# Intro to Reverse Engineering and Malware Analysis


**Jake Smith**
University of Virginia

**Credits:**
@MalwareUnicorn

# Cybersecurity News Segment

# Agenda

- Overview
  - What
  - Why/Context
  - How
- Basic Analysis
  - File, Strings, VirusTotal
- Static Analysis
  - HxD, IDA
- Detection / YARA

# Malware Types (What)


## Virus

- “Classic” malware, runs malicious code
- User action required

## Worm

- Self-propagating malware (ie exploit vuln, etc)
- Example: NotPetya

## Trojan

- Pretends to be legitimate software
- Example: Phone App that also steals your info

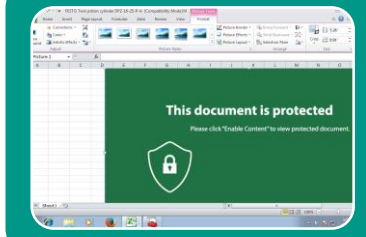
# Malware Types (What)


## Ransomware

- Encrypts all files and demands ransom
- Example: WannaCry, (Not)Petya, TeslaCrypt


## RAT/Backdoor

- Allows an attacker to have remote access to machine
- Example: Dark Comet




## Dropper

- “Initial” stage of malware
- Downloads malicious Stage 2, and executes it

# Malware Goals (Why)

## Data

- Company IP
- Personally Identifiable Information (PII)

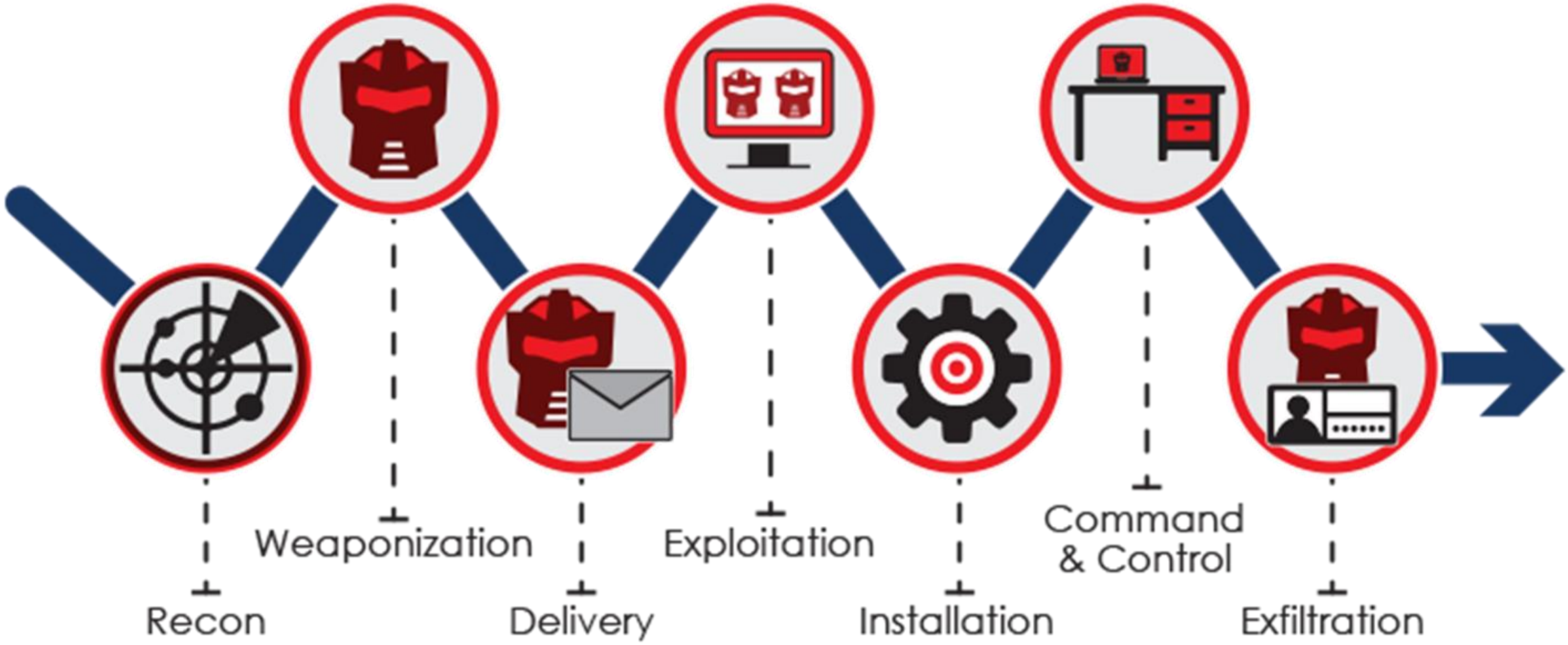
## Money

- Cryptocoins!
- Financial Info

## Damage

- Destroy Facilities
- Cause Harm

# Delivery and Techniques (How)




https://www.eventtracker.com/tech-articles/siemphonic-cyber-kill-chain/

# Delivery and Techniques (How)

- Obfuscation

```
#define NO_IDENT /* let's define it*/
#define MYMIN( x, y)  ( ( x) > ( y) ? ( y) : ( x) )  /* complex macros OK*/

#define HAVE_TSEARCH
int name_wide, verbose, max_width= 80;
int
main( int argc,  char * argv[ ] )
{
  int j;
  char version[ 80] ;
  while ( ( j =  getopt_helper( argc,  argv,  "n:o:vV:",  ((char)(0x2053+885-0x2360)),
 ((char)(0x2368+457-0x24db))) != -1) {
```

https://sites.google.com/secured.org/malwareunicorn/reverse-engineering/re101/section-2?authuser=0

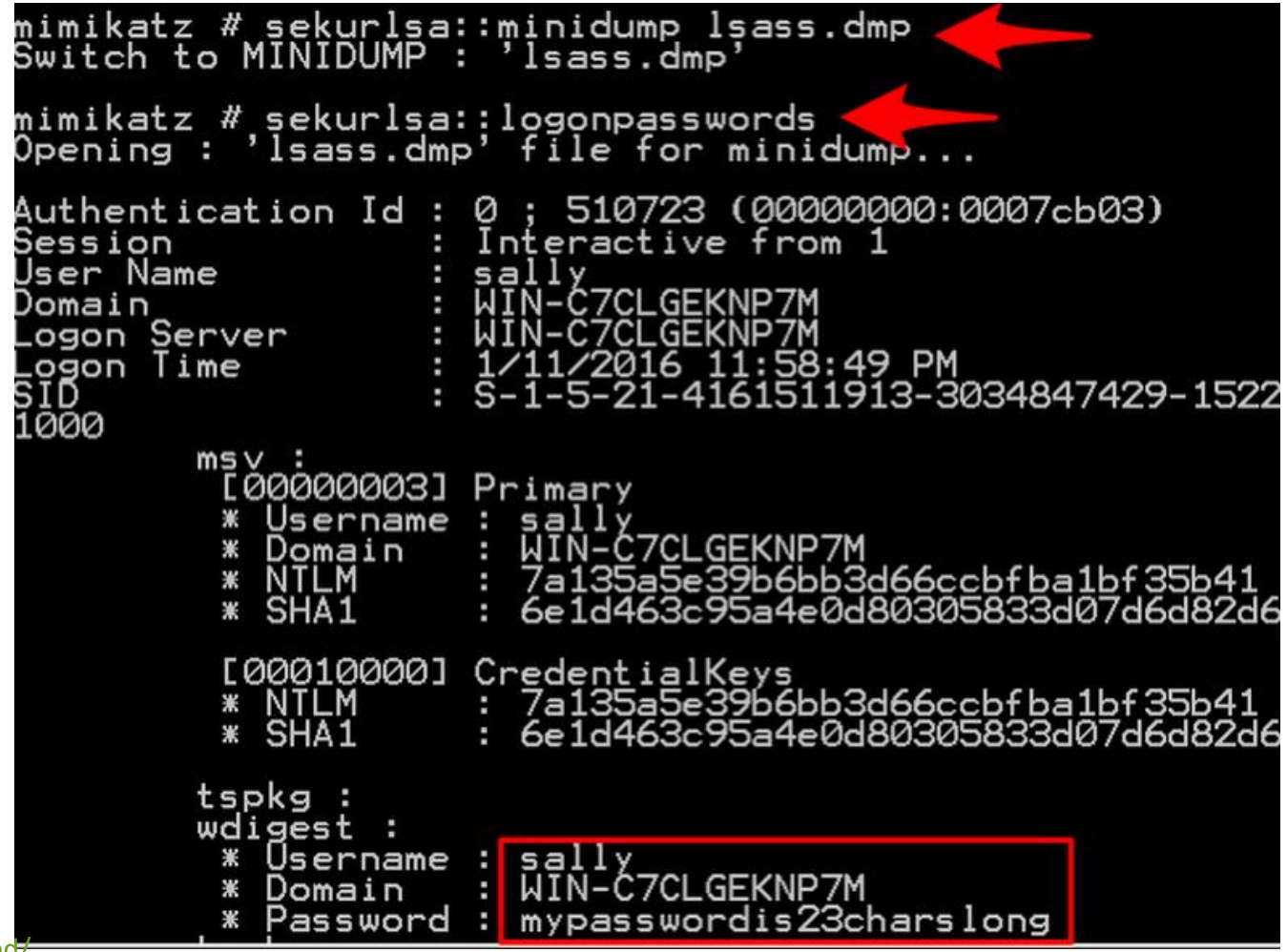
# Delivery and Techniques (How)

- Persistence


https://sites.google.com/secured.org/malwareunicorn/reverse-engineering/re101/section-2?authuser=0

# Delivery and Techniques (How)

- Credential Theft




https://www.blackhillsinfosec.com/your-password-is-wait-for-it-not-always-encrypted/

# Basic Analysis

# File Command

- Looks at “magic bytes” - first few bytes of file
- Compares byte sequence to see what type of file it is
- ELF = Executable and Linking Format
- Executable/ELF file:
  - Magic: 7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
- Syntax: `file <filename>`

# Note: Windows Subsystem

- Allows you to run Ubuntu and other linux without needing a full VM
- Access your files at:
  - `cd /mnt/c/Users/<you>/`
- Follow instructions at https://tinyurl.com/installwsl to install

# File Practice

- `wget problems.metactf.com/cns/file1`
- `file file1`
- Try downloading file2 and file3 to see what kind of files they are

# Strings Command

- Outputs all strings in the program/file
- Useful to see what you can deduce about program / its contents
- Syntax: `strings <filename>`

# Strings Practice

- Try running strings on the files from before
- You can use the –n flag to only output strings longer than a certain length,
  - Ex: `strings file2 -n 6`

# VirusTotal

- The “Google for Malware”
- Scans files with 60+ Anti-Virus (AV) providers
- Performs static and dynamic analysis
  - Static: Looking at metadata, properties, NO EXECUTE
  - Dynamic: Execute malware in sandbox and watch

# VirusTotal Examples

- Visit https://tinyurl.com/vtsample1 and or https://tinyurl.com/vtsample2

# Advanced Examination

# HxD / Hex Editor

- Enables you to view raw bytes of file
- Useful to check/edit magic bytes
- Can also be used to search for specific sequence of bytes

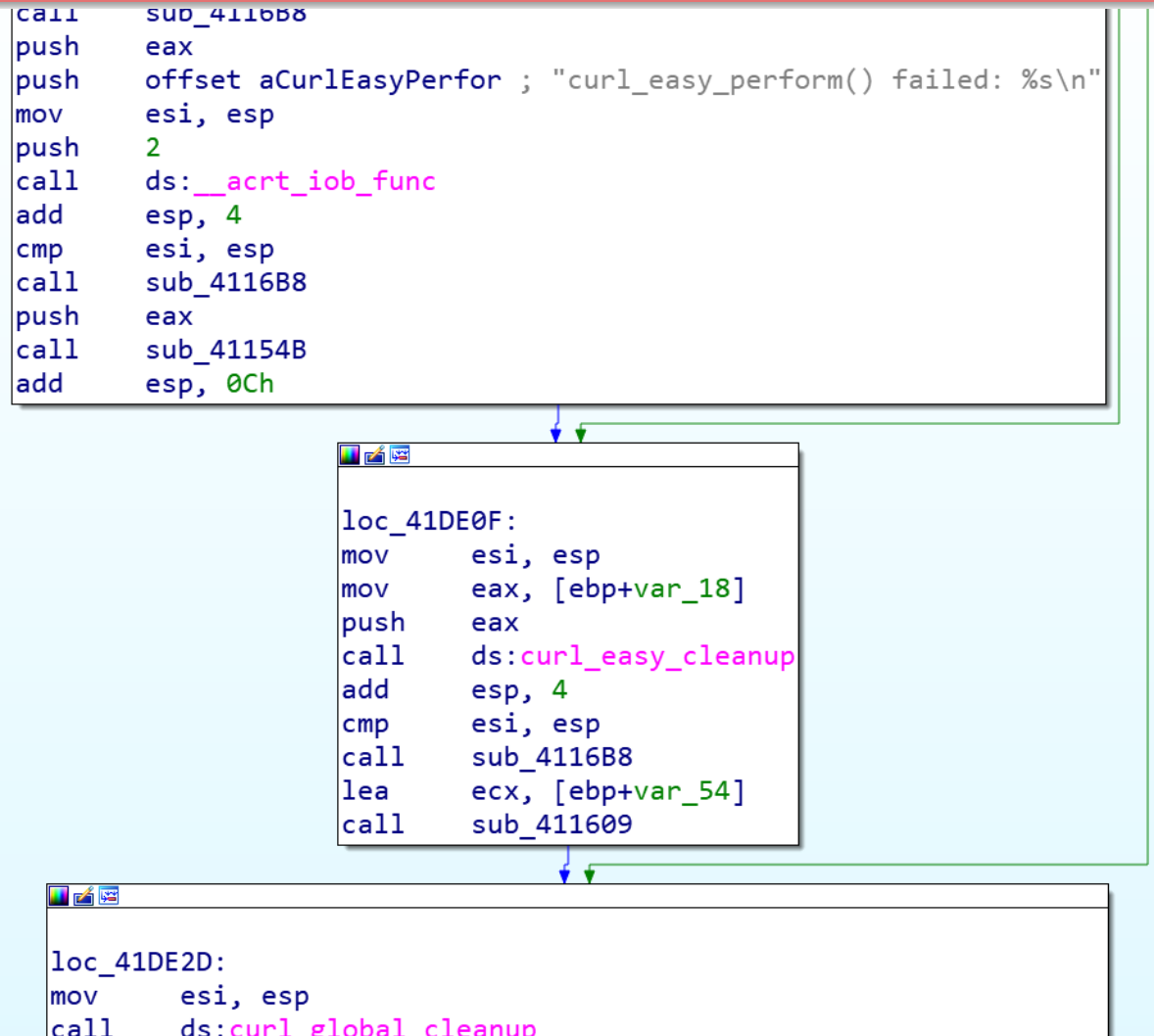
```
rule CloudDuke_Malware {
	meta:
		description = "Detects CloudDuke Malware"
		license = "https://creativecommons.org/licenses/by-nc/4.0/"
		author = "Florian Roth"
		reference = "https://www.f-secure.com/weblog/archives/00002822.html"
		date = "2015-07-22"
		score = 60
		hash1 = "97d8725e39d263ed21856477ed09738755134b5c0d0b9ae86ebb1cdd4cdc18b7"
		hash2 = "88a40d5b679bccf9641009514b3d18b09e68b609ffaf414574a6eca6536e8b8f"
		hash3 = "1d4ac97d43fab1d464017abb5d57a6b4601f99eaa93b01443427ef25ae5127f7"
		hash4 = "ed7abf93963395ce9c9cba83a864acb4ed5b6e57fd9a6153f0248b8ccc4fdb46"
		hash5 = "ee5eb9d57c3611e91a27bb1fc2d0aaa6bbfa6c69ab16e65e7123c7c49d46f145"
		hash6 = "a713982d04d2048a575912a5fc37c93091619becd5b21e96f049890435940004"
		hash7 = "56ac764b81eb216ebed5a5ad38e703805ba3e1ca7d63501ba60a1fb52c7ebb6e"
	strings:
		$s1 = "ProcDataWrap" fullword ascii
		$s2 = "imagehlp.dll" fullword ascii
		$s3 = "dnlibsh" fullword ascii
		$s4 = "%ws_out%ws" fullword wide
		$s5 = "Akernel32.dll" fullword wide

		$op0 = { 0f b6 80 68 0e 41 00 0b c8 c1 e1 08 0f b6 c2 8b } /* Opcode */
		$op1 = { 8b ce e8 f8 01 00 00 85 c0 74 41 83 7d f8 00 0f } /* Opcode */
		$op2 = { e8 2f a2 ff ff 83 20 00 83 c8 ff 5f 5e 5d c3 55 } /* Opcode */
	condition:
		uint16(0) == 0x5a4d and filesize < 720KB and 4 of ($s*) and 1 of ($op*)
}
```

# HxD Example

- Install HxD (or another hex editor)
  - https://mh-nexus.de/downloads/HxDSetup.zip
- Download problems.metactf.com/cns/file4.zip
- Extract file
- Open SampleBinary.exe in HxD

# IDA

- Disassembler
- Displays raw assembly code from executable
- Enables analyst to trace through specific sections of code



# IDA Example

- Install IDA Free
  - hex-rays.com/products/ida/support/download_freeware.shtml
- Open SampleBinary.exe in IDA

# Detection

# YARA

- “The pattern matching swiss knife for malware researchers”
- Create simple rules to match files on patterns, strings instead of a single hash of the whole file
- Can still detect malware even if it changes

# YARA Demo

# Questions?