External Technical Root Cause Analysis — Channel File 291

INTRODUCTION

This report elaborates on the information previously shared in our preliminary Post Incident Review, going into further depth on the findings, mitigations, technical details and root cause analysis of the incident. As of July 29 at 5 p.m. PT, using a week-over-week comparison, ~99% of Windows sensors are online compared to before the content update. We typically see a variance of ~1% week-over-week in sensor connections.

Throughout this RCA, we have used generalized terminology to describe the CrowdStrike Falcon platform for improved readability. Terminology in other documentation may be more specific and technical.

WHAT HAPPENED

The CrowdStrike Falcon sensor delivers powerful on-sensor AI and machine learning models to protect customer systems by identifying and remediating the latest advanced threats. These models are kept up-to-date and strengthened with learnings from the latest threat telemetry from the sensor and human intelligence from Falcon Adversary OverWatch, Falcon Complete and CrowdStrike threat detection engineers. This rich set of security telemetry begins as data filtered and aggregated on each sensor into a local graph store.

Each sensor correlates context from its local graph store with live system activity into behaviors and indicators of attack (IOAs) in an ongoing process of refinement. This refinement process includes a Sensor Detection Engine combining built-in Sensor Content with Rapid Response Content delivered from the cloud. Rapid Response Content is used to gather telemetry, identify indicators of adversary behavior, and augment novel detections and preventions on the sensor without requiring sensor code changes. Rapid Response Content is behavioral heuristics, separate and distinct from CrowdStrike's on-sensor AI prevention and detection capabilities.

Rapid Response Content is delivered through Channel Files and interpreted by the sensor's Content Interpreter, using a regular-expression based engine. Each Rapid Response Content channel file is associated with a specific Template Type built into a sensor release. The Template Type provides the Content Interpreter with activity data and graph context to be matched against the Rapid Response Content.

With the release of sensor version 7.11 in February 2024, CrowdStrike introduced a new Template Type to enable visibility into and detection of novel attack techniques that abuse named pipes and other Windows interprocess communication ("IPC") mechanisms. As outlined in the PIR, the new IPC Template Type was developed and tested according to our standard Sensor Content development processes and was integrated into the sensor to prepare for utilization in the field. IPC Template Instances are delivered as Rapid Response Content to sensors via a corresponding Channel File numbered 291.

The new IPC Template Type defined 21 input parameter fields, but the integration code that invoked the Content Interpreter with Channel File 291's Template Instances supplied only 20 input values to match against. This parameter count mismatch evaded multiple layers of build validation and testing, as it was not discovered during the sensor release testing process, the Template Type (using a test Template Instance) stress testing or the first several successful deployments of IPC Template Instances in the field. In part, this was due to the use of wildcard matching criteria for the 21st input during testing and in the initial IPC Template Instances.

On July 19, 2024, two additional IPC Template Instances were deployed. One of these introduced a non-wildcard matching criterion for the 21st input parameter. These new Template Instances resulted in a new version of Channel File 291 that would now require the sensor to inspect the 21st input parameter. Until this channel file was delivered to sensors, no IPC Template Instances in previous channel versions had made use of the 21st input parameter field. The Content Validator evaluated the new Template Instances, but based its assessment on the expectation that the IPC Template Type would be provided with 21 inputs.

Sensors that received the new version of Channel File 291 carrying the problematic content were exposed to a latent out-of-bounds read issue in the Content Interpreter. At the next IPC notification from the operating system, the new IPC Template Instances were evaluated, specifying a comparison against the 21st input value. The Content Interpreter expected only 20 values. Therefore, the attempt to access the 21st value produced an out-of-bounds memory read beyond the end of the input data array and resulted in a system crash.

In summary, it was the confluence of these issues that resulted in a system crash: the mismatch between the 21 inputs validated by the Content Validator versus the 20 provided to the Content Interpreter, the latent out-of-bounds read issue in the Content Interpreter, and the lack of a specific test for non-wildcard matching criteria in the 21st field. While this scenario with Channel File 291 is now incapable of recurring, it also informs process improvements and mitigation steps that CrowdStrike is deploying to ensure further enhanced resilience.

FINDINGS AND MITIGATIONS

1. The number of fields in the IPC Template Type was not validated at sensor compile time

Findings: At the time of the incident, the sensor code for the IPC Template Type described 20 different input sources for use by the Template Instance. This means that when the sensor wanted to make a detection decision based on the IPC Template Type, the sensor code would supply 20 different input sources to the Content Interpreter. However, the definition of the IPC Template Type in the Template Type Definitions file stated that it expected 21 input fields. This definition resulted in Template Instances in Channel File 291 that expected to operate on 21 inputs. This mismatch was not detected during development of the IPC Template Type. The test cases and Rapid Response Content used to test the IPC Template Type did not trigger a fault during feature development or during testing of the sensor 7.11 release.

Mitigation: Validate the number of input fields in the Template Type at sensor compile time

A patch for the Sensor Content Compiler that validates the number of inputs provided by a Template Type was developed on July 19, 2024, and went into production on July 27, 2024, as part of CrowdStrike's internal build tooling. The Sensor Content Compiler patch also verified that no other Template Types were providing an incorrect number of inputs, on any platform.

2. A runtime array bounds check was missing for Content Interpreter input fields on Channel File 291

Findings: The Rapid Response Content for Channel File 291 instructed the Content Interpreter to read the 21st entry of the input pointer array. However, the IPC Template Type only generates 20 inputs. As a result, once Rapid Response Content was delivered that used a non-wildcard matching criterion for the 21st input, the Content Interpreter performed an out-of-bounds read of the input array. This is not an arbitrary memory write issue and has been independently reviewed.

Mitigation: Add runtime input array bounds checks to the Content Interpreter for Rapid Response Content in Channel File 291

Bounds checking was added to the Content Interpreter function that retrieves input strings on July 25, 2024. An additional check that the size of the input array matches the number of inputs expected by the Rapid Response Content was added at the same time. These fixes are being backported to all Windows sensor versions 7.11 and above through a sensor software hotfix release. This release will be generally available by August 9, 2024.

The added bounds check prevents the Content Interpreter from performing an out-of-bounds access of the input array and crashing the system. The additional check adds an extra layer of runtime validation that the size of the input array matches the number of inputs expected by the Rapid Response Content.

We have completed fuzz testing of the Channel 291 Template Type and are expanding it to additional Rapid Response Content handlers in the sensor.

Mitigation: Correct the number of inputs provided by the IPC Template Type

The sensor code defining the IPC Template Type was updated to provide the correct number of inputs (21). This fix is being backported to all Windows sensor versions 7.11 and above through a sensor software hotfix release. This release will be generally available by August 9, 2024.

3. Template Type testing should cover a wider variety of matching criteria

Findings: Both manual and automated testing were performed during the development of the IPC Template Type. This testing was focused on functional validation of the Template Type including the correct flow of security-relevant data through it, and evaluation of that data to generate appropriate detection alerts based on criteria created in development test cases.

Automated testing leveraged internal and external tooling to create the required security-relevant data needed to exercise the IPC Template Type under all supported Windows versions within a broad subset of the expected operational use cases. For automated testing, a static set of 12 test cases was selected to be representative of broader operational expectations and to validate the creation of telemetry and detection alerts. Part of this testing included defining a channel file for use within the test cases. The selection of data in the channel file was done manually and included a regex wildcard matching criterion in the 21st field for all Template Instances, meaning that execution of these tests during development and release builds did not expose the latent out-of-bounds read in the Content Interpreter when provided with 20 rather than 21 inputs.

Mitigation: Increase test coverage during Template Type development

To confirm that we are validating all fields in each Template Type, automated tests have been created that test with non-wildcard matching criteria for each field. This step has been done for all existing Template Types and is required for all future Template Types.

Additionally, all future Template Types include test cases with additional scenarios that better reflect production usage.

4. The Content Validator contained a logic error

Findings: The Content Validator evaluated the new Template Instances. However, it based its assessment on the expectation that the IPC Template Type would be provided with 21 inputs. This resulted in the problematic Template Instance being sent to the Content Interpreter.

Mitigation: Create additional checks in the Content Validator

The Content Validator is being modified to add new checks to ensure that content in Template Instances does not include matching criteria that match over more fields than are being provided as input to the Content Interpreter. This fix will be released to production by August 19, 2024.

Mitigation: Prevent the creation of problematic Channel 291 files

The Content Validator was modified to only allow wildcard matching criteria in the 21st field, which prevents the out-of-bounds access in the sensors that only provide 20 inputs.

5. Template Instance validation should expand to include testing within the Content Interpreter

Findings: Newly released Template Types are stress tested across many aspects, such as resource utilization, system performance impact and detection volume. For many Template Types, including the IPC Template Type, a specific Template Instance is used to stress test the Template Type by matching against any possible value of the associated data fields to identify adverse system interactions.

A stress test of the IPC Template Type with a test Template Instance was executed in our test environment, which consists of a variety of operating systems and workloads. The IPC Template Type passed the stress test and was validated for use, and a Template Instance was released to production as part of a Rapid Response Content update.

However, the Content Validator-tested Template Instance did not observe that the mismatched number of inputs would cause a system crash when provided to the Content Interpreter by the IPC Template Type.

Mitigation: Update Content Configuration System test procedures

The Content Configuration System has been updated with new test procedures to ensure that every new Template Instance is tested, regardless of the fact that the initial Template Instance is tested with the Template Type at creation. This provides Template Instances with additional testing prior to production deployment.

6. Template Instances should have staged deployment

Findings: Each Template Instance should be deployed in a staged rollout.

Mitigation: The Content Configuration System has been updated with additional deployment layers and acceptance checks

Staged deployment mitigates impact if a new Template Instance causes failures such as system crashes, false-positive detection volume spikes or performance issues. New Template Instances that have passed canary testing are to be successively promoted to wider deployment rings or rolled back if problems are detected. Each ring is designed to identify and mitigate potential issues before wider deployment. Promoting a Template Instance to the next successive ring is followed by additional bake-in time, where telemetry is gathered to determine the overall impact of the Template Instance on the endpoint.

Mitigation: Provide customer control over the deployment of Rapid Response Content updates

The Falcon platform has been updated to provide customers with increased control over the delivery of Rapid Response Content. Customers can choose where and when Rapid Response Content updates are deployed. We are continuing to enhance this capability to provide more granular control over Rapid Response Content deployments together with content update details via release notes, to which customers can subscribe.

INDEPENDENT THIRD-PARTY REVIEW

CrowdStrike has engaged two independent third-party software security vendors to conduct further review of the Falcon sensor code for both security and quality assurance. Additionally, we are conducting an independent review of the end-to-end quality process from development through deployment. Both vendors have started reviews with an immediate focus on the July 19 impacted code and process.

TECHNICAL DETAILS

Background and Terminology

CrowdStrike delivers security content configuration updates to our sensors in two ways: Sensor Content that is shipped with our sensor directly, and Rapid Response Content that is designed to respond to the changing threat landscape at operational speed.

The processing of regex-based Rapid Response Content on the sensor involves several components:

- **Content Interpreter:** Part of the sensor C++ code, which can test input strings against regexes.
- **Template Types:** Contain predefined fields for threat detection engineers to leverage in Rapid Response Content. Template Types are expressed in code and compiled into the sensor at build time.
- **Template Type Definitions file:** Defines the parameters of each Template Type. Definitions in this file include information about which Channel File will deliver the Rapid Response Content for each Template Type, how many inputs the Template Type is meant to use and what kind of data is required for each input.
- **Sensor Content:** Determines how to combine security-relevant data with Rapid Response Content in order to make certain detection decisions. Sensor Content includes on-sensor AI and machine learning models as well as Template Types. It is compiled as part of the sensor release.
- **Template Instances:** Matching criteria developed by detection engineers. Template Instances consist of regex content intended for use with a specific Template Type. Template Instances identify specific data for use in security operations. Template Instances are defined using a UI driven by the Template Type Definitions file.
- **Rapid Response Content:** Consists of multiple Template Instances bundled together. Rapid Response Content is delivered by channel file.
- **Content Validator**: Checks the validity of channel files against their definition in the Template Type Definitions file.
- **Content Configuration System**: Used to create Template Instances, which are validated and deployed to the sensor through a mechanism called **Channel Files**.

Kernel Driver Usage in a Security Product

As outlined by David Weston on the Microsoft Security blog, security products in the Windows ecosystem, including the Falcon sensor, commonly leverage kernel drivers as core components of a robust security offering.

Presence in the kernel offers rich visibility into system wide security-relevant activities, such as process and thread creation, or files being written, deleted and modified on disk. The interfaces exposed by the kernel allow CrowdStrike's drivers to enforce critical controls for a security product, such as inline prevention of malicious processes or blocking of malware files being written to disk.

CrowdStrike's kernel driver is loaded from an early phase of system boot to allow the sensor to observe and defend against malware that launches prior to user mode processes starting.

Providing up-to-date security content (e.g., CrowdStrike's Rapid Response Content) to these kernel capabilities enables the sensor to defend systems against a rapidly evolving threat landscape without making changes to kernel code. Rapid Response Content is configuration data; it is not code or a kernel driver.

CrowdStrike certifies each new Windows sensor release through the Windows Hardware Quality Labs (WHQL) program, which includes extensive testing through all required tests in Microsoft's Windows Hardware Lab Kit (HLK) and Windows Hardware Certification Kit (HCK). The WHQL certification process marks the end of a comprehensive internal testing gauntlet involving functional tests, longevity tests, stress tests with fault injection, fuzzing and performance tests. During the testing required for the WHQL program, the sensors use the latest versions of channel files at the time of certification.

As new versions of Windows introduce support for performing more of these security functions in user space, CrowdStrike updates its agent to utilize this support. Significant work remains for the Windows ecosystem to support a robust security product that doesn't rely on a kernel driver for at least some of its functionality. We are committed to working directly with Microsoft on an ongoing basis as Windows continues to add more support for security product needs in userspace.

Crash Dump Analysis

To illustrate how the new Template Instances in Channel File 291 led to a system crash, we briefly examine a kernel crash dump from a system impacted by the problematic content. This expands on the crash analysis shared by David Weston on the Microsoft Security blog.

Opening the crash dump in the Windows kernel debugger and using the standard !analyze -v command for a quick summary, we see that a memory fault (also known as an "access violation") bugcheck has occurred. *(Note: Unrelated debugging details are omitted for brevity, and a representative crash dump is analyzed here. Variations of the dump exist, depending on the details of the machine state.)*

```
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
```

This automated triage command identifies `csagent.sys` as the driver performing the out-of-bounds memory access. `csagent.sys` is CrowdStrike's file system filter driver, a type of kernel driver that registers with components of the Windows operating system to receive notifications of security-relevant system activities in real time.

Among the notifications that CrowdStrike's driver registers for is a notification for the creation of named pipes. When the driver receives a named pipe notification, this data is combined with other contextual information about the system. This combined data is presented for evaluation against the Template Instances conveyed in Channel File 291.

To look more closely at this process, we view the register state at the point of the out-of-bounds memory read by restoring the trap frame and disassembling the preceding instructions to orient ourselves. *(Note: This disassembly listing has been modified from the standard debugger output in order to annotate the code with illustrative symbol names.)*

```
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=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=????????

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]
```

Prior to this code snippet, the context data from the named pipe notification has been prepared for the IPC Template Type as an array of 20 input pointers, each pointing to a string structure that holds a buffer address and a size value. This snippet intends to select one of the inputs to return its buffer address and size, according to an index specified by Channel File 291.

As we enter this code, the address of the 20-input pointer array is held in register rax, and register r11 indicates that the input to be retrieved is at index 0x14, i.e., the 21st element.

Examining the input array, we indeed find an array of 20 pointers to input string structures, followed by a 21st value which does *not* point to valid memory:

```
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
```

After reading this invalid pointer into register `r8`, the control flow in the snippet above takes the first jump to address `fffff802`0ebc14e8`, performs a NULL pointer check, and then attempts a read through the invalid pointer, resulting in an out-of-bounds read and a subsequent bugcheck.

Additional Resources

Remediation and Guidance Hub: Falcon Content Update for Windows Hosts
Blog: Technical Details: Falcon Content Update for Windows Hosts
Remediation Hub — Glossary of Terms