КОД БЕЗОПАСНОСТИ

115127, Россия, Москва а/я 66
+7 (495) 982-30-20
info@securitycode.ru
www.securitycode.ru

Добавление блокирующих правил по сигнатурам и защита от распространения вируса Petya

Сведения о способах защиты инфраструктуры предприятия от атаки и распространения вируса Petya

1. Одним из механизмов распространения вируса является эксплуатация уязвимости MS17-010(более известная как ETERNALBLUE), поэтому для защиты от этого типа атаки необходимо воспользоваться ранее данными рекомендациями по защите от вируса WannaCry:
   Документ по рекомендуемым способам защиты от вируса WannaCry
   Дополнительно для предотвращения эксплуатации MS17-010 была разработана обновленная(улучшенная) сигнатура, не блокирующая SMBv1 целиком[1].

2. Помимо эксплуатации MS17-010 для распространения внутри сети вирус Petya использует Windows Management Instrumentation (WMI) и утилиту PsExec, поэтому мы подготовили дополнительные сигнатуры, которые блокируют копирование тела вируса по сети и запрещают использование утилиты PsExec[2].

Сигнатуры

Описание	Сигнатура
Обновленная сигнатура для защиты от эксплуатации уязвимости MS17-010 (EternalBlue)	incoming && tcp.DataLength >= 0x49 && GetPacketDword(tcp_offset+tcp.HeaderLen + 0x5) == [53 4d 42 a0] && GetPacketDword(tcp_offset+tcp.HeaderLen + 0x2c) >= 66512 && GetPacketWord(tcp_offset+tcp.HeaderLen + 0x49) == [00 00]
Сигнатура для блокировки использования PsExec	incoming && tcp.DataLength >= 130 && GetPacketDword(tcp_offset+tcp.HeaderLen + 4) == [fe 53 4d 42] && GetPacketWord(tcp_offset+tcp.HeaderLen + 16) == [05 00] && SearchPacketData(tcp_offset+tcp.HeaderLen + 122, [50 00 53 00 45 00 58 00 45 00 53 00 56 00 43 00 2e 00 45 00 58 00 45]) > 0
Сигнатура для блокировки копирования perfc.dat	incoming && tcp.DataLength >= 130 && GetPacketDword(tcp_offset+tcp.HeaderLen + 4) == [fe 53 4d 42] && GetPacketWord(tcp_offset+tcp.HeaderLen + 16) == [05 00] &&

[1] При работе правила могут возникать ложные срабатывания при больших значениях расширенных атрибутов передаваемых файлов.

[2] Блокировать утилиту PsExec можно только на время эпидемии.

по SMBv2	SearchPacketData(tcp_offset+tcp.HeaderLen + 122, [70 00 65 00 72 00 66 00 63 00 2e 00 64 00 61 00 74 00]) > 0

Настройка новых сигнатур

Для добавления запрещающих правил по сигнатурам необходимо скопировать необходимые сигнатуры в файл Signatures.txt как указано ниже.

Добавление блокирующих правил по сигнатурам

Для того чтобы добавить запрещающие правило по разработанной специалистами Кода безопасности сигнатуре необходимо воспользоваться приведенными ниже скриптами.

Скрипты для добаления правил блокировки по сигнатурам

Примеры скриптов для добавления плавил блокировки по сигнатурам в межсетевой экран SNS.

SNS 8.2

- Для сетевой версии SNS необходимо выполнить powershell скрипт добавляющий правило для всех агентов
  Пример:
  Скрипт добавляет заприщающие правила доступа с сигнатурами из файла SignatureFile для всех агентов. Скрипт выполняется на сервере безопасности, выолнять на подчиненных серверах в одном домене безопасности не требуется.

  .ПАРАМЕТР SignatureFile
  Путь к файлу с сигнатурами для которых содзается запрещающее правило

  .ПАРАМЕТР Settingskst
  Путь к файлу settings.kst

  .Пример
  .\Set-PetyaCry.ps1 -SignatureFile 'C:\Signatures.txt' -Settingskst 'C:\Settings.kst'

```
[CmdletBinding()]
Param(
    [Parameter(Mandatory=$True,Position=1)]
    [ValidateScript( {Test-Path $_ -PathType Leaf})]
    [String]$SignatureFile,
    [Parameter(Mandatory=$false,Position=2)]
    [ValidateScript( {(Get-Item $_).Name -eq 'Settings.kst'})]
```

```
    [String]$Settingskst = 'C:\settings.kst'
)
$Signatures = Get-Content -Path $SignatureFile
[string]$allowrule = 'add nr %agent% /at allow /direction in_reply  /groups everyone'
[String]$rawrule   = 'add nr %agent% /at deny /direction in /local_ports %port% /grou
ps everyone /condition "%Signature%"'
$ports = @(139,445)
$rules =@()
$AuthServerInstallPath = (Get-ItemProperty "HKLM:\SOFTWARE\Security Code\Secret Net S
tudio\Server\Authentication Server").ProductInstallPath
$Realm = (Get-ItemProperty "HKLM:\SOFTWARE\Security Code\Secret Net Studio\Server\Aut
hentication Server").KRBREALM
$badmin = (Get-Content $Settingskst)[1]
$output = & "$($AuthServerInstallPath)\ScAuthSrvConfig.exe" $Realm '/p' "$badmin" '/q
' 'show computers'
$agents = $output[0..($output.IndexOf($output -like '*computer(s)*') - 2)]|ForEach-Ob
ject{($_ -split '\s+')[0].trim()}

$Signatures |ForEach-Object{ $rawruleitem = $rawrule.Replace('%Signature%', "$_")
    $agents |ForEach-Object{
        $agent = $_
        Write-Verbose -Message "Replacing names agents in template"
        $ports |ForEach-Object{
            $rules += ($rawruleitem.Replace('%agent%', "$agent")).Replace('%port%',$_
)
            }
        }
}
$agents |%{
    $rules += $allowrule.Replace('%agent%', $_)
}

$tmprulesfile = "$AuthServerInstallPath\rules"
if(Test-Path -Path $tmprulesfile){
    Write-Verbose -Message 'Rule file exist'
    Remove-Item -Path $tmprulesfile
}
$rules |Out-File $tmprulesfile -Encoding ascii -Append
& "$($AuthServerInstallPath)\ScAuthSrvConfig.exe"  $Realm '/p' "$badmin" '/s' "$tmpru
lesfile"
Remove-Item -Path $tmprulesfile
```

- Для локальной версии необходимо выполнить powershell скрипт добавляющий правило для всех сигнатур из файла
  Пример powershell скрипта: Скрипт добавляет запрещающие правила доступа с сигнатурами из файла SignatureFile.

.ПАРАМЕТР SignatureFile
Путь к файлу с сигнатурами для которых содзается запрещающее правило

.Пример
.\Set-PetyaCry.ps1 -SignatureFile 'C:\Signatures.txt'

```
[CmdletBinding()]
Param(
    [Parameter(Mandatory=$True,Position=1)]
    [ValidateScript( {Test-Path $_ -PathType Leaf})]
    [String]$SignatureFile
)
$Signatures = Get-Content -Path $SignatureFile
[String]$rawrule = 'add nr /at deny /direction in /local_ports 139;445 /groups everyo
ne /condition \"%Signature%\"'
[string]$allowrule = 'add nr /at allow /direction in_reply /groups everyone'
$rules =@()
$ScLocalSrvConfig = (Get-ItemProperty "HKLM:\SOFTWARE\Security Code\Secret Net Studio
\Client\Network Protection").ProductInstallPath + "\ScLocalSrvConfig.exe"

$Signatures |ForEach-Object{
    Write-Verbose -Message "Replacing names agents in template"
    $rules += $rawrule.Replace('%Signature%', "$_")
}
$rules |ForEach-Object{
    Write-Verbose -Message "Load rule $_"
    & $ScLocalSrvConfig '/q' $_
}
Write-Verbose -Message "Loading a rule that allows reply to incoming"
& $ScLocalSrvConfig '/q' $allowrule
```

SNS 8.1/8.0

- Для сетевой версии SNS необходимо выполнить powershell скрипт добавляющий правило для всех агентов
  Пример:
  Скрипт добавляет заприщающие правила доступа с сигнатурами из файла SignatureFile для всех агентов. Скрипт выполняется на сервере безопасности, выолнять на подчиненных серверах в одном домене безопасности не требуется.

  .ПАРАМЕТР SignatureFile
  Путь к файлу с сигнатурами для которых содзается запрещающее правило

  .ПАРАМЕТР Settingskst
  Путь к файлу settings.kst

  .Пример
  .\Set-PetyaCry.ps1 -SignatureFile 'C:\Signatures.txt' -Settingskst 'C:\Settings.kst'

```
[CmdletBinding()]
Param(
    [Parameter(Mandatory=$True,Position=1)]
    [ValidateScript( {Test-Path $_ -PathType Leaf})]
    [String]$SignatureFile,
    [Parameter(Mandatory=$false,Position=2)]
    [ValidateScript( {(Get-Item $_).Name -eq 'Settings.kst'})]
    [String]$Settingskst = 'C:\settings.kst'
)
$Signatures = Get-Content -Path $SignatureFile
[string]$allowrule = 'add nr %agent% /at allow /direction in_reply  /groups everyone'
[String]$rawrule   = 'add nr %agent% /at deny /direction in /local_ports %port% /grou
ps everyone /condition "%Signature%"'
$ports = @(139,445)
$rules =@()
$AuthServerInstallPath = (Get-ItemProperty "HKLM:\SOFTWARE\Security Code\Secret Net S
tudio\Server\Authentication Server").ProductInstallPath
$Realm = (Get-ItemProperty "HKLM:\SOFTWARE\Security Code\Secret Net Studio\Server\Aut
hentication Server").KRBREALM
$badmin = (Get-Content $Settingskst)[1]
$output = & "$($AuthServerInstallPath)\ScAuthSrvConfig.exe" $Realm '/p' "$badmin" '/q
' 'show computers'
$agents = $output[0..($output.IndexOf($output -like '*computer(s)*') - 2)]|ForEach-Ob
ject{($_ -split '\s+')[0].trim()}

$Signatures |ForEach-Object{ $rawruleitem = $rawrule.Replace('%Signature%', "$_")
    $agents |ForEach-Object{
        $agent = $_
        Write-Verbose -Message "Replacing names agents in template"
        $ports |ForEach-Object{
            $rules += ($rawruleitem.Replace('%agent%', "$agent")).Replace('%port%',$_
)
            }
        }
}
$agents |%{
    $rules += $allowrule.Replace('%agent%', $_)
}

$tmprulesfile = "$AuthServerInstallPath\rules"
if(Test-Path -Path $tmprulesfile){
    Write-Verbose -Message 'Rule file exist'
    Remove-Item -Path $tmprulesfile
}
$rules |Out-File $tmprulesfile -Encoding ascii -Append
& "$($AuthServerInstallPath)\ScAuthSrvConfig.exe"  $Realm '/p' "$badmin" '/s' "$tmpru
lesfile"
Remove-Item -Path $tmprulesfile
```

- Для локальной версии необходимо выполнить приведенный ниже скрипт для powershell
  Пример: .ПАРАМЕТР SignatureFile
  Путь к файлу с сигнатурами для которых содзается запрещающее правило

  .ПРИМЕР
  .\Set-PetyaCry.ps1 -SignatureFile 'C:\Signatures.txt'

```
[CmdletBinding()]
Param(
    [Parameter(Mandatory=$True,Position=1)]
    [ValidateScript( {Test-Path $_ -PathType Leaf})]
    [String]$SignatureFile
)
function CreateRuleNode
{
  Param(
    [System.Xml.XmlDocument]$XmlDoc,
    [String]$Name,
    [String]$Value
  )
  $Field = $XmlDoc.CreateElement("a")
  $Field.SetAttribute("name", $Name)
  $Field.SetAttribute("value", $Value)
  return $Field
}
function Set-RuleNode {
  Param(
    [Parameter(Mandatory=$True)][System.Xml.XmlDocument]$Xml,
    [Parameter(Mandatory=$false)][int]$index = 0,
    [Parameter(Mandatory=$false)][String]$ruleDirectionType = 'in',
    [Parameter(Mandatory=$false)][String]$ruleCondition = '',
    [Parameter(Mandatory=$false)] [ValidateSet('allow', 'deny')][String]$accesstype =
'allow',
    [Parameter(Mandatory=$True)][String]$localports,
    [Parameter(Mandatory=$True)][ref]$result
  )
  $Rule = $xml.CreateElement("Node")

  do
  {
    $guid = "{$([guid]::NewGuid().guid)}"
  } while($UsedGuid -icontains $guid)

  $Rule.SetAttribute("path", $guid)
  $tmp = CreateRuleNode -XmlDoc $xml -Name "order" -Value "$(110000 + $index)"
  $Rule.AppendChild($tmp)
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "enabled" -Value "1"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "ruletype" -Value "accessrule"
```

```
))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "service" -Value "network-tran
sport-with-auth"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "accesstype" -Value $accesstyp
e))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "audit-enabled" -Value "1"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "local-ports" -Value $localpor
ts))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "remote-ports" -Value "*"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "groups" -Value "{00000001-000
0-0000-0000-000000000000}"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "local-addrs" -Value "*"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "remote-addrs" -Value "*"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "protocol" -Value "*"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "rule-direction-type" -Value "
$ruleDirectionType"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "create-auth-rule" -Value "1")
)
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "flags" -Value "0"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "on-rule-action-cmd" -Value ""
))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "on-rule-action-cmd-folder" -V
alue ""))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "on-rule-action-cmd-start-type
" -Value "system"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "on-rule-action-cmd-token-type
" -Value "user"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "on-rule-action-cmd-beep" -Val
ue "0"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "accessmask" -Value ""))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "principals" -Value ""))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "rule-activate-regexp" -Value
""))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "network-level" -Value "packet
-level"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "rule-condition" -Value "$rule
Condition"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "adapters-to-include" -Value "
*"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "adapters-to-exclude" -Value "
"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "adapters-match" -Value "incl-
excl"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "is-emergency-rule" -Value "0"
))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "rule-scope" -Value "1"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "out-channel-protection-enable
d" -Value "0"))
```

```
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "reply-on-reject" -Value "1"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "processes-to-include" -Value
"*"))
  $Rule.AppendChild((CreateRuleNode -XmlDoc $xml -Name "processes-to-exclude" -Value
""))

  $result.Value = $rule
}

$ErrorActionPreference = "Stop"

$App = (Get-ItemProperty "HKLM:\SOFTWARE\Security Code\Secret Net Studio\Client\Netwo
rk Protection").ProductInstallPath + "\ScLocalCfg.exe"
$Signatures = Get-Content -Path $SignatureFile

$TempXML = $Env:TEMP + "\" + [guid]::NewGuid().Guid.ToUpper() + ".xml"

& $App FW Get main /file $TempXML

$xml = New-Object System.Xml.XmlDocument
$xml.Load($TempXML)

$NetAuth = $xml | Select-Xml -XPath "//Node[@path='network-transport-with-auth-rules'
]"

$UsedGuid = @()
# ChangeOrder
$NetAuth.Node.ChildNodes | %{
  $rule = $_
  $UsedGuid += $_.path
  $rule.a.GetEnumerator() | ?{$_.Name -eq "order"} | %{$_.value = ([Int32]$_.value +
($Signatures.Count + 1)).ToString()}
}
# Add New rule
$Signatures |ForEach-Object{$index = 1}{
  Set-RuleNode -Xml $xml -index $index -ruleCondition $_ -localports '139;445' -acces
stype 'deny' -result ([ref]$Rule)
  $NetAuth.Node.AppendChild($Rule)
  $index++
}
Set-RuleNode -Xml $xml -index 0 -ruleDirectionType 'in_reply' -localports '*'  -acces
stype 'allow' -result ([ref]$Rule)  #add rule allow in_replay data
$NetAuth.Node.AppendChild($Rule)
$xml.Save($TempXML)

& $App FW Set main /file $TempXML
```

```
Remove-Item $TempXML -Force
```

TrustAccess 1.3.2

- Для добавления правила защиты агентам TA необходимо выполнить следующий powershell скрипт
  Пример:

  Скрипт добавляет заприщающие правила доступа с сигнатурами из файла SignatureFile для всех агентов. Скрипт выполняется на сервере управления TrustAccess

  .ПАРАМЕТР TAadminPassword
  Пароль администратора TrustAccess

  .ПАРАМЕТР SignatureFile
  Путь к файлу с сигнатурами для которых содзается запрещающее правило

  .Пример
  .\Set-PetyaCryTA.ps1 -TAadminPassword 'test' -SignatureFile 'c:\docs\signatures.txt'

```
[CmdletBinding()]
Param(
    [Parameter(Mandatory=$True,Position=1)]
    [ValidateScript( {Test-Path $_ -PathType Leaf})]
    [String]$SignatureFile,
    [Parameter(Mandatory=$True)]
    [String]$TAadminPassword
)
$Signatures = Get-Content -Path $SignatureFile
[string]$allowrule = 'add nr %agent% /at allow /direction in_reply  /groups everyone'
[String]$rawrule   = 'add nr %agent% /at deny /direction in /local_ports %port% /grou
ps everyone /condition \"%Signature%\"'
$ports = @(139,445)
$rules =@()
$AuthServerInstallPath = (Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\Security Code\
TrustAccess\Management Server").ProductInstallPath
$Realm = (Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\Security Code\TrustAccess\Mana
gement Server").KRBREALM

$output = & "$($AuthServerInstallPath)\ScAuthSrvConfig.exe" $Realm '/p' "$TAadminPass
word" '/q' 'show computers'
$agents = $output[0..($output.IndexOf($output -like '*computer(s)*') - 2)]|ForEach-Ob
ject{($_ -split '\s+')[0].trim()}

$Signatures |ForEach-Object{ $rawruleitem = $rawrule.Replace('%Signature%', "$_")
$agents |ForEach-Object{
    $agent = $_
```

```
    Write-Verbose -Message "Replacing names agents $agent in template"
    $ports |ForEach-Object{
        $rules += ($rawruleitem.Replace('%agent%', "$agent")).Replace('%port%',$_)
        }
    }
}
$agents |%{
    Write-Verbose -Message "add rule allow in_replay data for $_"
    $rules += $allowrule.Replace('%agent%', $_)
}

$rules |ForEach-Object{
    Write-Verbose -Message "Load rule $_"
    & "$($AuthServerInstallPath)\ScAuthSrvConfig.exe"  $Realm '/p' "$TAadminPassword"
'/q' $_
}
```