Análise Técnica Externa da Causa Raiz — Arquivo de Canal (*Channel File*) 291

INTRODUÇÃO

Este relatório elabora as informações compartilhadas anteriormente em nossa Revisão Preliminar Pós-Incidente, aprofundando as descobertas, mitigações, detalhes técnicos e análise da causa raiz do incidente. Em 29 de julho às 17h (PT), usando uma comparação semana a semana, aproximadamente 99% dos sensores Windows estão online em comparação com antes da atualização do conteúdo. Normalmente, vemos uma variação de cerca de 1% semana a semana nas conexões de sensores.

Nesta RCA, usamos uma terminologia geral para descrever a plataforma CrowdStrike Falcon, de forma a facilitar a leitura. A terminologia em outros documentos pode ser mais específica e técnica.

O QUE ACONTECEU

O sensor CrowdStrike Falcon oferece poderosos modelos de inteligência artificial e machine learning no sensor para proteger os sistemas dos clientes, identificando e remediando as ameaças avançadas mais recentes. Esses modelos são mantidos atualizados e fortalecidos com o aprendizado da mais recente telemetria de ameaças de sensor e de inteligência humana do Falcon Adversary OverWatch, Falcon Complete e dos engenheiros de detecção de ameaças da CrowdStrike. Esse rico conjunto de telemetria de segurança começa quando os dados são filtrados e agregados em cada sensor em um armazenamento gráfico local.

Cada sensor correlaciona o contexto de seu armazenamento gráfico local com a atividade ativa do sistema em comportamentos e indicadores de ataque (IOAs) em um processo contínuo de refinamento. Este processo de refinamento inclui um Mecanismo de Detecção do Sensor que combina o Conteúdo do Sensor integrado com Conteúdo de Resposta Rápida fornecido pela nuvem. O Conteúdo de Resposta Rápida é usado para reunir telemetria, identificar indicadores de comportamento do adversário e aumentar novas detecções e prevenções no sensor sem exigir alterações no seu código. O Conteúdo de Resposta Rápida é uma Heurística Comportamental, separado e distinto das capacidades de prevenção e detecção por IA no sensor da CrowdStrike.

O Conteúdo de Resposta Rápida é entregue por meio de Arquivos de Canal (*Channel Files*) e interpretado pelo Interpretador de Conteúdo do Sensor, usando um mecanismo baseado em expressão regular. Cada arquivo de canal (*channel file*) de Conteúdo de Resposta

Rápida está associado a um Tipo de Modelo específico integrado a uma versão do sensor. O Tipo de Modelo fornece ao Interpretador de Conteúdo dados de atividade e contexto gráfico para serem comparados com o Conteúdo de Resposta Rápida.

Com o lançamento da versão 7.11 do sensor em fevereiro de 2024, a CrowdStrike introduziu um novo Tipo de Modelo para permitir a visibilidade e a detecção de novas técnicas de ataque que abusam de pipes nomeados (*named pipes*) e outros mecanismos de comunicação entre processos ("IPC") do Windows. Conforme descrito no PIR, o novo Tipo de Modelo IPC foi desenvolvido e testado de acordo com nossos processos padrão de desenvolvimento de Conteúdo do Sensor e foi integrado ao sensor para se preparar para a utilização em campo. As Instâncias de Modelo IPC são fornecidas como Conteúdo de Resposta Rápida aos sensores por meio de um Arquivo de Canal (*Channel File*) correspondente numerado 291.

O novo Tipo de Modelo IPC definiu 21 campos de parâmetros de entrada, mas o código de integração que invocou o Interpretador de Conteúdo com as Instâncias de Modelo do Arquivo de Canal (*Channel File*) 291 forneceu apenas 20 valores de entrada para comparação. Essa incompatibilidade de contagem de parâmetros evitou várias camadas de validação de compilação e testes, pois não foi descoberta durante o processo de teste de lançamento do sensor, o teste de estresse do Tipo de Modelo (usando uma Instância de Modelo de teste) ou as primeiras implementações bem-sucedidas de Instâncias de Modelo IPC no campo. Em parte, isso ocorreu devido ao uso de critérios de correspondência genérica para a 21ª entrada durante os testes e nas Instâncias de Modelo IPC iniciais.

Em 19 de julho de 2024, duas Instâncias de Modelo IPC adicionais foram implementadas. Um deles introduziu um critério de correspondência não genérica para o 21º parâmetro de entrada. Essas novas Instâncias de Modelo resultaram em uma nova versão do Arquivo de Canal (*Channel File*) 291 que agora exigiria que o sensor inspecionasse o 21º parâmetro de entrada. Até que esse arquivo de canal (*channel file*) fosse entregue aos sensores, nenhuma Instância de Modelo IPC nas versões anteriores do canal havia feito uso do 21º campo de parâmetro de entrada. O Validador de Conteúdo avaliou as novas Instâncias de Modelo, mas baseou sua avaliação na expectativa de que o Tipo de Modelo IPC fosse fornecido com 21 entradas.

Os sensores que receberam a nova versão do Arquivo de Canal (*Channel File*) 291 com o conteúdo problemático foram expostos a um problema latente de leitura fora dos limites no Interpretador de Conteúdo. Na próxima notificação de IPC do sistema operacional, as novas Instâncias de Modelo de IPC foram avaliadas, especificando uma comparação com o 21º valor de entrada. O Interpretador de Conteúdo esperava apenas 20 valores. Portanto, a tentativa de acessar o 21º valor produziu uma leitura de memória fora dos limites além do final da matriz de dados de entrada e resultou em uma falha no sistema.

Em resumo, foi a confluência desses problemas que resultou em uma falha no sistema: a incompatibilidade entre as 21 entradas validadas pelo Validador de Conteúdo e as 20 fornecidas ao Interpretador de Conteúdo, o problema de leitura latente fora dos limites no Interpretador de Conteúdo e a falta de um teste específico para critérios de correspondência não-genérica no 21º campo. Embora esse cenário com o Arquivo de Canal (*Channel File*) 291 agora não possa se repetir, ele também mostra as melhorias do processo e as etapas de mitigação que a CrowdStrike está implementando para garantir mais resiliência.

CONSTATAÇÕES E MITIGAÇÕES

1. O número de campos no Tipo de Modelo IPC não foi validado no momento da compilação do sensor

Conclusões: No momento do incidente, o código do sensor para o Tipo de Modelo IPC descrevia 20 fontes de entrada diferentes para uso pela Instância do Modelo. Isso significa que quando o sensor quisesse tomar uma decisão de detecção com base no Tipo de Modelo IPC, o código do sensor forneceria 20 fontes de entrada diferentes para o Interpretador de Conteúdo. No entanto, a definição do Tipo de Modelo IPC no arquivo Definições de Tipo de Modelo indicava que ele esperava 21 campos de entrada. Essa definição resultou em Instâncias de Modelo no Arquivo de Canal (*Channel File)* 291 que esperavam operar com 21 entradas. Essa incompatibilidade não foi detectada durante o desenvolvimento do Tipo de Modelo IPC. Os casos de teste e o Conteúdo de Resposta Rápida usados para testar o Tipo de Modelo IPC não acionaram uma falha durante o desenvolvimento da funcionalidade ou durante o teste da versão 7.11 do sensor.

Mitigação: Validar o número de campos de entrada no Tipo de Modelo no momento da compilação do sensor

Uma correção para o Compilador de Conteúdo do Sensor que valida o número de entradas fornecidas por um Tipo de Modelo foi desenvolvido em 19 de julho de 2024 e entrou em produção em 27 de julho de 2024, como parte das ferramentas de construção internas da CrowdStrike. Essa correção também verificou que nenhum outro Tipo de Modelo estava fornecendo um número incorreto de entradas, em qualquer plataforma.

2. Faltava uma verificação de limites de matriz em tempo de execução para os campos de entrada do Interpretador de Conteúdo no Arquivo de Canal (*Channel File*) 291

Descobertas: O Conteúdo de Resposta Rápida para o Arquivo de Canal (*Channel File*) 291 instruiu o Interpretador de Conteúdo a ler a 21ª entrada da matriz de ponteiro de entrada.

No entanto, o Tipo de Modelo IPC gera apenas 20 entradas. Como resultado, uma vez que o Conteúdo de Resposta Rápida foi entregue usando um critério de correspondência não genérico para a 21ª entrada, o Interpretador de Conteúdo executou uma leitura fora dos limites da matriz de entrada. Este não é um problema de gravação de memória arbitrária e foi revisado de forma independente.

Mitigação: Adicionar verificações de limites de matriz de entrada em tempo de execução ao Interpretador de Conteúdo para Conteúdo de Resposta Rápida no Arquivo de Canal (*Channel File*) 291

A verificação de limites foi adicionada à função Interpretador de Conteúdo que recupera *strings* de entrada em 25 de julho de 2024. Uma verificação adicional de que o tamanho da matriz de entrada corresponde ao número de entradas esperadas pelo Conteúdo de Resposta Rápida foi adicionada ao mesmo tempo. Essas correções estão sendo transferidas para todas as versões 7.11 e superiores de sensores Windows por meio de uma versão de hotfix do software do sensor. Esta versão estará disponível em 9 de agosto de 2024.

A verificação de limites adicionada impede que o Interpretador de Conteúdo execute um acesso fora dos limites da matriz de entrada e interrompa o sistema. A verificação adicional acrescenta uma camada extra de validação de tempo de execução para que o tamanho da matriz de entrada corresponda ao número de entradas esperado pelo Conteúdo de Resposta Rápida.

Concluímos o teste de fuzz do Tipo de Modelo do Canal (Channel) 291 e estamos expandindo-o para handlers de Conteúdo de Resposta Rápida adicionais no sensor.

Mitigação: Corrigir o número de entradas fornecidas pelo Tipo de Modelo IPC

O código do sensor que define o Tipo de Modelo IPC foi atualizado para fornecer o número correto de entradas (21). Essa correção está sendo transferida para todas as versões 7.11 e superiores de sensores Windows por meio de uma versão de hotfix do software do sensor. Esta versão estará disponível em 9 de agosto de 2024.

3. O teste de Tipo de Modelo deve abranger uma variedade maior de critérios de correspondência

Resultados: Testes manuais e automatizados foram realizados durante o desenvolvimento do Tipo de Modelo IPC. Esse teste se concentrou na validação funcional do Tipo de Modelo, incluindo o fluxo correto de dados relevantes para a segurança por meio dele, e na avaliação desses dados para gerar alertas de detecção apropriados com base nos critérios criados em casos de teste de desenvolvimento.

Os testes automatizados utilizaram ferramentas internas e externas para criar os dados relevantes à segurança necessários para exercer o Tipo de Modelo IPC em todas as versões compatíveis do Windows em um amplo subconjunto dos casos de uso operacional esperados. Para teste automatizado, um conjunto estático de 12 casos de teste foi selecionado para representar expectativas operacionais mais amplas e validar a criação de alertas de telemetria e detecção. Parte desse teste incluiu a definição de um arquivo de canal (*channel file*) para uso nos casos de teste. A seleção de dados no arquivo de canal (*channel file*) foi feita manualmente e incluiu um critério de correspondência genérica regex no 21º campo para todas as Instâncias de Modelo, o que significa que a execução desses testes durante o desenvolvimento e as compilações de versão não expôs a leitura latente fora dos limites no Interpretador de Conteúdo quando fornecido com 20 em vez de 21 entradas.

Mitigação: Aumentar a cobertura do teste durante o desenvolvimento do Tipo de Modelo

Para confirmar que estamos validando todos os campos em cada Tipo de Modelo, foram criados testes automatizados que testam com critérios de correspondência não genérica para cada campo. Essa etapa foi realizada para todos os Tipos de Modelo existentes e é necessária para todos os Tipos de Modelo futuros. Além disso, todos os futuros Tipos de Modelo incluem casos de teste com cenários adicionais que refletem melhor o uso na produção.

4. O Validador de Conteúdo continha um erro lógico

Conclusões: O Validador de Conteúdo avaliou as novas Instâncias de Modelo. No entanto, baseou sua avaliação na expectativa de que o Tipo de Modelo IPC fosse fornecido com 21 entradas. Isso resultou no envio da Instância de Modelo problemática ao Interpretador de Conteúdo.

Mitigação: Criar verificações adicionais no Validador de Conteúdo

O Validador de Conteúdo está sendo modificado para adicionar novas verificações para garantir que o conteúdo nas Instâncias de Modelo não inclua critérios correspondentes que correspondam a mais campos do que os que estão sendo fornecidos como entrada para o Interpretador de Conteúdo. Esta correção será lançada para produção em 19 de agosto de 2024.

Mitigação: Impedir a criação de arquivos problemáticos do Canal (*Channel*) 291

O Validador de Conteúdo foi modificado para permitir apenas critérios de correspondência genérica no 21º campo, o que impede o acesso fora dos limites nos sensores que fornecem apenas 20 entradas.

5. A validação da Instância de Modelo deve ser expandida para incluir testes no Interpretador de Conteúdo

Resultados: Os Tipos de Modelo recém-lançados são testados em vários aspectos, como utilização de recursos, impacto no desempenho do sistema e volume de detecção. Para muitos Tipos de Modelo, incluindo o Tipo de Modelo IPC, uma Instância de Modelo específica é usada para realizar Testes de Estresse do Tipo de Modelo, comparando com qualquer valor possível dos campos de dados associados para identificar interações adversas do sistema.

Um teste de estresse do Tipo de Modelo IPC com uma Instância de Modelo de teste foi executado em nosso ambiente de teste, que consiste em uma variedade de sistemas operacionais e cargas de trabalho. O Tipo de Modelo IPC passou no teste de estresse e foi validado para uso, e uma Instância de Modelo foi liberada para produção como parte de uma atualização de Conteúdo de Resposta Rápida.

No entanto, a Instância de Modelo testada pelo Validador de Conteúdo não observou que o número incompatível de entradas causaria uma falha no sistema quando fornecido ao Interpretador de Conteúdo pelo Tipo de Modelo IPC.

Mitigação: Atualizar procedimentos de teste do Sistema de Configuração de Conteúdo

O Sistema de Configuração de Conteúdo foi atualizado com novos procedimentos de teste para garantir que cada nova Instância de Modelo seja testada, independentemente do fato de a Instância de Modelo inicial ser testada com o Tipo de Modelo na criação. Isso fornece às Instâncias de Modelo testes adicionais antes da implantação de produção.

6. As Instâncias de Modelo devem ter uma implantação em estágios

Descobertas: Cada Instância de Modelo deve ser implementada em uma distribuição em estágios.

Mitigação: O Sistema de Configuração de Conteúdo foi atualizado com camadas adicionais de implementação e verificações de aceitação

A implementação em estágios reduz o impacto se uma nova Instância de Modelo causar falhas, como falhas no sistema, picos de volume de detecção de falsos positivos ou problemas de desempenho. As novas Instâncias de Modelo que passarem pelo teste parcial devem ser promovidas sucessivamente para implementações mais amplas em anel ou revertidas se forem detectados problemas. Cada anel é projetado para identificar e mitigar

possíveis problemas antes de uma implementação mais ampla. A promoção de uma Instância Modelo para o próximo anel sucessivo é seguida por um tempo adicional de preparação, em que a telemetria é coletada para determinar o impacto geral da Instância de Modelo no endpoint.

Mitigação: Fornecer controle ao cliente sobre a implementação de atualizações do Conteúdo de Resposta Rápida

A plataforma Falcon foi atualizada para oferecer aos clientes maior controle sobre o fornecimento de Conteúdo de Resposta Rápida. Os clientes podem escolher onde e quando as atualizações do Conteúdo de Resposta Rápida serão implementadas. Continuamos aprimorando essas capacidades para oferecer um controle mais granular sobre as implementações do Conteúdo de Resposta Rápida, juntamente com detalhes de atualização de conteúdo por meio de notas de versão, que os clientes podem assinar.

REVISÃO INDEPENDENTE DE TERCEIROS

A CrowdStrike contratou dois fornecedores independentes de segurança de software para realizar uma análise mais aprofundada do código do sensor Falcon para garantir a segurança e a qualidade. Além disso, estamos realizando uma análise independente do processo de qualidade de ponta a ponta, desde o desenvolvimento até a implementação. Ambos os fornecedores iniciaram revisões com foco imediato no código e no processo afetados em 19 de julho.

DETALHES TÉCNICOS

Histórico e terminologia

A CrowdStrike faz as atualizações da configuração de conteúdo de segurança de seus sensores de duas formas: Conteúdo dos Sensores, que é enviado diretamente com nosso sensor; e Conteúdo de Resposta Rápida, que é criado para responder ao dinâmico cenário de ameaça em velocidade operacional.

O processamento do Conteúdo de Resposta Rápida baseado em *regex* no sensor envolve vários componentes:

- **Interpretador de Conteúdo:** Parte do código C++ do sensor, que pode testar *strings* de entrada em relação a expressões regulares.

- **Tipos de Modelo:** Contêm campos predefinidos para engenheiros de detecção de ameaças aproveitarem o Conteúdo de Resposta Rápida. Os Tipos de Modelo são expressos em código e compilados no sensor no momento da construção.
- **Arquivo de Definições de Tipo de Modelo:** Define os parâmetros de cada Tipo de Modelo. As definições neste arquivo incluem informações sobre qual Arquivo de Canal (*Channel File*) entregará o Conteúdo de Resposta Rápida para cada Tipo de Modelo, quantas entradas o Tipo de Modelo deve usar e que tipo de dados são necessários para cada entrada.
- **Conteúdo do Sensor:** Determina como combinar dados relevantes para a segurança com o Conteúdo de Resposta Rápida para tomar determinadas decisões de detecção. O Conteúdo do Sensor inclui modelos de IA e machine learning no sensor, bem como Tipos de Modelo. Ele é compilado como parte da liberação do sensor.
- **Instâncias de Modelo:** Critérios de correspondência desenvolvidos por engenheiros de detecção. As Instâncias de Modelo consistem em conteúdo regex destinado ao uso com um Tipo de Modelo específico e identificam dados específicos para uso em operações de segurança. Elas são definidas usando uma interface do usuário orientada pelo arquivo de Definições de Tipo de Modelo.
- **Conteúdo de Resposta Rápida:** Consiste em várias Instâncias de Modelo agrupadas. O Conteúdo de Resposta Rápida é entregue por arquivo de canal (*channel file*).
- **Validador de Conteúdo**: Verifica a validade dos arquivos do canal (*channel files*) em relação à sua definição no arquivo de Definições de Tipo de Modelo.
- **Sistema de Configuração de Conteúdo**: Usado para criar Instâncias de Modelo, que são validadas e implementadas no sensor por meio de um mecanismo chamado **Arquivos de Canal** (***Channel Files***).

Uso do driver kernel em um produto de segurança

Conforme descrito por David Weston no blog Microsoft Security, os produtos de segurança no ecossistema Windows, incluindo o sensor Falcon, geralmente utilizam os drivers *kernel* como componentes principais de uma oferta de segurança robusta.

A presença no *kernel* oferece ampla visibilidade das atividades relevantes para a segurança em todo o sistema, como criação de processos e *threads* ou arquivos sendo gravados, excluídos e modificados no disco. As interfaces expostas pelo *kernel* permitem que os drivers da CrowdStrike imponham controles críticos para um produto de segurança, como prevenção em linha de processos maliciosos ou bloqueio de arquivos de *malware* gravados em disco.

O driver *kernel* da CrowdStrike é carregado desde uma fase inicial da inicialização do sistema para permitir que o sensor observe e se defenda contra *malwares* lançados antes do início dos processos no modo de usuário.

Fornecer conteúdo de segurança atualizado (por exemplo, Conteúdo de Resposta Rápida da CrowdStrike) para essas capacidades *kernel* permite que o sensor defenda os sistemas contra um cenário de ameaças em rápida evolução sem fazer alterações no código *kernel*. O Conteúdo de Resposta Rápida é um dado de configuração; não é um código ou um driver *kernel*.

O CrowdStrike certifica cada nova versão do sensor Windows por meio do programa Windows Hardware Quality Labs (WHQL), que inclui testes extensivos em todos os testes necessários no Windows Hardware Lab Kit (HLK) e no Windows Hardware Certification Kit (HCK) da Microsoft. O processo de certificação WHQL marca o fim de um desafio abrangente de testes internos envolvendo testes funcionais, testes de longevidade, testes de estresse com injeção de falhas, testes de *fuzzing* e testes de desempenho. Durante os testes necessários para o programa WHQL, os sensores usam as versões mais recentes dos arquivos de canal (*channel files*) no momento da certificação.

À medida que novas versões do Windows introduzem suporte para executar mais dessas funções de segurança no espaço do usuário, a CrowdStrike atualiza seu agente para utilizar esse suporte. Ainda há um trabalho significativo para que o ecossistema Windows ofereça suporte a um produto de segurança robusto que não depende de um driver kernel para pelo menos algumas de suas funcionalidades. Estamos comprometidos em trabalhar diretamente com a Microsoft de forma contínua, à medida que o Windows continua adicionando mais suporte às necessidades de produtos de segurança no espaço do usuário.

Análise de Crash Dump

Para ilustrar como as novas Instâncias de Modelo no Arquivo de Canal (*Channel File*) 291 levaram a uma falha do sistema, examinamos brevemente um *crash dump kernel* de um sistema afetado pelo conteúdo problemático. Isso expande a análise de crash compartilhada por David Weston no blog de Segurança da Microsoft.

Abrindo o *crash dump* no depurador de *kernel* do Windows e usando o padrão de comando !analyze -v para um resumo rápido, vemos que ocorreu uma falha de memória (também conhecida como “violação de acesso”). *(Observação: Detalhes de depuração não relacionados são omitidos por motivos de brevidade, e um crash dump representativo é analisado aqui. Existem variações do dump, dependendo dos detalhes do estado da máquina.*)

```
1: kd> !analyze -v
*******************************************************************************
*                                                                             *
*                        Bugcheck Analysis                                    *
*                                                                             *
*******************************************************************************

PAGE_FAULT_IN_NONPAGED_AREA (50)
Invalid system memory was referenced.  This cannot be protected by try-except.
Typically the address is just plain bad or it is pointing at freed memory.
Arguments:
Arg1: ffffd6030000006a, memory referenced.
Arg2: 0000000000000000, X64: bit 0 set if the fault was due to a not-present PTE.
        bit 1 is set if the fault was due to a write, clear if a read.
        bit 3 is set if the processor decided the fault was due to a corrupted PTE.
        bit 4 is set if the fault was due to attempted execute of a no-execute PTE.
        - ARM64: bit 1 is set if the fault was due to a write, clear if a read.
        bit 3 is set if the fault was due to attempted execute of a no-execute PTE.
Arg3: fffff8020ebc14ed, If non-zero, the instruction address which referenced the bad memory
        address.
Arg4: 0000000000000002, (reserved)

READ_ADDRESS:  ffffd6030000006a Paged pool

MM_INTERNAL_CODE:  2

IMAGE_NAME:  csagent.sys

MODULE_NAME: csagent

FAULTING_MODULE: fffff8020eae0000 csagent

PROCESS_NAME:  System

TRAP_FRAME:  ffffae035f57eca0 -- (.trap 0xffffae035f57eca0)
NOTE: The trap frame does not contain all registers.
Some register values may be zeroed or incorrect.
rax=ffffae035f57f280 rbx=0000000000000000 rcx=0000000000000003
rdx=ffffae035f57f250 rsi=0000000000000000 rdi=0000000000000000
rip=fffff8020ebc14ed rsp=ffffae035f57ee30 rbp=ffffae035f57ef30
 r8=ffffd6030000006a  r9=0000000000000000 r10=0000000000000000
r11=0000000000000014 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei ng nz na po nc
csagent+0xe14ed:
fffff802`0ebc14ed 458b08          mov     r9d,dword ptr [r8] ds:ffffd603`0000006a=????????
Resetting default scope

STACK_TEXT:
ffffae03`5f57ea78 fffff802`05add2da     : 00000000`00000050 ffffd603`0000006a 00000000`00000000
ffffae03`5f57eca0 : nt!KeBugCheckEx
ffffae03`5f57ea80 fffff802`05947efc     : ffffd603`000ed454 00000000`00000000 00000000`00000000
ffffd603`0000006a : nt!MiSystemFault+0x1bc19a
ffffae03`5f57eb80 fffff802`05a2707e     : 00000000`00000000 ffffd603`e33a019e ffffae03`5f57f0a0
ffffae03`5f57f0a0 : nt!MmAccessFault+0x29c
ffffae03`5f57eca0 fffff802`0ebc14ed     : 00000000`00000000 ffffae03`5f57ef30 ffffd603`f208200c
ffffd603`f207a05c : nt!KiPageFault+0x37e
```

```
ffffae03`5f57ee30 fffff802`0eb9709e     : 00000000`00000000 00000000`e01f008d ffffae03`5f57f202
fffff802`0ed6aaf8 : csagent+0xe14ed
ffffae03`5f57efd0 fffff802`0eb98335     : 00000000`00000000 00000000`00000010 00000000`00000002
ffffd603`f207a01c : csagent+0xb709e
ffffae03`5f57f100 fffff802`0edd20c7     : 00000000`00000000 00000000`00000000 ffffae03`5f57f402
00000000`00000000 : csagent+0xb8335
ffffae03`5f57f230 fffff802`0edcec44     : ffffae03`5f57f6e8 fffff802`060abae0 ffffd603`ed408580
00000000`00000003 : csagent+0x2f20c7
ffffae03`5f57f4b0 fffff802`0eb47a31     : 00000000`0000303b ffffae03`5f57f770 ffffd603`edc908a0
ffffc189`7fcd4098 : csagent+0x2eec44
ffffae03`5f57f670 fffff802`0eb46aee     : ffffd603`edc908a0 fffff802`0ebf1e7e 00000000`00006820
fffff802`0ed3f8f0 : csagent+0x67a31
ffffae03`5f57f7e0 fffff802`0eb4685b     : ffffae03`5f57fa58 ffffd603`edc97830 ffffd603`edc908a0
ffffc189`7f90f4b8 : csagent+0x66aee
ffffae03`5f57f850 fffff802`0ebe99ea     : 00000000`f047f4ef ffff49ac`ca0f55d4 00000000`00000000
ffffd603`ec18fc30 : csagent+0x6685b
ffffae03`5f57f8d0 fffff802`0eb3efbb     : 00000000`00000000 ffffae03`5f57fad9 ffffc189`7f90f010
ffffc189`7f7ea470 : csagent+0x1099ea
ffffae03`5f57fa00 fffff802`0eb3edd7     : ffffc189`7ab79000 00000000`00000000 ffffc189`7f90f010
ffffc189`00000001 : csagent+0x5efbb
ffffae03`5f57fb40 fffff802`0ebde681     : 00000000`00000000 00000000`00000000 ffffc189`7f5a97d0
ffffc189`7f7ea470 : csagent+0x5edd7
ffffae03`5f57fb70 fffff802`05879ca7     : ffffc189`7faa8040 00000000`00000080 fffff802`0ebde510
00000000`00000000 : csagent+0xfe681
ffffae03`5f57fbb0 fffff802`05a1af64     : ffffe601`bcf51180 ffffc189`7faa8040 fffff802`05879c50
00000000`00000000 : nt!PspSystemThreadStartup+0x57
ffffae03`5f57fc00 00000000`00000000     : ffffae03`5f580000 ffffae03`5f579000 00000000`00000000
00000000`00000000 : nt!KiStartSystemThread+0x34
```

Esse comando de triagem automatizada identifica `csagent.sys` como o driver que executa o acesso à memória fora dos limites. `csagent.sys` é o driver de filtro do sistema de arquivos da CrowdStrike, um tipo de driver *kernel* que se registra com componentes do sistema operacional Windows para receber notificações de atividades do sistema relevantes para a segurança em tempo real.

Entre as notificações para as quais o driver da CrowdStrike registra está uma notificação para a criação de pipes nomeados. Quando o driver recebe uma notificação de pipe nomeado, esses dados são combinados com outras informações contextuais sobre o sistema. Esses dados combinados são apresentados para avaliação em relação às Instâncias de Modelo transmitidas no Arquivo de Canal (*Channel File*) 291.

Para examinar mais de perto esse processo, visualizamos o estado do registro no ponto da leitura da memória fora dos limites, restaurando o quadro da armadilha e desmontando as instruções anteriores para nos orientarmos. *(Observação: Essa listagem de desmontagem foi modificada da saída do depurador padrão para anotar o código com nomes de símbolos ilustrativos.)*

```
1: kd> .trap 0xffffae035f57eca0
NOTE: The trap frame does not contain all registers.
Some register values may be zeroed or incorrect.
rax=ffffae035f57f280 rbx=0000000000000000 rcx=0000000000000003
rdx=ffffae035f57f250 rsi=0000000000000000 rdi=0000000000000000
rip=fffff8020ebc14ed rsp=ffffae035f57ee30 rbp=ffffae035f57ef30
 r8=fffffd6030000006a  r9=0000000000000000 r10=0000000000000000
r11=0000000000000014 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei ng nz na po nc
csagent+0xe14ed:
fffff802`0ebc14ed 458b08          mov     r9d,dword ptr [r8]
ds:ffffd603`0000006a=????????

1: kd> u @rip-16 L0n10
csagent!TemplateGetString+0xe:
fffff802`0ebc14d7 4e8b04d8        mov     r8,qword ptr [rax+r11*8]
fffff802`0ebc14db 750b            jne     csagent!TemplateGetString+0x1f
(fffff802`0ebc14e8)
fffff802`0ebc14dd 4d85c0          test    r8,r8
fffff802`0ebc14e0 7412            je      csagent!TemplateGetString+0x2b
(fffff802`0ebc14f4)
fffff802`0ebc14e2 450fb708        movzx   r9d,word ptr [r8]
fffff802`0ebc14e6 eb08            jmp     csagent!TemplateGetString+0x27
(fffff802`0ebc14f0)
fffff802`0ebc14e8 4d85c0          test    r8,r8
fffff802`0ebc14eb 7407            je      csagent!TemplateGetString+0x2b
(fffff802`0ebc14f4)
fffff802`0ebc14ed 458b08          mov     r9d,dword ptr [r8]
fffff802`0ebc14f0 4d8b5008        mov     r10,qword ptr [r8+8]
```

Antes deste trecho de código, os dados de contexto da notificação de pipe nomeado foram preparados para o Tipo de Modelo IPC como uma matriz de 20 ponteiros de entrada, cada um apontando para uma estrutura de string que contém um endereço de buffer e um valor de tamanho. Este trecho pretende selecionar uma das entradas para retornar seu endereço e tamanho de buffer, de acordo com um índice especificado pelo Arquivo de Canal (*Channel File*) 291.

Quando inserimos esse código, o endereço da matriz de ponteiro de 20 entradas é mantido no registro rax, e o registro r11 indica que a entrada a ser recuperada está no índice 0x14, ou seja, o 21º elemento.

Examinando o array de entrada, de fato encontramos um array de 20 ponteiros para estruturas de strings de entrada, seguido por um 21º valor que não *aponta* para memória válida:

```
1: kd> dp @rax l0n21
ffffae03`5f57f280  ffffae03`5f57f320 ffffae03`5f57f330
ffffae03`5f57f290  ffffae03`5f57f340 ffffae03`5f57f350
ffffae03`5f57f2a0  ffffae03`5f57f360 ffffae03`5f57f370
ffffae03`5f57f2b0  ffffae03`5f57f380 ffffae03`5f57f390
ffffae03`5f57f2c0  ffffae03`5f57f3a0 ffffae03`5f57f3b0
ffffae03`5f57f2d0  ffffae03`5f57f3c0 ffffae03`5f57f3d0
ffffae03`5f57f2e0  ffffae03`5f57f3e0 ffffae03`5f57f3f0
ffffae03`5f57f2f0  ffffae03`5f57f400 ffffae03`5f57f410
ffffae03`5f57f300  ffffae03`5f57f420 ffffae03`5f57f430
ffffae03`5f57f310  ffffae03`5f57f440 ffffae03`5f57f450
ffffae03`5f57f320  ffffd603`0000006a
1: kd> !pte ffffd603`0000006a
                                           VA ffffd6030000006a
PXE at FFFFFE7F3F9FCD60    PPE at FFFFFE7F3F9AC060    PDE at FFFFFE7F3580C000
PTE at FFFFFE6B01800000
contains 0A00000107A00863  contains 0000000000000000
pfn 107a00    ---DA--KWEV  contains 0000000000000000
not valid
```

Depois de ler esse ponteiro inválido no registro r8, o fluxo de controle no snippet acima dá o primeiro salto para o endereço `fffff802`0ebc14e8`, executa uma verificação de ponteiro nulo e, em seguida, tenta uma leitura por meio do ponteiro inválido, resultando em uma leitura fora dos limites e uma verificação de bug subsequente.

RECURSOS ADICIONAIS

Insira referências e links para recursos técnicos adicionais.

Hub de Orientação e Remediação: Atualização de Conteúdo do Falcon para hosts de Windows

Blog: Detalhes técnicos: Atualização de conteúdo do Falcon para hosts de Windows

Hub de Remediação — Glossário de termos

Este documento é uma tradução da seguinte versão em inglês: https://www.crowdstrike.com/wp-content/uploads/2024/08/Channel-File-291-Incident-Root-Cause-Analysis-08.06.2024.pdf. A versão traduzida é fornecida apenas para fins de referência e praticidade. Em caso de qualquer conflito ou ambiguidade, a versão em inglês sempre prevalecerá e terá precedência.