Was ist Application Whitelisting?

Einführung

Application Whitelisting bedeutet im Kontext von Cyber Crucible, einer Anwendung die Erlaubnis zu erteilen, eine Aktion auszuführen, die andernfalls als verdächtig oder bösartig eingestuft werden könnte.

Application-Whitelisting-Technologie findet sich typischerweise in defensiven Fachpraktiken der Verhaltensanalyse, obwohl einige Angreifer sie ebenfalls (hier nicht behandelt) in ihren Operationen einsetzen.

Man kann es sich vorstellen wie in der Schule: Normalerweise durften Schüler während des Unterrichts nicht durch die Flure laufen. Wenn ihnen jedoch von einer Autoritätsperson eine Art „Hallenausweis“ ausgestellt wurde, wurden sie nicht in ihr Klassenzimmer zurückgebracht oder ins Büro des Schulleiters geschickt.

Arten von Application Whitelisting - eine nicht-technische Beschreibung

Bleiben wir bei der Analogie des „Hallenausweises“, so gibt es verschiedene Techniken, die Schulverwaltungen im Laufe der Zeit entwickelt haben, um die Verhaltensweisen zu formalisieren, mit denen manche Schüler (ganz bestimmt nicht unser CEO Dennis) davonzukommen versuchten. All diese Techniken haben eine direkte Entsprechung im Sicherheitsbetrieb.

Der offensichtliche & umständliche Hallenausweis (eine gute Sache)

21987331.jpg?width=278
Erinnern Sie sich noch an einen Hallenausweis oder einen Schlüssel zu einem Raum, der unglaublich groß und auffällig war?

Dafür gab es ein paar gute Gründe:

1. Damit er nicht verloren ging.
2. Um den Zeit- und Energieaufwand einer Autoritätsperson bei der schnellen Überprüfung der genehmigten Aktivitäten des Schülers zu verringern.

Gute Software „verliert“ keine Whitelist, aber eine Application-Whitelist-Software sollte eine schnelle, ressourceneffiziente Überprüfung ermöglichen, dass die Anwendung geprüft wurde.

Der generische Hallenausweis (wahrscheinlich keine gute Sache)

Dies ist wahrscheinlich sowohl für die Anwendungsüberwachung als auch für Schulen am wenigsten nützlich, erfordert aber auch das geringste Management oder Fachwissen.

In diesem Fall darf der Schüler seine Besorgung im Flur erledigen. Der Lehrer weiß vielleicht Bescheid, aber es gibt keine Überprüfung durch Aufsichtspersonen, was der Schüler tatsächlich tut. Selbst wenn eine Überprüfung versucht wird, entstehen zwei Ebenen erheblichen Ressourcenaufwands:

1. der Aufwand, den Schüler anzusprechen und zu seinem Verhalten und seiner Absicht zu befragen
2. der Aufwand, sicherzustellen, dass der Schüler ehrlich ist, um seine Erklärung zu überprüfen

Bei Anwendungen zeigt sich dies meist dann, wenn ein Programm zu einer Whitelist hinzugefügt wird, ohne dass eine weitere Überprüfung der Verhaltensweisen oder erlaubten Aktivitäten erfolgt.

Leider steht die Möglichkeit, Kontext zu den Verhaltensweisen einer Anwendung zu erhalten, in der Regel nicht zur Verfügung, wenn dies nicht extern durch ein anderes Framework nachverfolgt wird.

Der generisch-spezifische Hallenausweis

Nicht zu verwechseln mit „Los, Anwendung, mach, was du willst!“

Kehren wir zum Toilettenausweis zurück, denn er passt hier besonders gut. Angesichts der Sichtbarkeit des Ausweises ist es relativ einfach zu beurteilen, was der Schüler tun sollte. Grobe Verhaltensweisen, die nicht mit dem „Toilettengang“ übereinstimmen, lassen sich ohne großen Aufwand erkennen.

Application Whitelisting mit generischen Grenzen der Aktivitäten ist am häufigsten anzutreffen, obwohl diese Grenzen in vielen Fällen dem Äquivalent entsprechen, einem Schüler mit Toilettenausweis das Spielen auf dem Schulhof zu erlauben.

Der spezifisch-spezifische Hallenausweis

Nicht zu verwechseln mit dem „generisch-spezifischen“ Hallenausweis!

22642689.jpg?width=340
Während wir beim generisch-spezifischen Hallenausweis wissen, dass der Schüler die Toilette benutzt, wissen wir in einer großen Schule nicht:
1. Wer dem Schüler die Erlaubnis erteilt hat

2. Wo er gerade herkommt.

3. Welche Toilette die Autoritätsperson bei ihrer Genehmigung im Sinn hatte.

4. Wie lange der Schüler schon „unterwegs“ ist.

Dies erforderte mehr Arbeit am Autoritätsrahmen der Schule.

Um zum Application Whitelisting zurückzukehren: Die Verhaltensanalyse wird genauer, je mehr Variablen in eine Entscheidungsfindungsfähigkeit einfließen.

Wenn man außerdem einen ausgestellten Hallenausweis überprüft, muss man normalerweise keine Kette von 5 Lehrern zurückverfolgen, um zur Quelle zu gelangen. Man kann auch fast immer darauf

vertrauen, dass der Lehrer keine Art „schlechter Lehrer“ war, der Schülern falsches Verhalten beibringt. Application Whitelisting erfordert im Beispiel einer spezifisch-spezifischen Whitelist ein Rückwärtsarbeiten, um jede übergeordnete oder vorangehende Anwendung zu untersuchen, die die nächste Anwendung geöffnet hat. Zum Beispiel könnte ein Stück Malware ein legitimes Windows-Dienstprogramm geöffnet haben.

Die folgenden Fragen entsprechen ungefähr dem oben genannten „Schüler-Hallenausweis“:

1. Wofür wurde die Anwendung gestartet, oder was wurde ihr aufgetragen? Wir müssen das beabsichtigte Verhalten kennen.
2. Wer hat die Anwendung gestartet? Dies ist typischerweise ein anderes Programm.
3. Welche „übergeordneten“ oder „vorangehenden“ Programme führten zum Öffnen dieses Programms? Was taten oder beabsichtigten diese jeweils zu tun?

Die kompromittierte Anwendung

Erinnern Sie sich an verschiedene Science-Fiction- oder Horrorliteratur und -filme, in denen ein Betrüger auf irgendeine Weise die

22413315.jpg?width=226
Kontrolle über das Verhalten einer Figur übernommen hat? In Schulen besteht kaum die Gefahr, dass eine Lehrkraft von Außerirdischen übernommen wird.

Die Übernahme von Anwendungen ist eine beliebte Technik von Angreifern, denn wie in den Filmen bemerken die meisten Beobachter nichts Ungewöhnliches, bis es zu spät ist.

In diesem Fall übernimmt ein Angreifer ein laufendes Programm, das gerade in den Speicher geladen ist. Das vollständig intakte Programm im Dateisystem wird dabei ignoriert. Das Programm im Speicher wird bearbeitet, um den Code des Angreifers hinzuzufügen - ähnlich wie ein Außerirdischer im Körper eines Menschen. Der Angreifer führt seine kriminellen Handlungen aus, und wenn das Programm fertig ausgeführt wurde, ist der größte Teil der Beweise verschwunden. Wenn das legitime Programm erneut läuft und der Angreifer den Code nicht erneut bearbeitet, verhält es sich genau so, wie es die ursprünglichen legitimen Programmierer beabsichtigt hatten. Vielleicht wäre in diesem Fall ein Werwolf die bessere Analogie?

Eine Application-Whitelist muss daher zusätzlichen Aufwand betreiben, um sicherzustellen, dass das Programm sowie alle vorangehenden oder übergeordneten Programme intakt sind und nicht von Hackern bearbeitet wurden, während sie ausgeführt werden.

Zusammenfassung - Fragen, die bei der Untersuchung einer Application-Whitelist-Funktion gestellt werden sollten

Stellen Sie sicher, dass Sie die Grenzen dessen verstehen, was eine Whitelist erlaubt, wenn Sie mit Ihrem Technologieteam oder Anbieter über Application Whitelisting sprechen.

1. Untersucht und verfolgt diese Funktion, was ein Programm tun soll, und vergleicht sie erwartetes mit beobachtetem Verhalten?
2. Wie viele Details erfasst die Whitelist-Funktion bei ihrer Entscheidungsfindung? Mehr oder weniger als ein detaillierter Schüler-Hallenausweis?
3. Verfolgt und untersucht die Whitelist-Funktion übergeordnete und vorangehende Programme auf Bösartigkeit? (Programm A öffnet B, öffnet C, öffnet D - sehr häufig)
4. Überprüft die Whitelist-Funktion, ob das laufende Programm nicht manipuliert wurde?

Vertiefende Lektüre des National Institute of Standards & Technology (NIST)

Eine maßgebliche Publikation, die nicht zur beiläufigen Lektüre gedacht ist, aber sehr anschaulich beschreibt.

Obwohl sie nicht jedes Szenario abdeckt und die Details nicht häufig genug aktualisiert werden, um mit jedem Szenario Schritt zu halten, sind die Beschreibungen strategisch genug, um in den meisten Fällen wertvoll zu sein.

Das Team von Cyber Crucible schätzt Erkenntnisse aus NIST-Publikationen stets sehr und nimmt sich entsprechend Zeit dafür.

https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-167.pdf

---

Revision #1
Created 2026-07-24 04:25:11 UTC by Dennis Underwood

Updated 2026-07-24 04:25:11 UTC by Dennis Underwood