Fingerprinting Malware Authors

Introductory Case Study of a Chinese APT
The Bad Guys are Winning

• Cybercrime & espionage is the dominant criminal problem globally, surpassing the drug trade
  – Russians made more money last year in banking fraud than the Colombians made selling cocaine
  – Chinese are crawling all over commercial & government networks

• The largest computing cloud in the world is controlled by Conficker
  – 6.4 million computer systems*
  – 230 countries
  – 230 top level domains globally
  – 18 million+ CPUs
  – 28 terabits per second of bandwidth

Humans

- Attribution is about the human behind the malware, not the specific malware variants
- Focus must be on human-influenced factors

We must move our aperture of visibility towards the human behind the malware
$100.00 per 1000 infections

$10,000+ for 0-day

$1,000+ $1000+
$500+ $500 +

$10,000+ for 0-day

$10,000+

Small Transfers

Wizard

Rootkit Developer

Exploit Pack Vendor

Bot Vendor

Exploit Developer

Back Office Developer

Rogueware Developer

Payment system developer

Victims

$100.00 per 1000 infections

Country that doesn’t co-op w/ LE

Keep 10%

Keep 50%

$5,000 incrnm.

Country where account is physically located

A single operator here may recruit 100’s of mules per week

Keep 50%

Keep 10%

$50

$50

Forger

Cashier / Mule

Bank Broker

Drop Man

Account Buyer

Affiliate Botmaster ID Thief

Sells accounts in bulk

$5.00 per

eGold

~4% of bank customers

PPI

Victims

Secondary

Wizard

Small Transfers

Wizard

Small Transfers

A single operator here may recruit 100’s of mules per week

$5,000 incrnm.

Country where account is physically located

$10,000+

$1,000+

$100.00

Country that doesn’t co-op w/ LE

Keep 10%

Country that doesn’t co-op w/ LE

Keep 10%

Country that doesn’t co-op w/ LE

Keep 10%

Country that doesn’t co-op w/ LE

Keep 10%
Intelligence Spectrum

Blacklists
Developer Fingerprints
Social Cyberspace DIGINT
Physical Surveillance HUMINT

MD5 Checksum of a single malware sample

Nearly Useless

Sweet Spot
NIDS & HIDS signatures with long-term viability
Predict the attacker’s next moves

Nearly Impossible

SSN & Missile Coordinates of the Attacker
Rule #1

• The human is lazy
  – The use kits and systems to change checksums, hide from A/V, and get around IDS
  – They DON’T rewrite their code every morning
Rule #2

• Most attackers are focused on rapid reaction to network-level filtering and black-holes
  – Multiple DynDNS C2 servers, multiple C2 protocols, obfuscation of network traffic

• They are not-so-focused on host level stealth
  – Most malware is simple in nature, and works great
  – Enterprises rely on A/V for host, and A/V doesn’t work, and the attackers know this
Rule #3

• Physical memory is King
  – Once executing in memory, code has to be revealed, data has to be decrypted
MD5 is not consistent.

Software Traits remain consistent.

MD5 is Useless In memory.
Software Traits remain consistent

OS Loader

IN MEMORY IMAGE

Packer #1

Packer #2

Decrypted

Original

Physical memory tends to get around the ‘packing’ problem
DISK FILE

MD5 Checksums all different

IN MEMORY IMAGE

Software Traits remain consistent

OS Loader

Same malware compiled in three different ways
Attribution is Not Hard

• If you can read a packet sniffer, you can attribute malware
  – Yes, this means more people in your organization can do this
  – Focus on strings and human-readable data within a malware program
  – In most cases, code-level reverse engineering is not required
The Flow of Forensic Toolmarks
(host perspective)

- Malware Developer
  - Core ‘Backbone’ Source code
  - Tweaks & Mods
  - 3rd party Source code
  - 3rd party libraries

- Compiler
  - Runtime Libraries
    - Time/Date
    - Paths
    - MAC address

- Machine
  - Malware
    - Packing
    - Encrypting
    - Obfuscation

- Many Variants
  - Malware 1
  - Malware 2
  - Malware 3
  - Malware 4
  - Malware 5
  - Malware ~N

- Automation

© 2010 HB Gary, Inc. All Rights Reserved
Developer Fingerprints

- Communications Functions
- Installation & Deployment Method
- Command & Control Functions
- Compiler Environment
- Stealth & Anti-forensic Techniques

Sample
- Malware
- Packing

Developer
Toolkit Fingerprints

Machine

PPI Affiliate

Toolkit

Packed Malware
IN MEMORY IMAGE

Toolkit traits are apparent

OS Loader

Packed

Toolkits can be detected

Malware Toolkit

Different Malware Authors Using Same Toolkit
Example: Gh0stNet
GhostNet: Dropper

This program cannot be run in DOS mode.

Embedded executable

NOTE: Packing is not fully effective here.

Packer Signature

MZx90

UPX!

UPX!
GhostNet: Dropper

This program cannot be run in DOS mode

The embedded executable is tagged with Chinese PRC Culture code
GhostNet: Dropper

The embedded executable is extracted to disk. The extracted module is **not packed**. PDB path reveals malware name, E: drive.

This program cannot be run in DOS mode

E:\gh0st\Server\Release\install.pdb

Embedded PDB Path
For Immediate Defense...

MD5 of the Gh0stNet dropper.EXE

PDB Path found within extracted EXE

Query: “Find Attacker’s PDB Path”

RawVolume.File.BinaryData
contains
“gh0st\”
Link Analysis

The web reveals Chinese hacker sites that reference the “gh0st” artifact.
The dropped EXE is loaded as svchost.exe on the victim. It then drops another executable, a device driver.
Our defense...

Query: “Find Attacker’s PDB Path”

RawVolume.File.BinaryData contains "gh0st\"

Even if we had not known about the second executable, our defense would have worked. This is how moving towards the human offers predicative capability.
What do we know...

i386 directory is common to device drivers. Other clues:
1. sys directory
2. ‘SSDT’ in the name

SSDT means **System Service Descriptor Table** – this is a common place for rootkits and HIPS products to place **hooks**.

Also, embedded strings in the binary are known driver calls:
1. IoXXXX family
2. KeServiceDescriptorTable
3. ProbeForXXXX

**KeServiceDescriptorTable** is used when SSDT hooks are placed. We know this is a hooker.
What do we know...

IofCompleteRequest, IoCreateDevice, IoCreateSymbolicLink, and friends are used when the driver communicates to usermode. This means there is a usermode module (a process EXE or DLL) that is used in conjunction with the device driver.

When communication takes place between usermode & kernelmode, there will be a device path.
For Immediate Defense...

MD5 of the Gh0stNet dropper.EXE

Device Path of the kernel mode driver and the Symbolic Link name

Query: “Find Rootkit Device Path or Symlink”

Physmem.WindowsObject.Name

contains

“RESSDT”
A readme file on Kasperky’s site references a Ressdt rootkit.
Forensic Toolmarks

e:\gh0st\server\sys\i386\RESSDT.pdb

e:\job\gh0st\Release\Loader.pdb

.?AVCgh0stDoc@@

.?AVCgh0stApp@@

.?AVCgh0stView@@

Cgh0stView

Cgh0stDoc

e:\job\gh0st\Release\gh0st.pdb

C:\gh0st3.6_src\HACKER\i386\HACKE.pdb

\gh0st3.6_src\Server\sys\i386\CHENQI.pdb

Rootkit

Dropper

GUI (MFC)

Doc/View is usually MFC

Already at version 3.6

Rootkits
Case Study: Chinese APT

SvcHost.DLL.log

SvcHost.DLL.log & “bind cmd frist!”

SvcHost.DLL.log

Just “bind cmd frist!”
timestamps

developer

core 'backbone' sourcecode

tweaks & mods

3rd party sourcecode

3rd party libraries

machine

compiler

time

paths

mac address

runtime libraries

sample

malware

packing
PE Timestamps

Module timestamp*
- time_t (32 bit)

The ‘lmv’ command in WinDBG will show this value..

Debug timestamp
- time_t (32 bit)

This is present if an external PDB file is associated with the EXE

*This is not the same as NTFS file times, which are 64 bit and stored in the NTFS file structures.
Timestamp Formats

• **time_t** – 32 bit, seconds since Jan. 1 1970 UTC
  – 0x3DE03E0A \(\leftarrow\) usually start with ‘3’ or ‘4’
    • ‘3’ started in 1995 and ‘4’ ends in 2012
  – Use ‘ctime’ function to convert

• **FILETIME** – 64 bit, 100-nanosecond intervals since Jan. 1 1600 UTC
  – 0x01C195C2.5100E190 \(\leftarrow\) usually start with ‘01’ and a letter
    • 01A began in 1972 and 01F ends in 2057
  – Use FileTimeToSystemTime(), GetDateFormat(), and GetTimeFormat() to convert
Compile times extracted from ‘soysauce’ backdoor program.
For Immediate Defense...

Compile time

Useless <-> Human

Query: “Find Modules Created Within Attack Window”

<table>
<thead>
<tr>
<th>RawVolume.File.CompileTime</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 3/1/2010</td>
<td></td>
</tr>
<tr>
<td>&lt; 3/31/2010</td>
<td></td>
</tr>
</tbody>
</table>
MAC Address

Developer

Core ‘Backbone’ Sourcecode
Tweaks & Mods
3rd party Sourcecode
3rd party libraries

Machine

Compiler
Runtime Libraries

Sample
Malware
Packing

Time
Paths
MAC address
GUID V1

• The OSF specified algorithm for GUID V1 uses the MAC address of the network card for the last 48 bits of the 128 bit GUID
  – This was deprecated on Windows 2000 and greater, so this has limited value

{21EC2020-3AEA-1069-A2DD-08002B30309D}

V1 GUIDS have a 1 in this position

This is the MAC of the machine

This technique was used to track the author of the Melissa virus
Visual Studio

- Static or dynamic linked runtime library?
- Single-threaded or multi-threaded?
- Use of STL?
- Use of older iostream libraries?*

See: * support.microsoft.com/kb/154753
### Visual Studio – Static Linking

<table>
<thead>
<tr>
<th>Version</th>
<th>Libraries linked with</th>
<th>Type</th>
<th>Compiler flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC++ .NET 2003 and earlier</td>
<td>LIBC.LIB, LIBCP.LIB</td>
<td>Single Threaded Static</td>
<td>/ML</td>
</tr>
<tr>
<td>VC++ .NET 2003 and earlier</td>
<td>LIBCD.LIB, LIBCPD.LIB</td>
<td>Single Threaded Static</td>
<td>/MLd</td>
</tr>
<tr>
<td>All</td>
<td>LIBCMT.LIB, LIBCPMT.LIB</td>
<td>Multi-threaded Static</td>
<td>/MT</td>
</tr>
<tr>
<td>All</td>
<td>LIBCMTD.LIB, LIBCPMTD.LIB</td>
<td>Multi-threaded Static</td>
<td>/MTd</td>
</tr>
</tbody>
</table>

### Visual Studio – Dynamic Linking

<table>
<thead>
<tr>
<th>Version</th>
<th>DLL Linked with</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC++ 4.2</td>
<td>MSVCRT.DLL/MSVCRTD.DLL</td>
</tr>
<tr>
<td>VC++ 5.0</td>
<td>MSVCR50.DLL</td>
</tr>
<tr>
<td>VC++ 6.0</td>
<td>MSVCR60.DLL</td>
</tr>
<tr>
<td>VC++ .NET 2002</td>
<td>MSVCR70.DLL</td>
</tr>
<tr>
<td>VC++ .NET 2003</td>
<td>MSVCR71.DLL</td>
</tr>
<tr>
<td>VC++ .NET 2005</td>
<td>MSVCR80.DLL</td>
</tr>
<tr>
<td>VC++ .NET 2008</td>
<td>MSVCR90.DLL</td>
</tr>
</tbody>
</table>
Static Linking

- C runtime library strings will be embedded in the EXE itself, as opposed to being in an external DLL
  - DOMAIN error
  - TLOSS error
  - SING error
  - R6027
Debug Symbols

• Debug timestamp (time_t – seconds since 01.01.1970)
• Version of the PDB file
  • NB09 - Codeview 4.10
  • NB11 - Codeview 5.0
  • NB10 - PDB 2.0
  • RSDS - PDB 7.0
• Age – number of times the malware has been compiled
## Name Mangling

<table>
<thead>
<tr>
<th>Compiler</th>
<th>void h(int)</th>
<th>void h(int, char)</th>
<th>void h(void)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel C++ 8.0 for Linux</td>
<td>_Z1hi</td>
<td>_Z1hic</td>
<td>_Z1hv</td>
</tr>
<tr>
<td>HP aC++ A.05.55 IA-64</td>
<td>_Z1hi</td>
<td>_Z1hic</td>
<td>_Z1hv</td>
</tr>
<tr>
<td>GNU GCC 3.x and 4.x</td>
<td>_Z1hi</td>
<td>_Z1hic</td>
<td>_Z1hv</td>
</tr>
<tr>
<td>HP aC++ A.03.45 PA-RISC</td>
<td>h__Fi</td>
<td>h__Fic</td>
<td>h__Fv</td>
</tr>
<tr>
<td>GNU GCC 2.9x</td>
<td>h__Fi</td>
<td>h__Fic</td>
<td>h__Fv</td>
</tr>
<tr>
<td>Microsoft VC++ v6/v7</td>
<td>?h@YAXH@Z</td>
<td>?h@YAXHD@Z</td>
<td>?h@YAXXZ</td>
</tr>
<tr>
<td>Digital Mars C++</td>
<td>?h@YAXH@Z</td>
<td>?h@YAXHD@Z</td>
<td>?h@YAXXZ</td>
</tr>
<tr>
<td>Borland C++ v3.1</td>
<td>@h$qi</td>
<td>@h$qicz</td>
<td>@h$qv</td>
</tr>
<tr>
<td>OpenVMS C++ V6.5 (ARM mode)</td>
<td>H__XI</td>
<td>H__XIC</td>
<td>H__XV</td>
</tr>
<tr>
<td>OpenVMS C++ V6.5 (ANSI mode)</td>
<td>CXX$__7H__FI0ARG51T</td>
<td>CXX$__7H__FIC26CDH77</td>
<td>CXX$__7H__FV2CB06E8</td>
</tr>
<tr>
<td>OpenVMS C++ X7.1 IA-64</td>
<td>CXX$__Z1H2DSQ26A</td>
<td>CXX$__Z1HIC2NP3LI4</td>
<td>CXX$__Z1HV0BCA19V</td>
</tr>
<tr>
<td>SunPro CC</td>
<td>_<em>1cBh6Fi_v</em></td>
<td>_<em>1cBh6Fic_v</em></td>
<td>_<em>1cBh6F_v</em></td>
</tr>
<tr>
<td>Tru64 C++ V6.5 (ARM mode)</td>
<td>h__Xi</td>
<td>h__Xic</td>
<td>h__Xv</td>
</tr>
<tr>
<td>Tru64 C++ V6.5 (ANSI mode)</td>
<td>__7h__Fi</td>
<td>__7h__Fic</td>
<td>__7h__Fv</td>
</tr>
<tr>
<td>Watcom C++ 10.6</td>
<td>W?h$sn(i)v</td>
<td>W?h$sn(ia)v</td>
<td>W?h$sn(v)</td>
</tr>
</tbody>
</table>
Undecorate

Visual C++ demangle:
DWORD WINAPI UnDecorateSymbolName(
    __in PCTSTR DecoratedName,
    __out PTSTR UnDecoratedName,
    __in DWORD UndecoratedLength,
    __in DWORD Flags );

Also, see source to winedbg

GNU C++ demangle
see libiberty/cplus-dem.c and include/demangle.h
Delphi

- Give-away strings:

  SOFTWARE\Borland\Delphi\RTL

  This program must be run under Win32
Delphi

- Uses specific function names – easy to identify
- Language is derived from Pascal

78 hits for pascal, only 2 for c++
Embedded Manifest

• Contains name, description, platform
• Contains list of dependent modules + versions
  – May contain key tokens that identify specific dependent modules (aka strongly named)
• May contain public key that is tied to the developer if assembly itself is strongly named
  – not likely!
  – Public/private key pair (sn.exe)
Tracking Source Code

Developer

Core ‘Backbone’ Sourcecode
- Tweaks & Mods
- 3rd party Sourcecode
- 3rd party libraries

Compiler
- Time
- Paths
- MAC address

Runtime Libraries

Machine

Sample
- Malware
- Packing

© 2010 HBGary, Inc. All Rights Reserved
Main Functions

• Main
  – Same argument parsing
  – Init of global variables
  – WSAStartup
• DllMain
• ServiceMain
Service Routines

• Install / Uninstall Service
• RunDll32
• Service Start/Stop
• ServiceMain
• ControlService
Skeleton of a service

DllMain()
{
    // store the HANDLE to the module in a global variable
}

ServiceMain()
{
    // RegisterServiceCtrlHandler & store handle to service in global variable
    // call SetServiceStatus, set PENDING, then RUNNING
    // call to main malware function(s)
}

ServiceCtrlHandler_Callback
{
    // handle various commands, start/stop/pause/etc
}
Main_Malware_Function
{
    // do stuff
}
InstallService()
{
    // OpenSCManager
    // CreateService
}
UninstallService()
{
    // OpenSCManager
    // DeleteService
}
Filename Creation

- Log files, EXE’s, DLL’s
- Subdirectories
- Environment Variables
- Random numbers
Case Study: Chinese APT

2005 posting of similar source code, includes poster’s handle.
Case Study: Chinese APT

Continued searching will reveal many, many references to the base source code of this malware.

All malware samples for this attacker are derived from this basic framework, but many additions & modifications have been made.
3rd Party SourceCode

- Core ‘Backbone’ Sourcecode
- Tweaks & Mods
- 3rd party Sourcecode
- 3rd party libraries

Compiler
- Time
- Paths
- MAC address

Runtime Libraries

Machine

Developer

Sample
- Malware
- Packing

© 2010 HBGary, Inc. All Rights Reserved
**Format Strings**

- These are written by humans, so they provide good uniqueness

```plaintext
http://%s:%d/%d%04d
```

| 00 6D 73 65 77 6D 76 00 | %s\%s.%s.msewmv.
| 6C 6C 61 2F 34 2E 30 20 | 200.Mozilla/4.0
| 62 6C 65 3B 20 4D 53 49 | (comPatIble: MSI
| 69 6E 64 6F 77 73 20 4E | E 9.0; Windows N
| 4E 45 54 20 43 4C 52 20 | T 8.0; .NET CLR
| 29 00 57 54 68 74 74 70 | 1.1.4322).WHttp
| 2F 25 64 25 30 34 64 00 | ://%s:%d/%d%04d.
| 64 61 74 00 44 65 66 61 | %s\%05d.dat.Def
| 74 61 31 00 50 72 6F 65 | alt.WinStud.Pro
| 0D 0A 25 73 20 25 73 0D | penc0127_%s_%d
| 64 2D 25 30 32 64 2D 2D | ...[%04d-%02d-%
| 3A 25 30 32 64 3A 25 30 | 02d %02d:%02d:%0
| 5B 46 31 31 5D 00 00 00 | 2d].hko.[F11]
| 5B 46 31 32 5D 00 00 00 | [F9]...[F12]
| 5B 46 38 5D 00 00 00 00 | [F10]...[F8]
| 5B 46 37 5D 00 00 00 00 | [F5]...[F7]
| 5B 46 34 5D 00 00 00 00 | [F6]. [F4]
```
Searching for:
- “Unable to determine” &
- “Unknown type!”

Reveals that the attacker is using the source-code of BO2k for cut-and-paste material.
```c
81: char *sRplmeminfo;
82: char *sRplerrdisk;
83: char *sRpldiskrmv;

// Reply: "Memory: %D in use: %D Page file: %D free: %D"

// Reply: "Unable to determine."

// Reply: "Removable"

87: char *sRpldiskram;
88: char *sRpldiskuk;
89: char *sRpldiskinfo;

// Reply: "Ramdisk"

// Reply: "Unknown type!"

// Reply: "Bytes free: $u MB($s)/$u MB($s)"

```
Mutex Names

Mutex names remain consistent at least for one infection-push, as they are designed to prevent multiple-infections for the same malware.
Link Analysis

void WriteChar(char* sText)
{
    //加锁
    HANDLE hMutex = OpenMutex(MUTEX_ALL_ACCESS, FALSE, "PsKey400");
    if(hMutex != NULL)
        WaitForSingleObject(hMutex, 300);

    FILE fp;
    if ((fp = &fopen(m_CharFileName,"ab")) == NULL)
    {
        MessageBox(NULL,"打开了出错","打开了出错",MB_OK);
        fclose(&fp);
    }
    if (fwrite(sText,strlen(sText),1,&fp) != 1)
    {
        MessageBox(NULL,"写入出错","写入出错",MB_OK);
        fclose(&fp);
    }
    fclose(&fp);
3rd Party Libraries

Developer

Core ‘Backbone’ Sourcecode
Tweaks & Mods
3rd party Sourcecode
3rd party libraries

Compiler

Runtime Libraries

Time
Paths
MAC address

Machine

Sample

Malware
Packing

© 2010 HBGary, Inc. All Rights Reserved
Copyright & Version Strings

OpenSSL/0.9.6
RAND part of OpenSSL 0.9.8e 23 Feb 2007
MD5 part of OpenSSL 0.9.8k 25 Mar 2009
libdes part of OpenSSL 0.9.7b 10 Apr 2003
inflate 1.2.1 Copyright 1995-2003 Mark Adler
inflate 1.1.4 Copyright 1995-2002 Mark Adler
inflate 1.2.3 Copyright 1995-2005 Mark Adler
inflate 1.0.4 Copyright 1995-1996 Mark Adler
inflate 1.1.3 Copyright 1995-1998 Mark Adler
inflate 1.1.2 Copyright 1995-1998 Mark Adler
inflate 1.2.2 Copyright 1995-2004 Mark Adler
zlib Fingerprinting

• Every new version of zlib has a unique pattern of bits in the data tables – these are modified for each version specifically

• This pattern is a data constant and can be used even if the copyright notices have been removed

http://www.enyo.de/fw/security/zlib-fingerprint/zlib.db
inflate library patterns

• Not as specific as zlib patterns but can be used to detect the inflate decompressor

http://www.enyo.de/fw/security/zlib-fingerprint/inflate.db
Installation & Deployment

Developer

Communications Functions

Installation & Deployment Method

Command & Control Functions

Compiler Environment

Stealth & Antiforensic Techniques

Sample

Malware

Packing
Case Study: Chinese APT

Alters the DLL value of an existing service named “RemoteRegistry”:
- Original ServiceDll value: regsvc.dll
- Trojan ServiceDll value: regsvr.dll

Registers a service named “IPRIP” which operates as a DLL loaded under svchost.exe

 Registers a service named “IPRIP” which operates as a DLL loaded under svchost.exe
Command & Control

- Communications Functions
- Installation & Deployment Method
- Compiler Environment
- Stealth & Antiforensic Techniques

Sample
- Malware
- Packing
Command and Control

Once installed, the malware phones home...

| TIMESTAMP | SOURCE COMPUTER USERNAME | VICTIM IP | ADMIN? | OS VERSION | HD SERIAL NUMBER |
C&C Hello Message

1) this queries the uptime of the machine..
2) checks whether it's a laptop or desktop machine...
3) enumerates all the drives attached to the system, including USB and network...
4) gets the windows username and computername...
5) gets the CPU info... and finally,
6) the version and build number of windows.
Command and Control Server

- The C&C system may vary
  - Custom protocol (Aurora-like)
  - Plain Old URL’s
  - IRC (not so common anymore)
  - Stealth / embedded in legitimate traffic

- Machine identification
  - Stored infections in a back end SQL database
A) Command is stored as a number, not text. It is checked here.

B) Each individual command handler is clearly visible below the numerical check

C) After the command handler processes the command, the result is sent back to the C&C server
Advanced Fingerprinting
GhostNet: Screen Capture Algorithm

Loops, scanning every 50th line (cY) of the display.

Reads screenshot data, creates a special DIFF buffer

LOOP: Compare new screenshot to previous, 4 bytes at a time

If they differ, enter secondary loop here, writing a ‘data run’ for as long as there is no match.
GhostNet: Searching for sourcecode

Large grouping of constants

Search source code of the ‘Net

Search public source code.
Further refine the search by including ‘WAVE_FORMAT_GSM610’ in the search requirements...
We discover a nearly perfect ‘c’ representation of the disassembled function. Clearly cut-and-paste.

We can assume most of the audio functions are this implementation of ‘CAudio’ class – no need for any further low-level RE work.
On link analysis...
1. Implant
2. Forensic
   Toolmark specific to Implant
3. Searching the ‘Net reveals source code that leads to Actor
4. Actor is supplying a backdoor
5. Group of people asking for technical support on their copies of the backdoor

Example: Link Analysis with Palantir™
Working back the timeline

• Who sells it, when did that capability first emerge?
  – Requires ongoing monitoring of all open-source intelligence, presence within underground marketplaces
  – Requires budget for acquisition of emerging malware products
Conclusion
Takeaways

• Actionable intelligence can be obtained from malware infections for immediate defense:
  – File, Registry, and IP/URL information
• Existing security doesn’t stop ‘bad guys’
  – Go ‘beyond the checkbox’
• Adversaries have intent and funding
  – Failure is hiccup – doesn’t stop mission
• Need to focus on the criminal, not malware
  – Attribution is possible thru forensic toolmarking combined with open and closed source intelligence
Continued Work

• Will be presenting additional research at BlackHat Vegas this year
  – Trend over 500k malware samples
• HBGary will be releasing a free tool that will dump fingerprint information from a binary or livebin
Fingerprint Utility

Developer Fingerprint Utility, Copyright 2010 HBGary, INC
File: 1228ad2e39befa4319733e98d8ed2890.livebin

<table>
<thead>
<tr>
<th>Original project name:</th>
<th>RESSDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer's project directory:</td>
<td>e:\gh0st\server\sys\i386</td>
</tr>
<tr>
<td>Compiler:</td>
<td>Microsoft Visual C++ 6.0 release</td>
</tr>
<tr>
<td>User interface:</td>
<td>Windows GDI/Common Controls</td>
</tr>
<tr>
<td>Media:</td>
<td>Windows multimedia API</td>
</tr>
<tr>
<td>Media:</td>
<td>Microsoft VfW (Video for Windows)</td>
</tr>
<tr>
<td>Compression:</td>
<td>Inflate Library version: 1.1.4</td>
</tr>
<tr>
<td>Networking:</td>
<td>Windows sockets (TCP/IP)</td>
</tr>
<tr>
<td>Networking:</td>
<td>Windows Internet API</td>
</tr>
<tr>
<td>Source directory:</td>
<td>e:\gh0st\server\sys\i386</td>
</tr>
</tbody>
</table>
Thank You

- HBGary, Inc. (www.hbgary.com)
- HBGary Federal (www.hbgaryfederal.com)