from 00C71387
s1 = "ADMINISTRATOR"
s2 = "MALWARE"
```

Part 2. 샌드박스

- CreateProcessA 함수의 첫 1byte를 JMP와 비교해 후킹여부 탐지

```
CALL to GetProcAddress from 00C7129B
 hModule = 7C7D0000 (kernel32)
 ProcNameOrOrdinal = "CreateProcessA"
CALL to GetModuleHandleA
 pModule = "KERNEL32.dll"
CALL to ReadProcessMemory from 00C712AB
 hProcess = FFFFFFFF
 pBaseAddress = 7C7D236B
 Buffer = 0012F427
 BytesToRead = 1
 pBytesRead = NULL
```

```
7C7D236B CreateProcessA   8BFF        MOV EDI,EDI
7C7D236D                  55          PUSH EBP
7C7D236E                  8BEC        MOV EBP,ESP
```

```
00C712A7   FF56 60      CALL DWORD PTR DS:[ESI+60]   ReadProcessMemory
00C712AA   50           PUSH EAX
00C712AB   FF56 5C      CALL DWORD PTR DS:[ESI+5C]
00C712AE   33C0         XOR EAX,EAX
00C712B0   807D 0B E9   CMP BYTE PTR SS:[EBP+B],0E9  E9 = JMP
```

Part 2. 자체적 요소

PE Header

Code

CALL LoadLibrary("kernel32.dll")

CALL GetProcAddress("CreateFileA")

MOV NewIAT+OFFSET, EAX

...

CALL [NewIAT + OFFSET]

Import Address Table

0x7C7DAAAA 0x7C7DBBBB 0x7C7DCCCC

0x7C7DDDDD 0x7C7DEEEE 0x7C7DFFFF

New Import Address Table

0x7C7DAAAA 0x7C7DBBBB 0x7C7DCCCC

0x7C7DDDDD 0x7C7DEEEE 0x7C7DFFFF

기본적인 IAT 재구성

```
v6 = sub_401810("S^CloseHandle");
dword_441938 = (int)GetProcAddress(v1, v6);
v7 = sub_401810("S^GetSystemDirectoryA");
dword_44195C = (int)GetProcAddress(v1, v7);
v8 = sub_401810("S^SetFileTime");
dword_441930 = (int)GetProcAddress(v1, v8);
v9 = sub_401810("S^GetFileTime");
dword_441940 = (int)GetProcAddress(v1, v9);
```

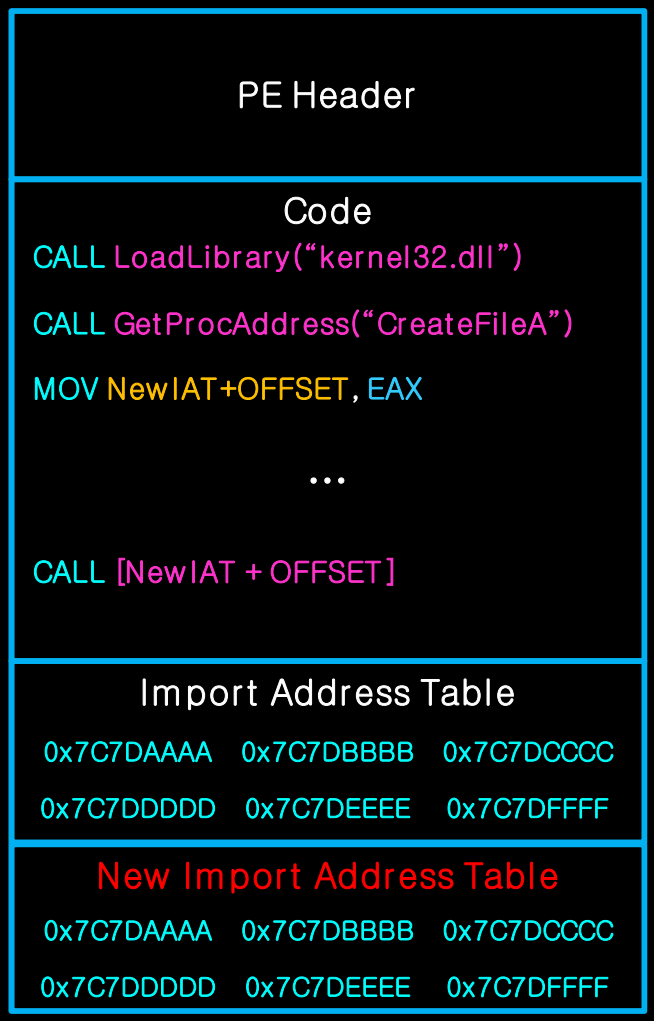
배열과 Index를 이용한 IAT 재구성

```
HMODULE __stdcall sub_405CFF(int a1)
{
  const CHAR *v1; // edi@1
  HMODULE result; // eax@1

  v1 = DllTable_409200[2 * a1];
  result = GetModuleHandleA(DllTable_409200[2 * a1]);
  if ( result || (result = LoadLibraryA(v1)) != 0 )
    result = GetProcAddress(result, ApiTable_409204[2 * a1]);
  return result;
}
```

```
DllTable_409200 dd offset aKernel32     ; DATA XREF: sub_405CFF+B↑r
                                        ; "KERNEL32"
ApiTable_409204 dd offset aGetdiskfreespa ; DATA XREF: sub_405CFF:1
                                        ; "GetDiskFreeSpaceExA"
                dd offset aKernel32     ; "KERNEL32"
                dd offset aMovefileexa  ; "MoveFileExA"
                dd offset aAdvapi32     ; "ADVAPI32"
                dd offset aRegdeletekeyex ; "RegDeleteKeyExA"
                dd offset aAdvapi32     ; "ADVAPI32"
                dd offset aOpenprocesstok ; "OpenProcessToken"
                dd offset aAdvapi32     ; "ADVAPI32"
                dd offset aLookupprivileg ; "LookupPrivilegeValueA"
```

Part 2. 자체적 요소



API 호출 후 IAT 구성

```
int sub_407DC5()
{
  int (*v0)(void); // eax@1

  v0 = sub_40A850("user32", "GetSystemMetrics", &off_40DCF3);
  return v0();
}
```

API 인코딩

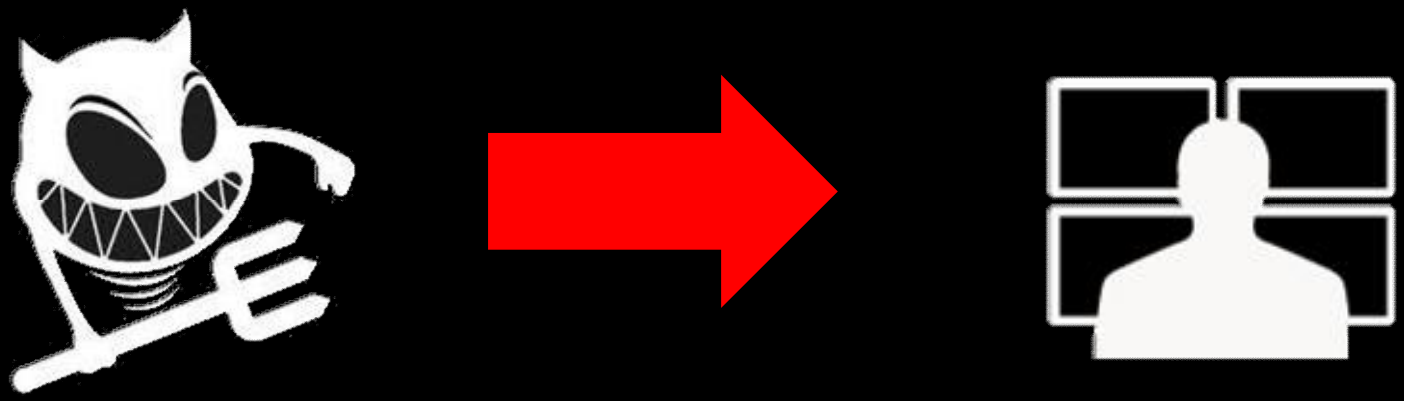
```
dword_414EB4 = GetProcAddress(v2, &ProcName);
dword_414F00 = sub_4028C0("RvtOkvmKvbEcW", v0);
dword_414EC4 = sub_4028C0("RvtDvovgvKvbW", v0);
dword_414ED8 = sub_4028C0("RvtQfvibVaofvEcW", v0);
dword_414EDC = sub_4028C0("RvtColhvKvb", v0);
dword_414EF4 = sub_4028C0("RvtSvgVaofvEcW", v0);
dword_414EF8 = sub_4028C0("RvtCivagvKvbW", v0);
```

상관변이

상관변이, 나는 이 말을 성장과 발달의 기간 주에는
모든 체제가 긴말하게 결합해 있고, 어느 부분에
경미한 변이가 일어나 자연선택에 의해 축적되면
다른 부분도 변화하게 된다는 의미로 사용한다.

종의 기원 5장 변이의 법칙 中 156p

Part 3. 악성코드 VS …



Part 3. 성전이다.

- SendMessageA 함수에 JMP SELF (0xEBFE) 삽입 후 호출
  - 디버깅 시 무한루프에 빠질 수 있음

```
VirtualProtectEx_418FBF();                      // SendMessageA -> RW
if ( lp[1] != &v60 )
  ERROR_418F9B(6);
InstallJMPSELF_419FB0(3, (int)dword_41F504, 0, 0xA0000101, 0, 0, 0x800
lp[1] = &v60;
lp[0] = 0;
VirtualProtectEx_418FBF();                      // SendMessageA -> Read
```

```
77D0F3C2 SendMessageA   8BFF    MOV EDI,EDI
77D0F3C4                55      PUSH EBP
```

```
77D0F3C2 SendMessageA - EB FE   JMP SHORT USER32.SendMessageA
77D0F3C4                55      PUSH EBP
```

- ZwSuspendProcess / ZwResumeThread 를 통해 디버깅 방해
  - 분석시 마우스를 제외하고 화면이 멈춰 분석 진행 불가

```
*(a1 - 0x14) = OpenProcess_4087E3(v10, 0x1F0FFF, v46, v47);
if ( *(a1 - 0x14) > 0 )                         // OpenProcess(Csrss.exe)
{
  *(a1 - 0x24) = &v48;
  v47 = *(a1 - 0x14);
  v11 = ZwSuspendProcess_419109();
```

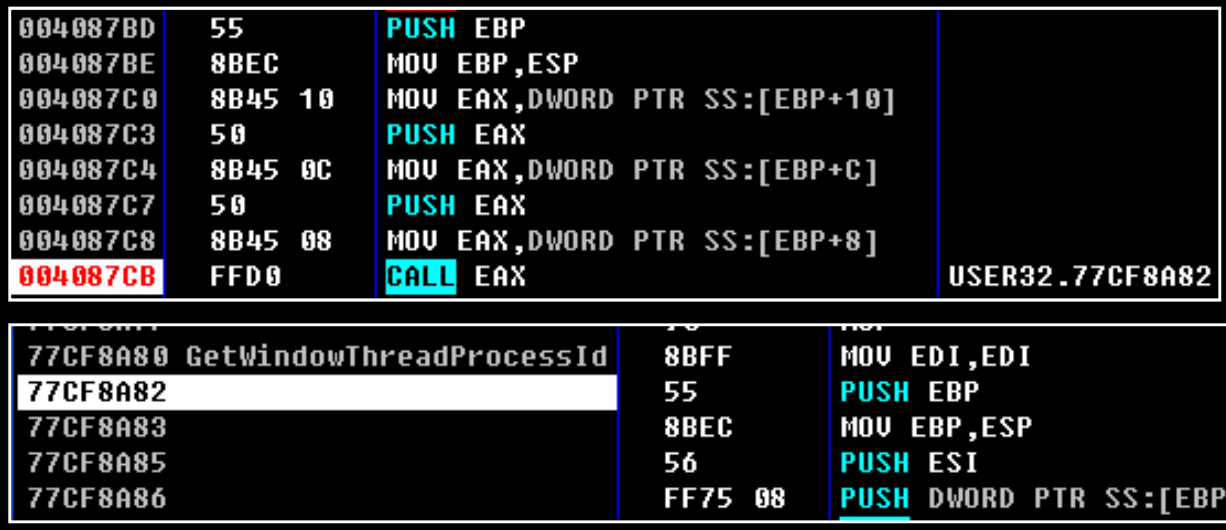
Part 3. 성전이다.

- GetTickCount와 Sleep를 통한 실행 시간체크
  - GetTickCount는 함수를 호출하지 않고 함수를 복사한 후 사용
  - Sleep 함수 내에 BreakPoint (0xCC) 가 존재하는 지 확인해 우회가 어렵도록 구성
  - 차이가 커질 경우 DeadLoop로 이동

```
v3 = GetRunningTime_403632();
if ( GetSleepEPCode_40392E(0) == 0xCC )
  break;
SLEEP_40A7E5(500);
v2 = GetRunningTime_403632();
if ( (double)v3 + 2100.0 - (double)v2 < -0.0000001 )
  sub_40A366(v0);
```

Part 3. 성전이다.

- CALL API+2
  - 일반적으로 API에 BP를 걸고 분석을 진행
  - 이를 우회하기 위해 API의 +2위치 (API실행흐름과 관계없는 부분)를 호출



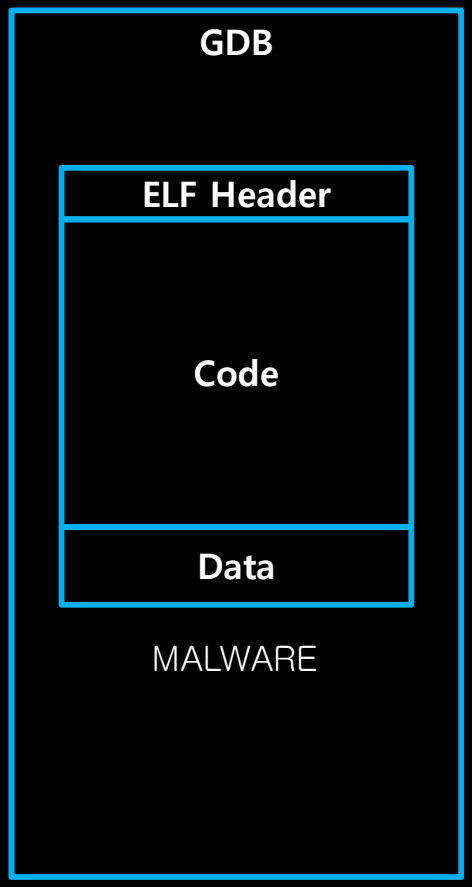
비 보편적인 종의 발전

널리 분포해있고 분산성이 높은
보편적인 종이 더 많이 발전한다

종의 기원 2장 자연상태에서 발생하는 변이 中 72p

Part 4. 리눅스 악성코드

- GDB Detect



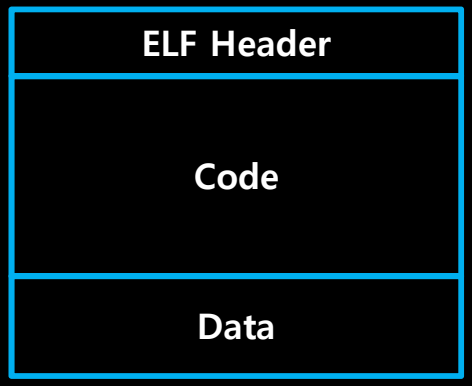

```
CUtility *__stdcall CUtility::GetParentPath(CUtility *this)
{
  char v1; // al@1
  char buf; // [sp+13h] [bp-205h]@1
  char filename; // [sp+113h] [bp-105h]@1
  char v5; // [sp+213h] [bp-5h]@1

  memcpy(&filename, &CUtility::GetParentPath(void)::C.92, 256);
  v1 = getppid();
  sprintf(&filename, "/proc/%d/exe", v1);
  memcpy(&buf, &CUtility::GetParentPath(void)::C.93, 256);
  readlink(&filename, &buf, 0xFFu);
  std::allocator<char>::allocator(&v5);
  std::string::string(this, &buf, &v5);
  std::allocator<char>::~allocator(&v5);
  return this;
}
```

```
CUtility::GetParentPath((CUtility *)&v14);
v6 = std::string::c_str((std::string *)&v14);
if ( strstr(v6, "gdb") != 0 )
  v0 = 0;
```

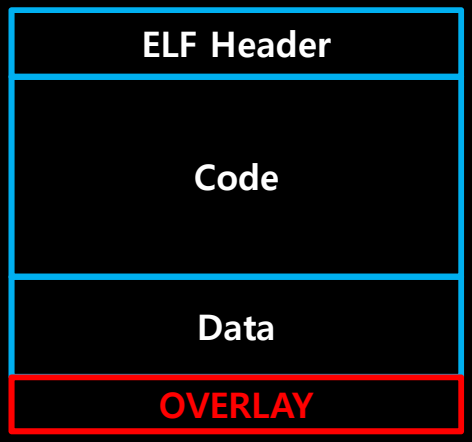
Part 4. 리눅스 악성코드

- Random Hash , Random Name



A56BCF1F87

```
randstr((int)&v24, 10);                   // Get Random File Name
snprintf((int)&v30, 1024, "%s%s", &v23, &v24);
snprintf((int)&v29, 1024, "%s%s", &v22, &v24);
snprintf((int)&v28, 1024, "%s%s", &v21, &v24);
get_self(&filename, 1024);
copyfile(&filename, &v18);
if ( copyfile(&filename, &v30) )
{
  randmd5(&v30);
  LinuxExec(&v30);
}
```



54AC9FG1D6

```
fd = open(filename, 1, v2);
if ( fd > 0 )
{
  lseek(fd, 0, 2);
  randstr(&addr, 10);
  write(fd, &addr, 0xBu);
  close(fd);
}
return 0;
```

```
v4 = doublefork();
if ( !v4 )
{
  for ( fd = 3; fd <= 1023; ++fd )
    close(fd);
  argv = file;
  execvp(file, &argv);
  exit(0);
}
```

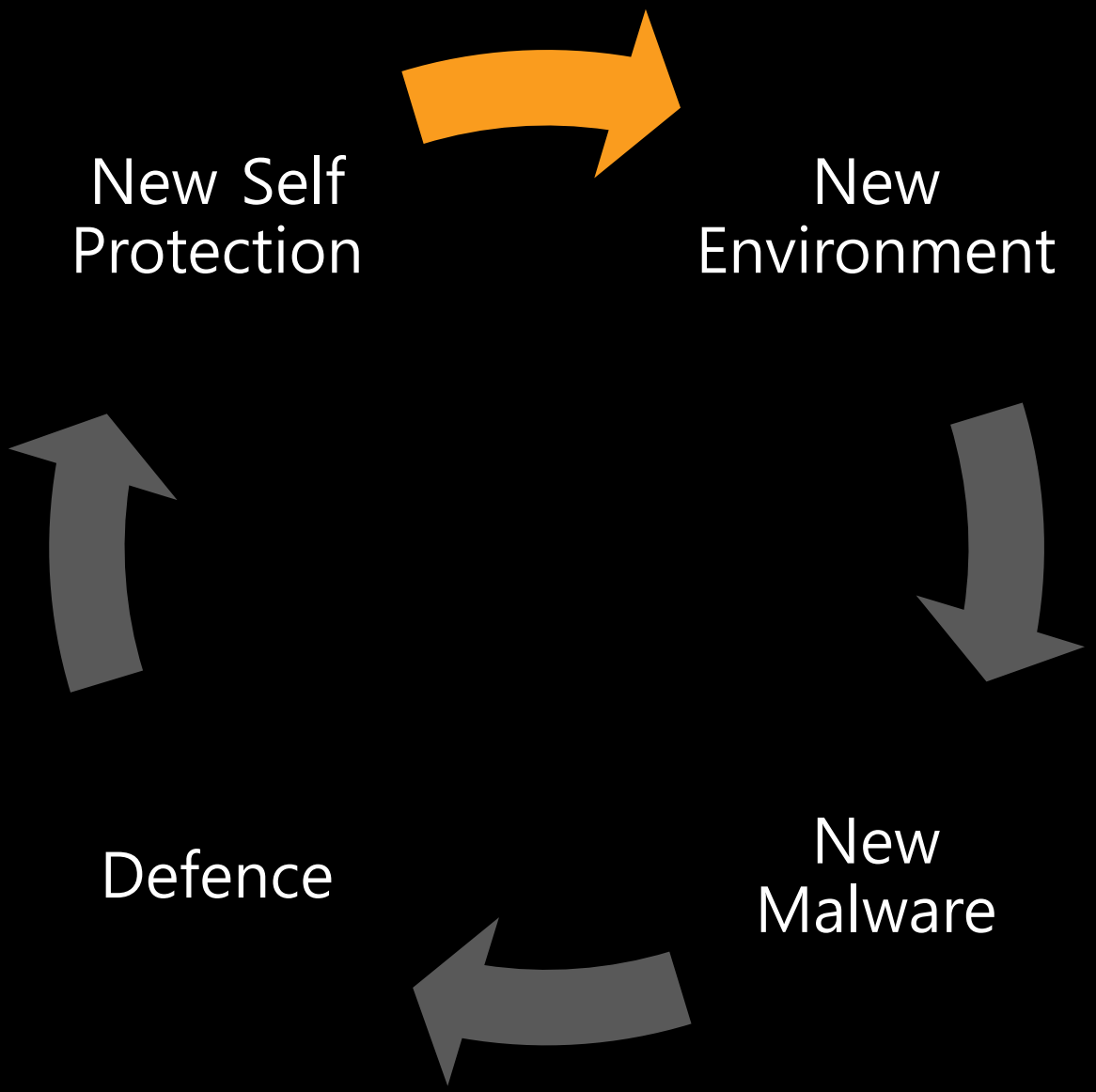
The Mental Breaker

```
a__h3ya__UuMadR:.ascii "..h3yA!.. Uu MAD R3v3rz3r???Wn"<0>
                .ascii "Wn"<0>
aFuckyou:       .ascii "fuckyou ;)Wn"<0>
aEatShit___:    .ascii "eat shit...Wn"<0>
                .ascii "Wn"<0>
```

Fuck Yeah!!

```
; char Text[]
Text            db 'This binary is invalid.',0Dh,0Ah
                                        ; DATA XREF: sub_98A5
                db 0Dh,0Ah
                db 'Main reasons:',0Dh,0Ah
                db '- you stupid cracker',0Dh,0Ah
                db '- you stupid cracker...',0Dh,0Ah
                db '- you stupid cracker?!',0Dh,0Ah,0
```

결론



결론

~~인간은 변종을 창조해 낼 수도 없고,~~
또 변종의 발생을 막을 수 도 없다. 다만
그것이 발생할 것을 보존해서 누적시킬 수 있을 따름이다.

종의 기원 4장 자연선택 또는 적자생존 中 95p

Any Questions?