Winnti Polymorphism

Takahiro Haruyama

Symantec

Who am I?

- Takahiro Haruyama (@cci_forensics)
- Reverse Engineer at Symantec
  - Managed Adversary and Threat Intelligence (MATI)
    - https://www.symantec.com/services/cyber-security-services/deepsight-intelligence/adversary
- Speaker
  - BlackHat Briefings USA/EU/Asia, SANS DFIR Summit, CEIC, DFRWS EU, SECURE, FIRST, RSA Conference JP, etc…

Motivation

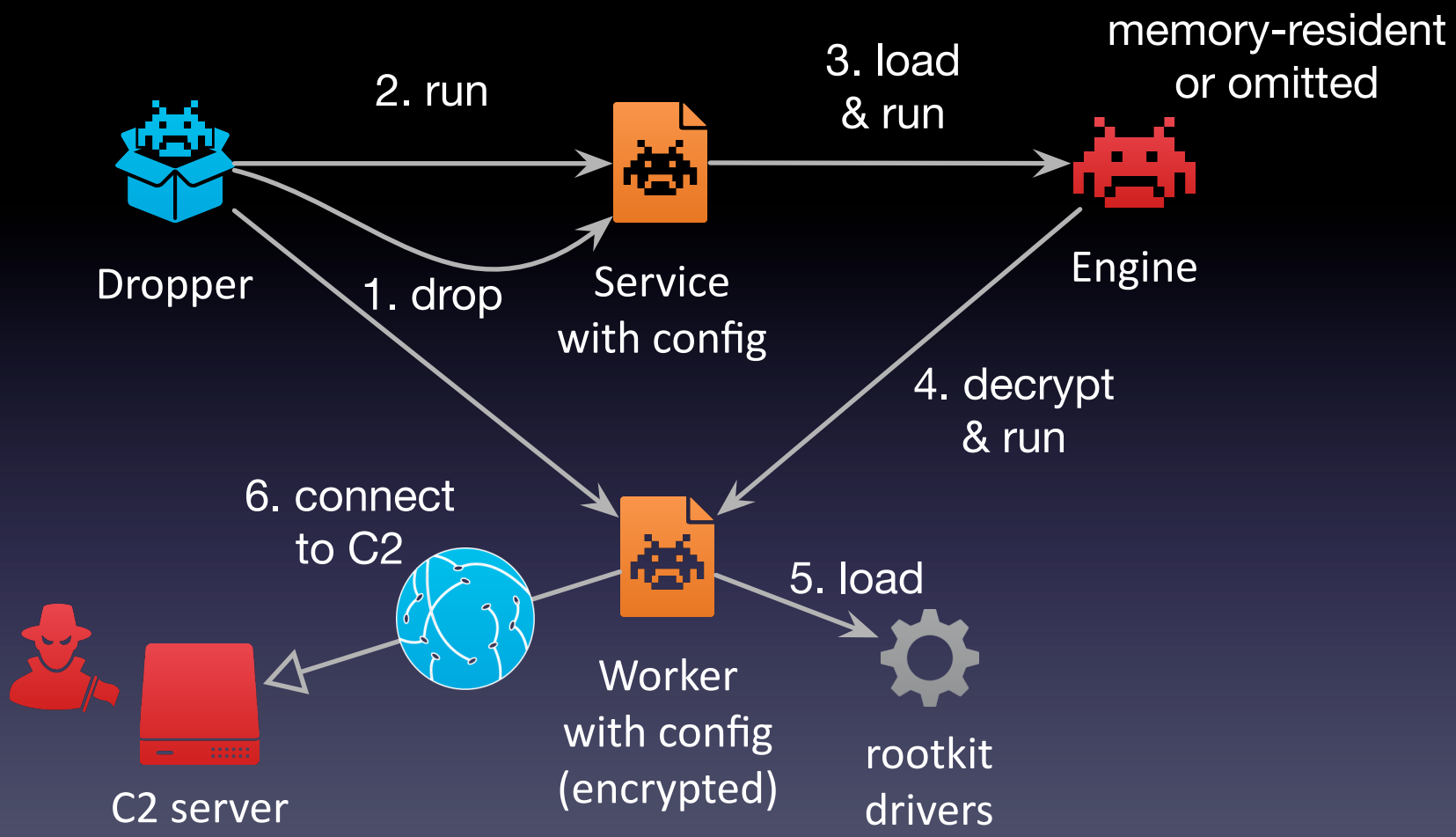
- Winnti is malware used by Chinese threat actor for cybercrime and cyber espionage since 2009
- Kaspersky and Novetta published good white papers about Winnti [1] [2]
- Winnti is still active and changing
  - Variants whose behavior is different from past reports
  - Targets except game and pharmaceutical industries
- I'd like to fill the gaps

Agenda

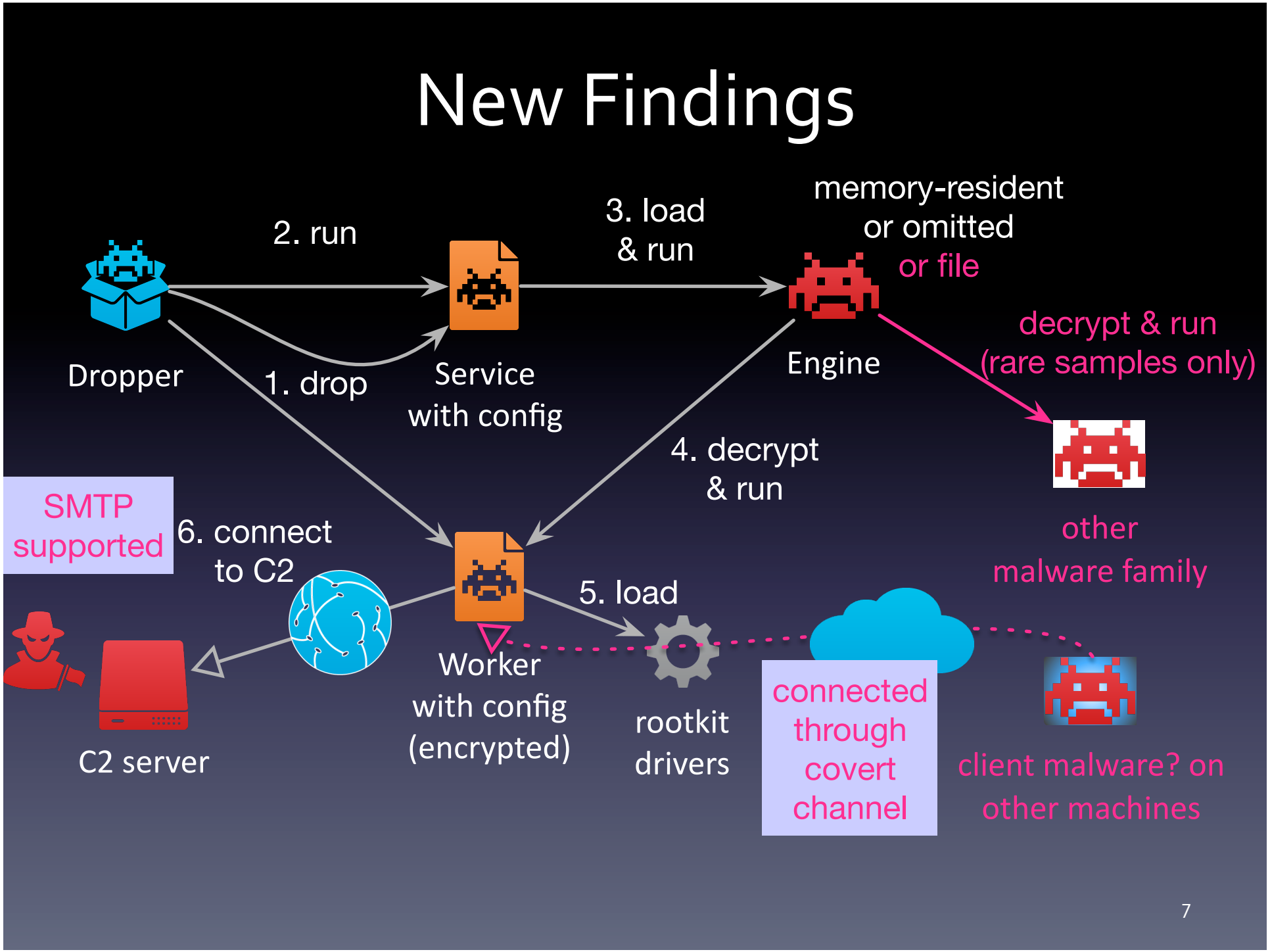
- Winnti Components and Binaries
- Getting Target Information from Winnti Samples
- Wrap-up

Winnti Components and Binaries

Winnti Execution Flow



New Findings



Dropper Component

- extract other components from inline DES-protected blob
  - the dropped components are
    - service and worker
    - **additionally engine with other malware family (but that is rare)**
  - the password is passed from command line argument
  - Some samples add dropper's configuration into the overlays of the components
- run service component
  - /rundll32.exe "%s", \w+ %s/
  - **the export function name often changes**
    - Install, DlgProc, gzopen_r, Init, sql_init, sqlite3_backup_deinit, etc…

Service Component

- load engine component from inline blob
  - the values in PE header are eliminated
    - e.g., MZ/PE signatures, machine architecture, NumberOfRvaAndSizes, etc…
- call engine's export functions
  - **some variants use the API hashes**
    - e.g., 0x0C148B03 = "Install", 0x3013465F = "DeleteF"

```
def calculate_hash(name):
    n = [ord(x) for x in name]
    h = 0
    for i in range(len(n)):
        h = n[i] + 131 * h
    return h & 0x7FFFFFFF
```

Engine Component

- memory-resident
  - **some samples are saved as files with the same encryption of worker component**
- export function names
  - Install, DeleteF, and Workmain
- try to bypass UAC dialog then create service
- decrypt/run worker component
  - PE header values eliminated, 1 byte xor & nibble swap

Worker Component

- export function names
  - work_start, work_end
- plugin management
  - the plugins are cached on disk or memory-resident
- supported C2 protocols
  - TCP = header + LZMA-compressed payload
  - HTTP, HTTPS = zlib-compressed payload as POST data
  - **SMTP**

SMTP Worker Component

- Some worker components support SMTP
  - the config contains email addresses and more obfuscated (incremental xor + dword xor)
- Public code is reused
  - The old code looks copied from PRC-based Mandarin-language programming and code sharing forum [3]
    - The hard-coded sender email and password are "attach_111@sina.com" and "test123456"
  - The new code looks similar to the one distributed in Code Project [4]
    - STARTTLS is newly supported to encrypt the SMTP traffic

SMTP Worker Component (Cont.)

```
struct struct_config_part1
{
  int field_0_xor_key;
  int field_4_imm1;
  SYSTEMTIME field_8_timestamp;
  int field_18_immFh;
  int field_1C_imm1;
  int field_20_imm0;
  char field_24_id?[64]; // xx
  char field_64_sender_QQMailID[64]; // 827762398
  char field_A4_sender_password[64]; // zkxgowarprrwbdjg
  char field_E4_working_folder[256]; // c:\wen
  struct_recipient_emails field_1E4_recipient_emails;
  int field_6E8_fn_check_explorer_process;
  int field_6EC;
};

struct struct_recipient_emails
{
  __int16 field_1E4_null;
  __int16 field_1E6_num_of_recipients; // 2
  char field_1E8_recipient_email[256]; // testattach126@126.com
  char field_2E8_recipient_email[256]; // attach_111@sina.com
  char field_3E8_blob1[760];
  int field_6E0;
  int field_6E4;
};
```

for decrypting each member

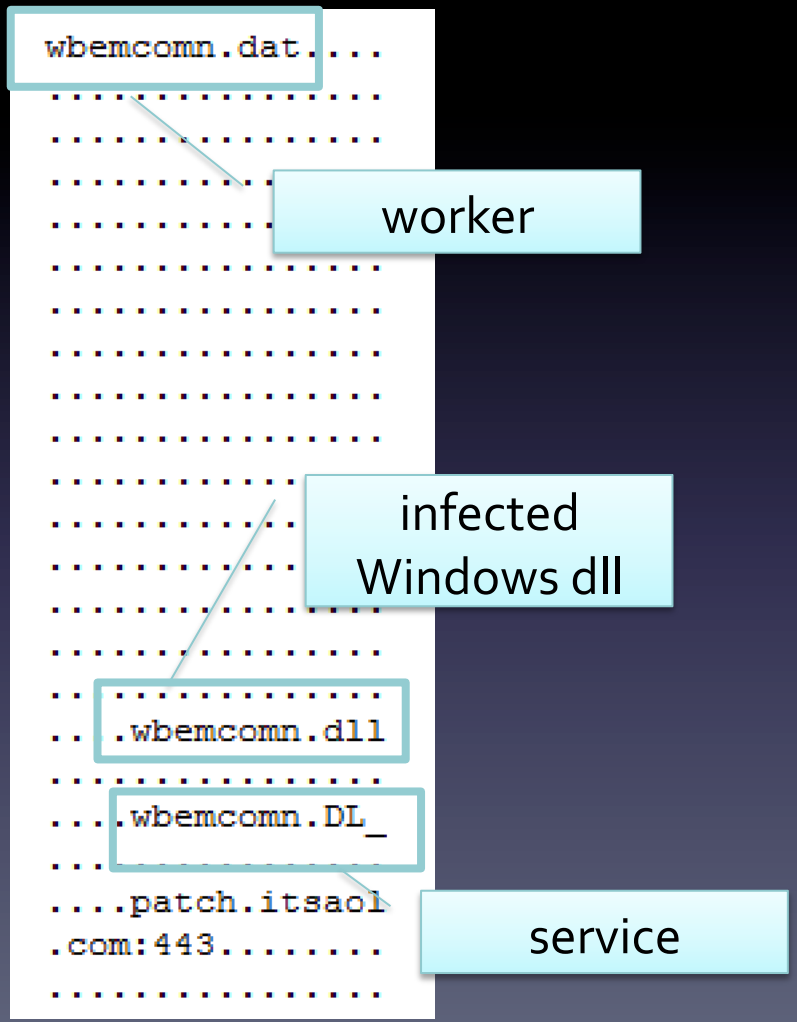
QQMail [5] account is used for sending

recipient email addresses

VSEC Variant [6]

- Two main differences compared with Novetta variant [2]
  - no engine component
    - service component directly calls worker component
  - worker's export function name is "DllUnregisterServer"
    - takes immediate values according to the functions
      - e.g., 0x201401 = delete file, 0x201402 = dll/code injection, 0x201404 = run inline main DLL
- recently more active than Novetta variant?

VSEC Variant (Cont.)



- unique persistence
  - Some samples modify IAT of legitimate windows dlls to load service component
  - the target dll name is included in the configuration
    - e.g., wbemcomn.dll, loadperf.dll

Winnti as a Loader

```
struct XSetting
{
  XHeader field_0_xheader;
  int field_8_flags?;
  int field_C_timer_connection_interval;
  int field_10_timer_sleep?;
  char field_14_active_time_table[672];
  int field_2B4_customDNS1;
  int field_2B8_customDNS2;
  int field_2BC_customDNS3;
  int field_2C0_customDNS4;
  C2Setting field_2C4_C2_hostname1;
  C2Setting field_308_C2_hostname2;
  C2Setting field_34C_C2_hostname3;
  C2Setting field_390_C2_hostname4;
  char field_3D4_C2Setting_URL1[128];
  char field_454_C2Setting_URL2[128];
  char field_4D4_C2Setting_URL3[128];
  char field_554_C2Setting_URL4[128];
  struc_ProxySettings field_5D4_proxySetting1;
  struc_ProxySettings field_698_proxySetting2;
  struc_ProxySettings field_75C_proxySetting3;
  struc_ProxySettings field_820_proxySetting4;
  __int16 field_8E4_install_folder_path[256];
  char field_AE4_winnti_service_comp_name[32]; // ne
  char field_B04_winnti_engine_comp_name[32];  // ne
  char field_B24_http_location[256]; // new, "Http L
  char field_C24_network_config_and_location[256];
ntiguration and location information, and notities a
```

Some engine components embeds other malware family like Ghost and PlugX

- the configuration is encrypted by Winnti and the malware algorithm
- the config members are the malware specific + Winnti strings

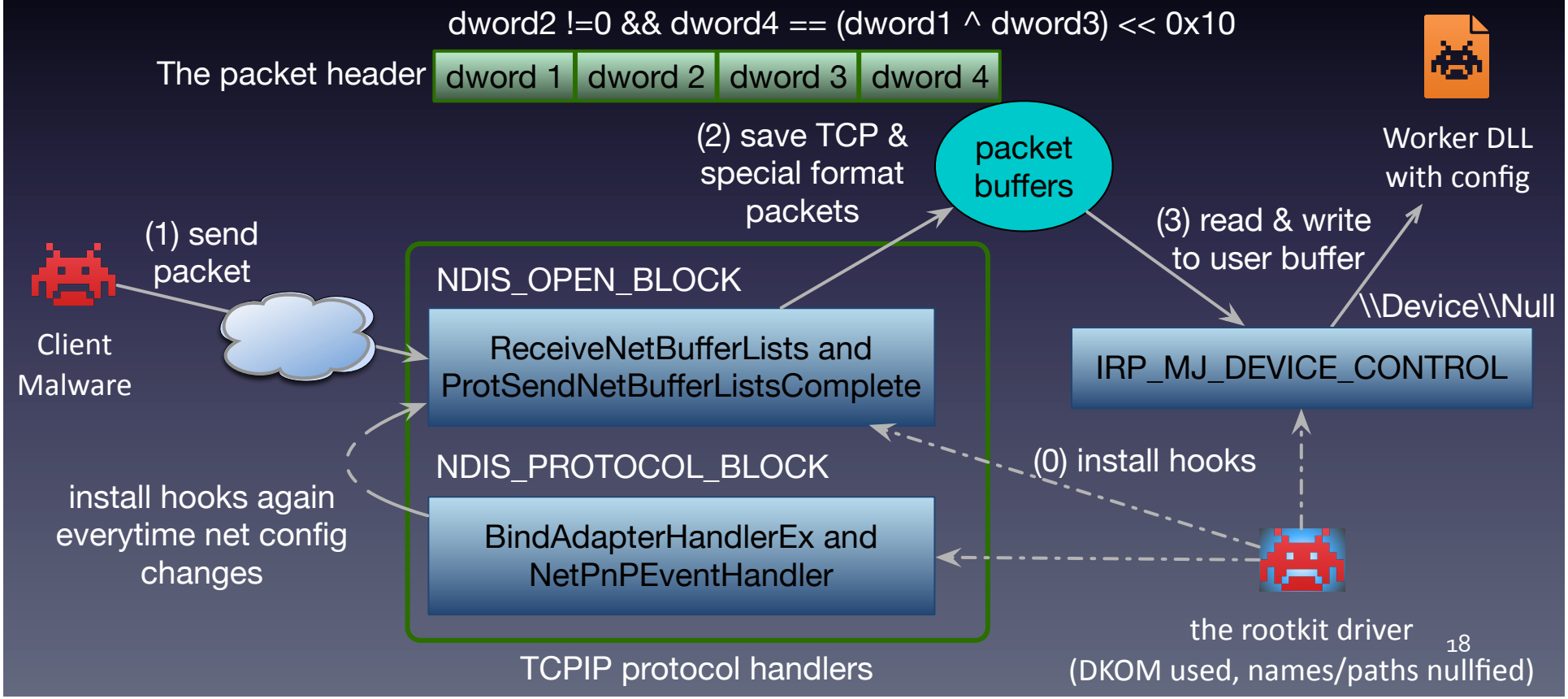
Winnti-related members

Related Kernel Drivers

- Kernel rootkit drivers are included in worker components
  - hiding TCP connections
    - The same driver is also used by Derusbi [7]
  - **making covert channels** with other client machines
    - The behavior is similar to WFP callout driver of Derusbi server variant [8] but the implementation is different

Related Kernel Drivers (Cont.)

- The rootkit hooks TCPIP Network Device Interface Specification (NDIS) protocol handlers
  - intercepts incoming TCP packets then forward to worker DLL



Related Attack Tools

- bootkit found by Kaspersky when tracking Winnti activity [9]
- "skeleton key" to patch on a victim's AD domain controllers [10]
- custom password dump tool (exe or dll)
  - Some samples are protected by VMProtect or unique xor or AES
  - the same API hash calculation algorithm used (function name = "main_exp")

```
def decrypt(enc):
    dec = [ord(x) for x in enc]
    key = dec[0]
    for i in range(1, len(dec)):
        tmp = (key + i) & 0xff
        dec[i] = (((tmp ^ dec[i]) >> 4) + ((tmp ^ dec[i]) << 4)) & 0xff
    dec = [chr(x) for x in dec]
    return "".join(dec)
```

- PE loader
  - decrypt and run a file specified by the command line argument
    - *((_BYTE *)buf_for_cmdline_file + offset) ^= 7 * offset + 90;

includes two drivers compiled on August 22 and September 4, 2014. The sample has an encrypted configuration block placed in overlay. This block may include a tag for the sample – usually it is a campaign ID or victim ID/name. This time the operators put such tag in the configuration and it turned out to be the name of the **well-known global pharmaceutical company headquartered in Europe**:

One of the mentioned drivers (a known, malicious Winnti network rootkit) was **signed with a stolen certificate of a division of a huge Japanese conglomerate**. Although this division is involved in microelectronics manufacturing, other business directions of the conglomerate include **development and production of drugs as well as medical equipment**.

from Kaspersky blog [11]

Getting Target Information from Winnti Samples

Two Sources about the Targets

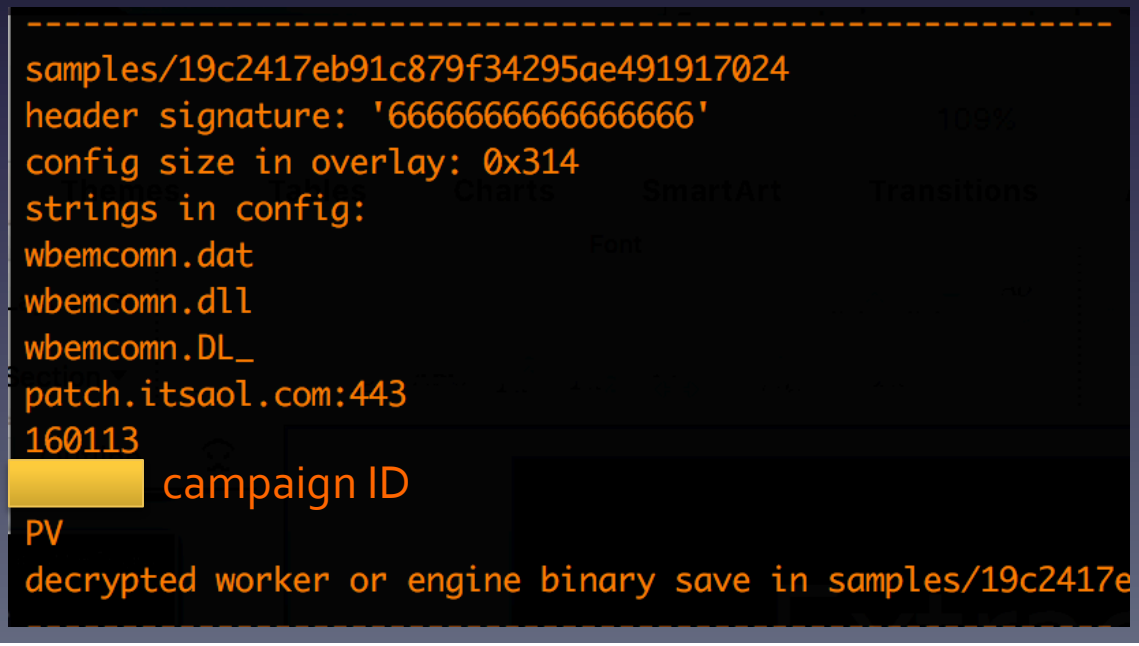
- campaign ID from configuration data
  - target organization/country name
- stolen certificate from rootkit drivers
  - already-compromised target name
- I checked over 170 Winnti samples
  - Which industry is targeted by the actor, except game and pharma ones?

Extraction Strategy

- regularly collect samples from VT/Symc by using detection name or yara rules
- try to crack the DES password if the sample is dropper component
  - or just decrypt the config if possible
- run config/worker decoder for service/worker components
  - campaign IDs are included in worker rather than service
- extract drivers from worker components then check the certificates
- exclude the following information
  - not identifiable campaign ID (e.g., "a1031066", "taka1100")
  - already-known information by public blogs/papers

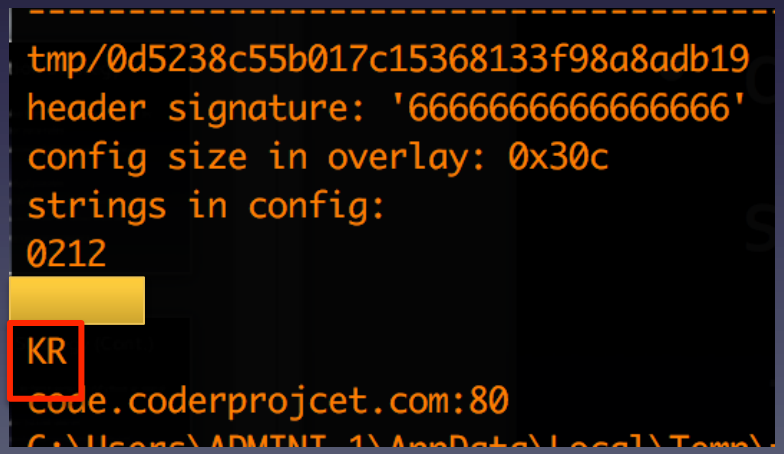
Extraction Strategy (Cont.)

- automation
  - config/worker decoder (stand-alone)
    - decrypt config data and worker component if detected
    - additionally decrypt for PlugX loader or SMTP worker variants
  - dropper password brute force script (IDAPython or stand-alone)

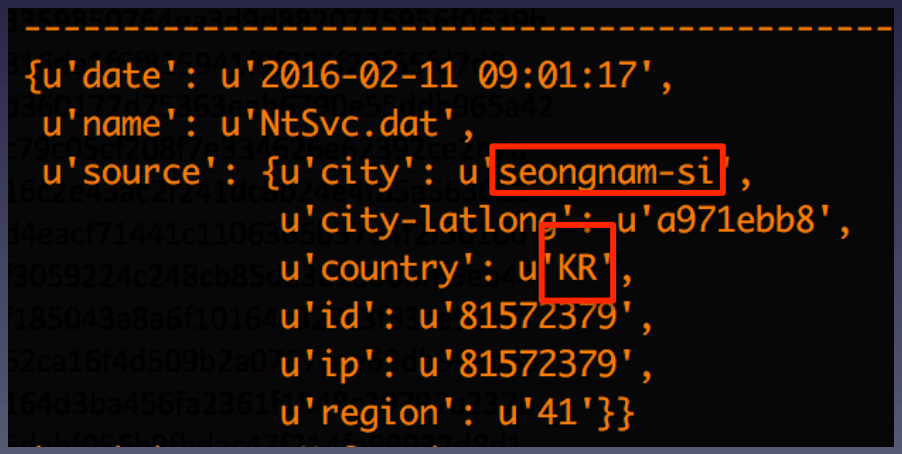


Extraction Strategy (Cont.)

- double-check campaign IDs by using VT submission metadata
  - the company has its HQ or branch office in the submitted country/ city?
- e.g., the ID means 2 possible companies in different industries
  - The submission city helps to identify the company



decrypted config



VT submission metadata

Result about Campaign ID

- only 27 % samples contained configs ☹
  - Most of them are service components
    - service components usually contains just path information
  - difficult to collect dropper/worker components by detection name
    - Yara retro-hunt can search samples within only 3 weeks
- 19 unique campaign IDs found
  - 12 IDs were identifiable and not open

Result about Campaign ID (Cont.)

1st seen year from VT metadata	submission country / city from VT metadata	Industry
2014	Russia / Moscow	Internet Information Provider? (typo)
2015	China / Shenzhen	University? (not sure)
2015	South Korea / Seongnam-si	Game
2015	South Korea / Seongnam-si	Game
2015	South Korea / Seongnam-si	Game
2016	Japan / Chiyoda	Chemicals
2016	Vietnam / Hanoi	Internet Information Provider, E-commerce, Game
2016	South Korea / Seoul	Investment Management Firm
2016	South Korea / Seongnam-si	Anti-Virus Software
2016	USA / Bellevue	Game
2016	Australia / Adelaide	IT, Electronics
2016	USA / Milpitas	Telecommunications

Result about Certificate

- 12 unique certificates found but most of them are known in [1] [12]
- 4 certificates are not open
  - One of them is signed by an electronics company in Taiwan
  - The others are certificates of chinese companies
    - "Guangxi Nanning Shengtai'an E-Business Development CO.LTD", "BEIJING KUNLUN ONLINE NETWORK TECH CO.,LTD", "成都优昂文化传播有限责任公司"
  - I'm not sure if they were stolen or not
    - One is a primary distributor of unwanted software? [13]

Wrap-up

Wrap-up

- Winnti malware is polymorphic, but
  - The variants and tools have common codes
    - e.g., config/binary encryption, API hash calculation
  - Some driver implementations are identical or similar to Derusbi's ones
- Today Winnti threat actor(s?) targets at chemical, e-commerce, investment management firm, electronics and telecommunications companies
  - Game companies are still targeted
- Symantec telemetry shows they are just a little bit of targets ☹

Reference

1. http://kasperskycontenthub.com/wp-content/uploads/sites/43/vlpdfs/winnti-more-than-just-a-game-130410.pdf
2. https://www.novetta.com/wp-content/uploads/2015/04/novetta_winntianalysis.pdf
3. http://blog.csdn.net/lishuhuakai/article/details/27852009
4. http://www.codeproject.com/Articles/28806/SMTP-Client
5. https://en.mail.qq.com/
6. http://blog.vsec.com.vn/apt/initial-winnti-analysis-against-vietnam-game-company.html
7. https://assets.documentcloud.org/documents/2084641/crowdstrike-deep-panda-report.pdf
8. https://www.novetta.com/wp-content/uploads/2014/11/Derusbi.pdf
9. https://securelist.com/analysis/publications/72275/i-am-hdroot-part-1/
10. https://www.symantec.com/connect/blogs/backdoorwinnti-attackers-have-skeleton-their-closet
11. https://securelist.com/blog/incidents/70991/games-are-over/
12. http://blog.airbuscybersecurity.com/post/2015/11/Newcomers-in-the-Derusbi-family
13. https://www.herdprotect.com/signer-guangxi-nanning-shengtaian-e-business-development-coltd-1eb0f4d821e239ba81b3d10e61b7615b.aspx