



# Evading Anti-virus Detection with Metasploit

David Maloney

Metasploit Software Engineer

Rapid7

# Get CPE Credits for this Webcast

- Attendees of this Webcast are eligible for 1 CPE credit
- Self-report on your organization's website
- Keep the email invitation as confirmation for possible future audits
- More info: <http://bit.ly/R7CPE>



# Agenda

- › Payloads
- › Executable templates
  - AV detection of executive templates
- › Creating your own executable templates
- › The encoder myth
- › Manually create executable templates
- › Dynamic executable template generator
- › Live demo
- › Q&A

# Quick Glossary Before We Get Started

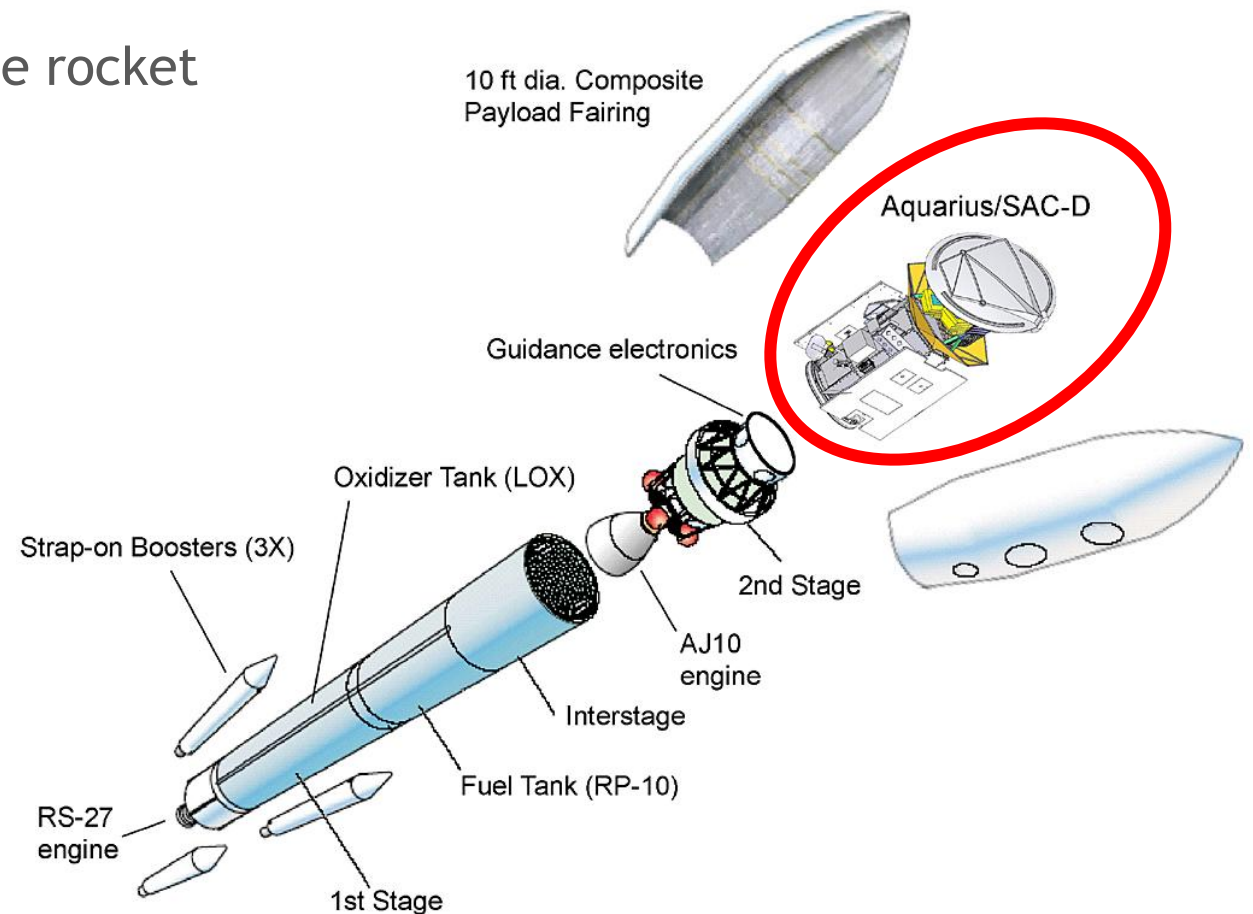
- › **Payload:** Code delivered to a victim by an exploit
- › **Signature:** Set of rules or pattern match against code
- › **Heuristics:** Matching specific behavior instead of patterns in code
- › **Sandbox:** Protected segments in OS, where code can be run safely

# Payloads

- Code delivered by the exploit
  - What you do once you're on the machine
  - Think: Satellite delivered by the space rocket

- Types of Payloads

- Singles
  - Everything at once
- Staged
  - Stage 1: Get foot in the door
  - Stage 2: Pull full payload



# The Obvious Ways

- › Turn off AV before payload executes
- › Suspend the AV processes
- › Migrate
- › Do not write executables to disk!
  - Memory corruption exploits
  - Alternative methods such as powershell

# Executable Templates

- Payloads can be injected in memory or written as a file
  - Memory injection has much lower AV detection rates
- Some types of exploits can only write files
  - Need executable template to write the payload into (e.g. use MSFVenom)
- AV vendors often just look for executable templates

# Executable Template Without Malicious Code

- No malicious payload included but 33/40 AV vendors flag the file as malicious
  - Matching is signature-based on executable template only, not the payload
- Signature detection is lazy...



SHA256: a8be7714ce60d8dba4348591efb4c7e992b39952daffa0d14dcc6327e684b209

File name: smona\_a8be7714ce60d8dba4348591efb4c7e992b39952daffa0d14dcc6327e684b209.bin

Detection ratio: 33 / 40

Analysis date: 2012-10-01 14:27:49 UTC ( 2 months ago )





# Using Different Executable Templates

- Select the psexec module and hit 'show advanced'

```
Name      : EXE::Custom
Current Setting:
Description : Use custom exe instead of automatically generating a payload exe

Name      : EXE::Fallback
Current Setting:
Description : Use the default template in case the specified one is missing

Name      : EXE::Inject
Current Setting:
Description : Set to preserve the original EXE function

Name      : EXE::OldMethod
Current Setting:
Description : Set to use the substitution EXE generation method.

Name      : EXE::Path
Current Setting:
Description : The directory in which to look for the executable template

Name      : EXE::Template
Current Setting:
Description : The executable template file name.

Name      : EnableContextEncoding
```

- Or use your own template

# The Encoder Myth

- *“AV keeps picking up my payload no matter what encoder I try.”*
- Encoders are not meant to evade AV but handle bad characters
  - With memory corruption exploits some characters may break your exploit
  - Common characters for this are space, tab, carriage return, and line feed
- Advanced encoders use a rolling XOR encoding to create polymorphic code
  - Prepend decoder stub decodes it
  - In other words, payload is not recognizable until decoded
- This has unintentionally evaded AV

# Dynamic Executable Generator

- Avoid AV detection
- Generates the C code for the executable template
- Includes the payload directly
  - Not injecting it after the template has been generated
- C code is written random & on-the-fly
- Compiled with Metasm
- Uses several techniques to try and evade AV

# Dynamic Executable Templates in Metasploit Pro

➤ Metasploit Pro offers the option to dynamically generate executable templates when using the psexec module

- Select from Advanced Options when running the PSEXec module

SMBName	<input type="text" value="*SMBSERVER"/>	The NetBIOS hostname (required for port 139 connections) (string)
SSL	<input type="checkbox"/>	Negotiate SSL for outgoing connections (bool)
SSLVersion	<input type="button" value="SSL3"/>	Specify the version of SSL that should be used (accepted: SSL2, SSL3, TLS1) (enum)
VERBOSE	<input type="checkbox"/>	Enable detailed status messages (bool)
WfsDelay	<input type="text" value="0"/>	Additional delay when waiting for a session (integer)
DynamicExe	<input type="checkbox"/>	Generate a dynamic EXE payload (bool)

**Dynamic EXE Option**

- Select from Payload Settings in the Bruteforcer when selecting SMB

**Payload Settings**

Payload Type	<input type="text" value="Meterpreter"/>
Listener Ports	<input type="text" value="1024-65535"/>
Connection Type	<input type="text" value="Auto"/>
Listener Host	<input type="text"/>
Auto Launch Macro	<input type="text"/>
<input type="checkbox"/> Dynamically generate payload EXE for SMB	

**Dynamic EXE Option for Metasploit Pro Bruteforce**

# Create Executable Template Manually

- Use exploits/pro/windows/dynamic\_exe module
- Generates a dynamic exe on Metasploit Pro's local file system
- Payload can then be delivered by various out-of-band methods

```
msf exploit(dynamic_exe) > use exploit/pro/windows/dynamic_exe
msf exploit(dynamic_exe) > show options

Module options (exploit/pro/windows/dynamic_exe):

  Name      Current Setting  Required  Description
  ----      -
  FILENAME  msf.exe          yes       The file name.
  SERVICE   false            yes       Generate a Service EXE instead of a regular one?

Exploit target:

  Id  Name
  --  ---
  0    Automatic

msf exploit(dynamic_exe) > exploit

[*] Generating EXE template...
[*] Creating 'msf.exe' file...
[+] msf.exe stored at /Users/dmaloney/.msf4/local/msf.exe
msf exploit(dynamic_exe) > 
```

# Dynamic Executable Generator Advantages


- Dynamic nature
  - Code is different every time
- Appear harmless
  - Random C functions use completely harmless, innocuous operations
- Hide the payload
  - Scrambles payload in memory
- Detect debuggers
  - Avoids reverse engineering
- Dynamic run-time linking
  - AV often look at the imports table
  - Metasploit Pro evades detection through dynamic runtime linking
  - Keeps imports table almost empty
- All of this makes life for AV vendors really difficult
  - Sorry, guys!

# But Wait...

- Evasion still isn't 100%
- Some AV vendors are already catching up 8 months later
- So what are we missing?

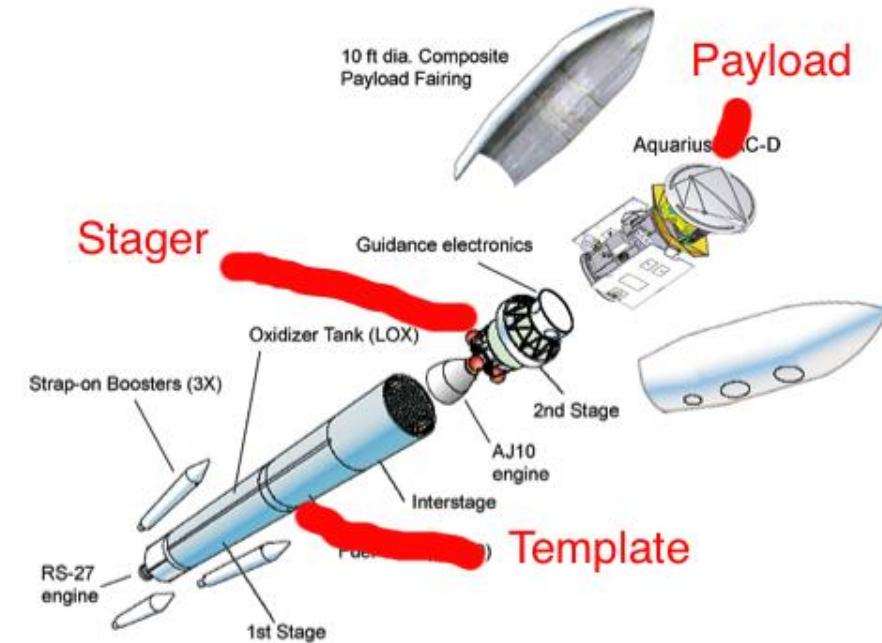
SHA256:	bfa8a84db89e5029c372f902808c39fe593144e5b76a434109127fa457a902d6
File name:	msf.exe
Detection ratio:	9 / 45
Analysis date:	2013-01-23 14:53:52 UTC ( 13 minutes ago )

[More details](#)



# The stager

- › Template 'stages' the stager
- › Stager talks back to multi-handler
- › Stager then stages the payload
- › Too many moving pieces
- › Redundant for our purposes





# Stager Template Hybrid

- › Make the EXE actually talk to the handler
- › EXE then stages the stage 2 payload
- › Profit!
- › Stay tuned!

# Other Resources

- <http://www.scriptjunkie.us/2011/04/why-encoding-does-not-matter-and-how-metasploit-generates-exes/>
- <http://schierlm.users.sourceforge.net/avevasion.html>
- <http://www.pentestgeek.com/2012/01/25/using-metasm-to-avoid-antivirus-detection-ghost-writing-asm/>
- Check out matt Block at CarolinaCon



*Live Demo*



## Q&A

David Maloney, Metasploit Software Engineer, Rapid7

[David\\_Maloney@rapid7.com](mailto:David_Maloney@rapid7.com)

@TheLightCosine