Cisco ESA 및 CES에서 정규식 구성 및 검증

목차

소개

이 문서에서는 ESA와 CES가 필터에서 정규식을 사용하는 방법, 주요 동작 차이 및 시행 전 테스트의 필요성에 대해 설명합니다.

배경 정보

이 문서에서는 Cisco ESA(Email Security Appliance) 및 Cisco CES(Cloud Email Security)가 메시지 필터 및 콘텐츠 필터 내에서 사용될 때 정규식을 처리하는 방법에 대해 설명합니다. 특히 정규식이 이러한 구성 요소에서 작동하는 방식 및 정규식이 이메일 헤더, 본문 내용 및 첨부 파일과 상호 작용하는 방식을 이해하는 데 중점을 둡니다.

DLP 모듈에서 사용되는 정규식 엔진이 다르게 동작함을 처음부터 명확히 하는 것이 중요합니다. 따라서 이 문서에 설명된 모든 내용은 메시지 필터 및 콘텐츠 필터에만 적용되고 DLP 정책에는 적용되지 않습니다.

ESA에서 정규식으로 작업할 때 관리자는 이메일 콘텐츠가 메일 클라이언트에 시각적으로 표시되는 것과 같은 방식으로 평가되지 않는다는 점을 이해해야 합니다. 이메일 메시지는 봉투 정보, 구조화된 헤더, MIME 부분 및 잠재적으로 인코딩된 콘텐츠를 포함합니다. 그 결과, 메시지 구조 및 regex 동작을 완전히 이해하지 못하는 경우 필터에 의해 수행되는 비교는 예기치 않은 결과를 생성할 수 있다.

따라서 정규식을 사용하는 새 필터는 시행 전에 항상 모니터 모드에서 활성화할 수 있습니다. 이를 통해 실제 트래픽에 대한 검증을 수행할 수 있으며 의도하지 않은 차단이나 성능 영향을 방지할 수 있습니다.

사전 및 검색 용어

메시지 필터 또는 콘텐츠 필터를 만들 때 여러 조건에 입력한 용어는 정규식으로 해석됩니다. 이는 매우 중요한 개념입니다. 관리자가 리터럴 텍스트를 확인하려는 경우에도 ESA는 regex 논리를 사용하여 입력을 처리할 수 있습니다.

이는 모든 조건 유형에 일률적으로 적용되는 것은 아닙니다. 예를 들어, 특정 구조화된 조건에서 특정 IP 주소를 검색할 때 이 값은 정규식으로 해석되지 않습니다. 그러나 Subject 헤더, 메시지 본문, 특정 헤더 필드 또는 첨부 파일 이름 내에서 검색할 경우 이 값은 일반적으로 regex 패턴으로 처리됩니다.

일반적인 예는 이것을 명확하게 보여준다. 제목이 포함된 이메일을 차단하는 것이 목표라고 가정합니다.

```
Receipt number (123456)
```

괄호는 정규식(그룹화에 사용됨)의 특수 문자이므로 이스케이프해야 합니다.

올바른 식은 다음과 같습니다.

```
Receipt number \(123456\)
```

괄호가 이스케이프되지 않으면 regex 엔진은 괄호를 리터럴 문자가 아닌 그룹화 연산자로 해석합니

다. 패턴에 따라 의도하지 않은 일치 또는 예상과 다른 동작이 발생할 수 있습니다.

이 때문에 regex에서 어떤 문자가 특별한 의미를 갖는지 이해하고, 리터럴 매칭이 필요할 때 적절하게 이스케이프되도록 하는 것이 필수적이다.

특수 문자와 이스케이프된 구문의 예

첫 번째 열에는 특수 문자가 포함된 샘플 텍스트가 표시되고, 두 번째 열에는 Cisco ESA(Python-style regex)의 리터럴 텍스트와 일치하도록 올바른 정규식 구문을 작성하는 방법이 표시됩니다.

일치시킬 리터럴 텍스트	올바른 정규식 구문
수금 번호(123456)	수신 번호 \(123456\)
user@example.com	user@example\.com
www.test.abc	www\.test\.abc
file_name.txt	file_name\.txt
가격은 10.50	가격은 10\.50입니다
C:\Users\Admin	C:\\Users\\Admin을 참조하십시오.
[기밀]	\[기밀\]
{송장}	\{invoice\}
+34 600 123 456	\+34 600 123 456
질문?	질문\?
100% 보장	100% 보장(% 이스케이프 불필요)
별표 * 기호	별표 * 기호
A|B	A|B
캐럿 ^시작	캐럿 \^시작
100달러	달러 \$100

정규식 사용 제한

정규식은 신중하게 사용해야 하며 필요한 경우에만 사용해야 합니다. 강력한 매칭 기능을 제공하지만, 과도한 표현식이나 잘못 설계된 표현식은 메시지 처리 시간을 늘리고 의도하지 않은 일치를 생성할 수 있습니다.

주의를 요하는 특정 구문은 .*이며, 이는 "임의의 문자, 0회 이상"을 나타냅니다. 식의 시작 또는 끝에 배치하면 과도한 역추적과 불필요한 처리 오버헤드가 발생할 수 있습니다.

Cisco 설명서에는 시작 또는 끝에 .*를 사용하는 항목이 특정 MIME 부분을 매칭할 때 특정 조건에

서 시스템이 잠길 수 있음을 나타냅니다. 따라서 가능하면 선행 또는 후행 .*을 사용하지 않는 것이 좋습니다.

많은 시나리오에서 관리자는 .*invoice.*와 같은 패턴을 사용하여 간단하게 송장을 작성하고 ESA에서 동일한 실질적인 결과를 생성할 수 있습니다. 검사 엔진은 관련 콘텐츠 영역을 이미 검색하므로 .*로 단어를 둘러싼 경우는 종종 중복되고 계산적으로 비효율적입니다.

⚠ 주의: 일반적인 권장 사항은 정규식을 최대한 단순하고 정확하게 유지하는 것입니다.

메시지 필터, 콘텐츠 필터 및 사전

Cisco ESA는 메시지를 평가하고 조치를 적용하는 여러 메커니즘을 제공합니다. 메시지 필터는 파이프라인의 시작 부분에서 작동하며 스크립팅 스타일 구문을 사용합니다. 매우 유연하며 봉투 데이터, 헤더 및 첨부 파일 속성과 관련된 고급 논리를 허용합니다. 그러나 프로세스 체인의 초기에 실행되므로 비효율적인 메시지 필터가 성능에 부정적인 영향을 미칠 수 있습니다.

콘텐츠 필터는 그래픽 인터페이스를 통해 구성되며 메시지가 수락된 후 작동합니다. 대부분의 콘텐츠 검사 활용 사례에서 Content Filters는 성능 측면에서 관리가 더 쉽고 안전합니다.

메시지 필터 및 콘텐츠 필터 모두에서 정규식을 조건에 직접 또는 사전을 사용하여 간접적으로 도입할 수 있습니다.

관리자는 사전을 통해 재사용 가능한 검색 용어를 중앙 집중화할 수 있습니다. 각 항목은 별도의 줄에 작성되며 일반 텍스트 또는 정규식이 될 수 있습니다. 사전은 ASCII가 아닌 문자도 지원하므로 다국어 환경에 적합합니다.

일부 상황에서는 특정 복잡한 정규식 구문이 사전 내에서 동일하게 동작하지 않을 수 있습니다. 이 경우 정규식은 사전 내부가 아니라 필터 조건에 직접 배치해야 합니다.

Cisco ESA에서는 최대 150개의 콘텐츠 사전을 만들 수 있습니다. CLI를 통해 dictionaryconfig 명령을 사용하여 제한을 수정하지 않는 한 기본적으로 100개의 사전을 구성할 수 있습니다.

사전은 또한 용어 가중치를 구현할 수 있습니다. 각 용어에 숫자 가중치를 할당할 수 있으며, ESA가 메시지를 스캔할 때 해당 용어의 발생 횟수를 가중치와 곱합니다. 결과 점수를 필터에 정의된 임계값과 비교합니다. 이 점수부여 모델을 통해 더 유연하고 단계적인 정책 시행이 가능합니다.

또한 사전에는 사회 보장 번호 또는 뱅킹 식별자와 같은 구조화된 숫자 패턴에 대한 알고리즘 탐지기인 스마트 식별자가 포함될 수 있습니다.

정규식 엔진

Cisco ESA는 Python re 모듈 스타일을 기반으로 정규식을 사용합니다. 이는 일반적인 Python regex 구문과 호환되지만, 전체 Python 환경에서 지원되는 모든 고급 기능이 반드시 ESA에서 지원되는 것은 아닙니다.

정확한 문자열 일치를 위해 표현식은 시작 부분에 ^를 사용하고 끝 부분에 $ 를 사용하여 고정해야 합니다. 이러한 앵커가 없으면 regex 엔진은 전체 값이 아닌 하위 문자열을 매칭할 수 있습니다.

예를 들어, 식은 다음과 같습니다.

```
sun.com
```

다음과 같은 일치 문자열:

```
thegodsunocommando
```

그러나 표현식은 다음과 같습니다.

```
^sun\.com$
```

정확한 문자열 sun.com만 일치시킵니다.

빈 문자열을 매칭할 때는 ""를 사용하지 않는 것이 중요합니다. 모든 문자열과 효과적으로 매칭되기 때문입니다. 대신 올바른 표현식은 다음과 같습니다.

```
^$
```

Cisco ESA는 Python 스타일 정규식을 사용하므로, 두 가지 방법으로 대/소문자를 구분하지 않고 비교할 수 있습니다.

기본적으로, 앞에서 언급한 것처럼 정규식은 대/소문자를 구분합니다. 즉 다음을 검색합니다.

```
foo
```

foo만 일치하고 FOO, Foo 또는 Foo는 일치하지 않습니다.

대/소문자를 구분하지 않는 일치를 수행하려면 정규식 시작 부분에 인라인 플래그(?i)를 사용할 수 있습니다. 이렇게 하면 regex 엔진이 나머지 패턴의 대소문자를 무시하게 됩니다.

예를 들면 다음과 같습니다.

```
(?i)foo
```

이 식은 다음과 일치합니다.

- 먹이
- 푸우
- 푸
- 오

대/소문자를 무시하고 전체 문자열을 정확히 일치시키려면 대/소문자를 구분하지 않는 플래그를 앵커와 결합할 수 있습니다.

```
(?i)^foo$
```

이렇게 하면 대문자 여부에 관계없이 전체 값이 정확히 "foo"가 됩니다.

또 다른(덜 실용적인) 대안은 문자 클래스를 사용하여 가능한 모든 조합을 명시적으로 정의하는 것입니다. 예를 들면 다음과 같습니다.

```
[Ff][Oo][Oo]
```

그러나 이 접근 방식은 유지하기가 어려워지며 대신 (?i) 플래그를 사용할 수 있는 경우에는 권장되지 않습니다.

대부분의 ESA 시나리오에서 대/소문자를 구분하지 않는 일치에 대해 선호되고 가장 깨끗한 방법은 다음과 같습니다.

```
(?i)
```

정규식의 시작 부분입니다.

비 ASCII 문자 및 단어 경계

더블바이트 문자 집합을 사용하는 언어에서는 단어 경계 또는 대/소문자를 예상대로 사용할 수 없습니다. \w와 같은 구문에 의존하는 복잡한 식은 인코딩이나 로캘을 알 수 없는 경우 일관되지 않은 결과를 생성할 수 있습니다.

이러한 경우 사전 구성에서 단어 경계 적용을 비활성화하거나 모호한 문자 클래스에 대한 종속성을

방지하기 위해 식을 간소화하는 것이 좋습니다.

비 ASCII 사전으로 작업할 때 CLI 표시는 터미널 인코딩에 따라 문자를 올바르게 렌더링할 수 없습니다. 이러한 경우 사전을 텍스트 파일로 내보내고 외부에서 편집한 다음 다시 가져오는 것이 좋습니다.

효율적인 필터 작성

특히 대용량 환경에서 필터를 작성할 때는 효율성이 매우 중요합니다. 일반적인 실수는 비슷한 일치를 위해 OR 조건의 긴 체인을 작성하는 것입니다.

예를 들어, 수십 개의 첨부 파일 확장명을 개별적으로 검사하면 regex 엔진이 반복적으로 초기화됩니다. 따라서 CPU 사용량이 증가하고 유지 관리 가능성이 줄어듭니다.

여러 개별 비교를 작성하는 대신 단일 정규식 내에서 대안을 사용하여 그룹화하면 처리 오버헤드가 크게 줄어듭니다. 이렇게 하면 regex 엔진이 호출되는 횟수가 줄어들고 필터를 쉽게 유지 관리할 수 있습니다.

효율적인 필터 설계는 가독성뿐만 아니라 시스템 성능에 직접적인 영향을 미칩니다.

PDF 및 정규식

PDF 파일 내의 내용을 일치시키면 PDF가 생성된 방식에 따라 예기치 않은 결과가 발생할 수 있습니다. 일부 PDF는 내부 표현에 논리적 공백 또는 줄 바꿈이 없습니다. 스캐닝 엔진은 단어 위치 지정에 기초하여 논리적 간격을 재구성하려고 시도한다.

단어가 여러 글꼴이나 글꼴 크기를 사용하여 구성된 경우 내부 표현은 텍스트를 조각화할 수 있습니다. 예를 들어 "callout"이라는 단어는 내부적으로 "call out" 또는 "c a l lout"으로 해석될 수 있습니다.

이러한 경우 내부 표현에 정확한 연속 문자열이 포함되지 않으므로 "callout" 표현식을 일치시키려는 시도가 실패할 수 있습니다. 관리자는 PDF 첨부 파일을 대상으로 하는 콘텐츠 기반 정책을 설계할 때 이러한 제한 사항에 유의해야 합니다.

정규식 테스트

정규식을 프로덕션에 배포하기 전에 테스트하는 것은 중요한 운영 요구 사항입니다. 구문적으로 올바른 것처럼 보이는 정규식은 실제 이메일 트래픽과 비교하여 평가하면 매우 다르게 작동할 수 있습니다. 적절한 테스트가 없으면 필터가 오탐을 발생시키거나, 의도한 패턴을 감지하지 못하거나, 성능 오버헤드를 유발하거나, 의도치 않게 정상적인 메일 흐름을 방해할 수 있습니다.

테스트는 프로덕션에서 필터를 활성화하기 전에 위험을 최소화하기 위해 구조화된 2단계 프로세스로 접근해야 합니다.

1단계 - 정규식 설계 및 검증
첫 번째 단계에서는 정규식을 Cisco ESA에 통합하기 전에 정규식 자체를 설계하고 검증하는 데 중점을 둡니다.

1. regex101 또는 이와 유사한 툴의 사용

http://regex101.com과 같은 온라인 플랫폼(또는 그에 상응하는 툴)은 설계 단계에서 매우 유용합니다. 이러한 툴을 사용할 경우 ESA의 regex 엔진에 근접하려면 Python 버전을 선택해야 합니다.

이러한 플랫폼을 통해 관리자는 다음과 같은 작업을 수행할 수 있습니다.

- 구문 정확성 검증
- 특수 문자가 제대로 이스케이프되었는지 확인합니다.
- 일치하는 케이스와 일치하지 않는 케이스 모두 테스트
- 그룹화 및 한정자 동작 시각화
- 잠재적으로 탐욕스러운 구문(예: .*)을 식별합니다.

그러나 이러한 툴은 표준 Python regex 동작을 시뮬레이션하며 Cisco ESA에서 완전히 구현되지 않은 기능을 지원할 수 있습니다. 따라서 확정적 호환성 테스트보다는 예비 검증 툴로 간주해야 합니다.

2. AI 모델 사용(ChatGPT, Copilot 등)

AI 기반 보조자는 특히 복잡한 매칭 시나리오의 경우 regex 생성을 가속화할 수 있습니다. 원하는 행동을 자연어로 기술함으로써 관리자는 초기 regex 제안을 얻을 수 있으며, 그런 다음 수정이 가능합니다.

AI 툴은 다음과 같은 경우에 특히 유용합니다.

- 복합 그룹 식 생성
- 비즈니스 요구 사항을 regex 구문으로 변환
- 긴 OR 기반 조건을 그룹화된 대안으로 단순화

그럼에도 불구하고 AI가 생성한 표현은 항상 비판적으로 검토해야 한다. 비효율성, 지원되지 않는 구조 또는 지나치게 복잡한 논리를 도입할 수 있습니다. AI 지원은 최종적인 검증이 아니라 제도적 지원으로 취급해야 한다. AI가 생성한 모든 표현식은 여전히 구조화된 검증 방법을 사용하여 테스트해야 합니다.

2단계 - Cisco ESA의 필터 동작 검증

표현식 자체가 검증되면 두 번째 단계에서는 실제 메시지 처리에 적용할 때 Cisco ESA 내부에서 어떻게 동작하는지 확인하는 데 중점을 둡니다.

1. CES 콘솔에서 추적 기능 사용

관리자는 Cisco CES(Email Security) 콘솔의 추적 기능을 사용하여 특정 메시지가 처리되는 방식을 시뮬레이션하고 분석할 수 있습니다. 이는 시행 전에 필터 동작을 검증하는 가장 안정적인 방법 중

하나입니다.

추적 기능으로 다음에 대한 가시성 제공:

- 메시지 구문 분석 방법
- 평가되는 필터
- 조건이 트리거되는지 여부
- 규칙 실행 순서

ESA는 MIME 구문 분석, 헤더 정규화 및 콘텐츠 디코딩을 수행하므로 어플라이언스 내부의 동작은 외부 regex 테스트 툴과 다를 수 있습니다. 자세한 지침은 관리자가 Cisco 공식 문서를 참조해야 합니다.

https://www.cisco.com/c/en/us/td/docs/security/ces/ces_16-0-3/user_guide/b_ESA_Admin_Guide_ces_16-0-3/b_ESA_Admin_Guide_12_1_chapter_0101001.html

Trace(추적)를 사용하면 필터가 실제 처리 엔진 내에서 예상대로 작동합니다.

2. 로깅 작업으로 필터 생성

안전하고 권장되는 또 다른 접근 방식은 메시지를 삭제, 반송 또는 격리하는 것과 같은 적극적인 조치를 적용하는 대신 로깅과 같은 중단 없는 작업으로 필터를 구축하는 것입니다.

일치하는 항목을 로깅하도록 필터를 구성하면 관리자는 다음을 수행할 수 있습니다.

- 일치 빈도 관찰
- 예기치 않은 트리거 탐지
- 성능에 미치는 영향 확인
- 실제 트래픽 동작 분석

이 접근 방식은 프로덕션 트래픽 내에서 필터를 제어된 모니터링 단계에 효과적으로 배치합니다. 충분한 검증이 완료되고 동작이 올바른 것으로 확인되면 작업을 시행 모드로 안전하게 변경할 수 있습니다.

컨텐트 필터 및 사전의 표현식 소개

정규식이 올바르게 설계되고 검증되었으면 다음 단계는 Cisco ESA 내에서 정규식을 어떻게 입력하는지 이해하는 것입니다. 구문은 표현식이 콘텐츠 필터 조건에서 직접 구성되었는지 또는 사전 내에서 구성되었는지에 따라 약간 다르게 나타날 수 있습니다. 이러한 차이는 종종 혼란을 일으킨다.

콘텐츠 필터에서 표현식 소개

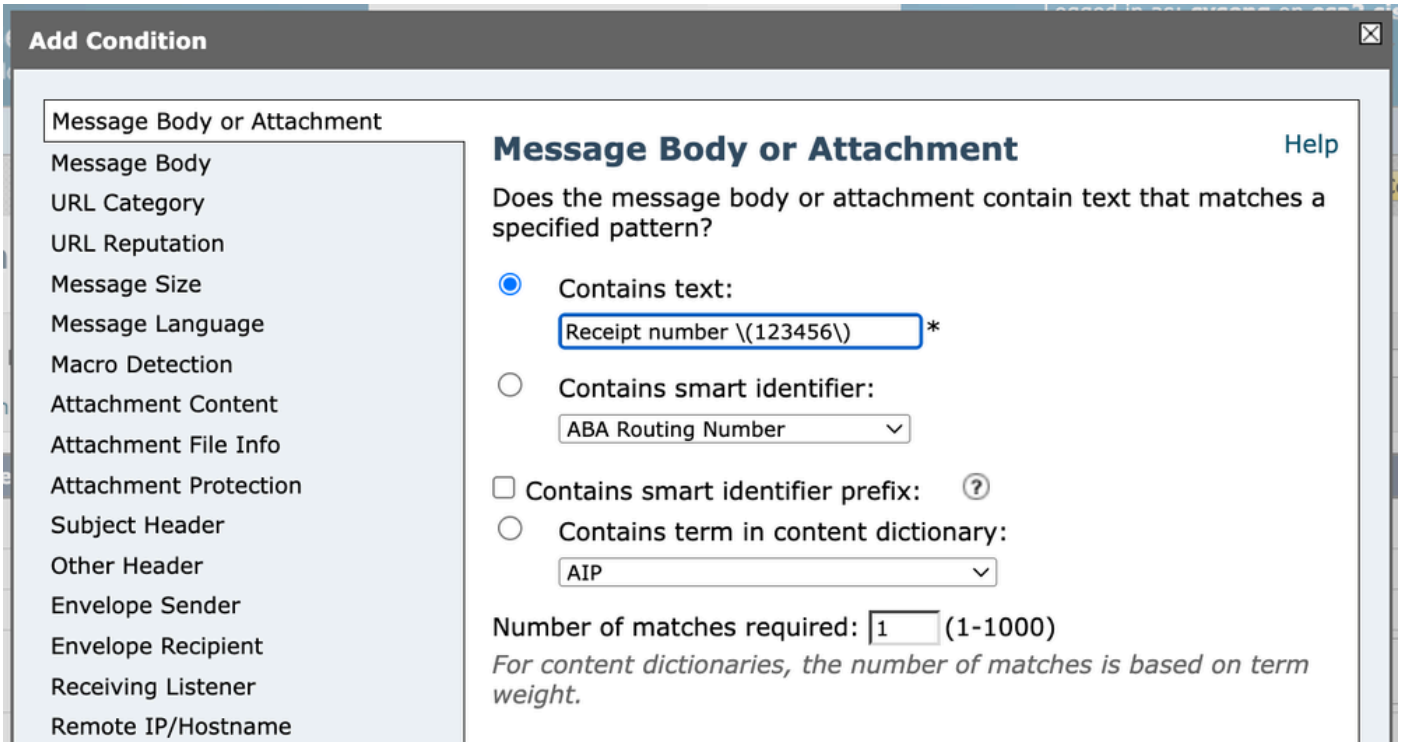
Content Filter 조건을 구성할 때(예: Subject 헤더와 일치) 조건 필드에 정규식을 입력해야 합니다. 리터럴 텍스트와 일치시키려면

```
Receipt number (123456)
```

괄호는 정규식에서 특수 문자이므로 괄호를 이스케이프해야 합니다.

따라서 regex 자체는 다음과 같이 작성되어야 합니다.

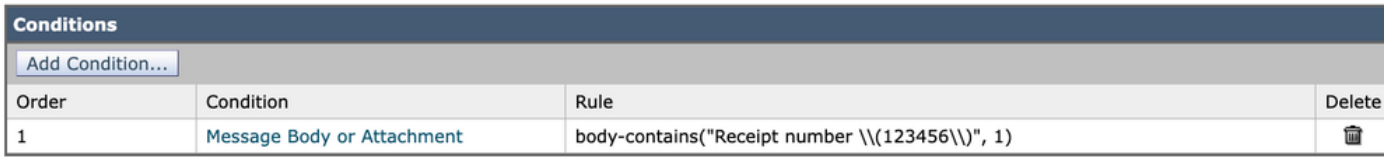
```
Receipt number \(123456\)
```



콘텐츠 필터 1

그러나 GUI 또는 고급 컨피그레이션 출력에서 전체 필터 조건을 볼 때 다음과 같이 표시될 수 있습니다.

```
subject == "Receipt number \\(123456\\)"
```



콘텐츠 필터 2

이것은 얼핏 보기에 혼란스러울 수 있다. 이중 백슬래시(\\)를 사용하는 이유는 백슬래시 자체도 따옴표 붙은 문자열 내의 특수 문자이기 때문입니다. 이 컨텍스트에서는 하나의 백슬래시를 사용하여 regex 엔진의 괄호를 이스케이프하고 두 번째 백슬래시를 사용하여 따옴표 붙은 문자열 내에서 백슬래시를 이스케이프합니다.

실제적인 관점에서

\(123456\)는 실제 정규식입니다.

\\(시스템이 따옴표 붙은 구성 문자열 내에서 \()을 나타내는 방법입니다.

표시되는 경우 다르게 표시되지만 평가 중인 논리 정규식은 그대로 유지됩니다.

수신 번호 \(123456\)

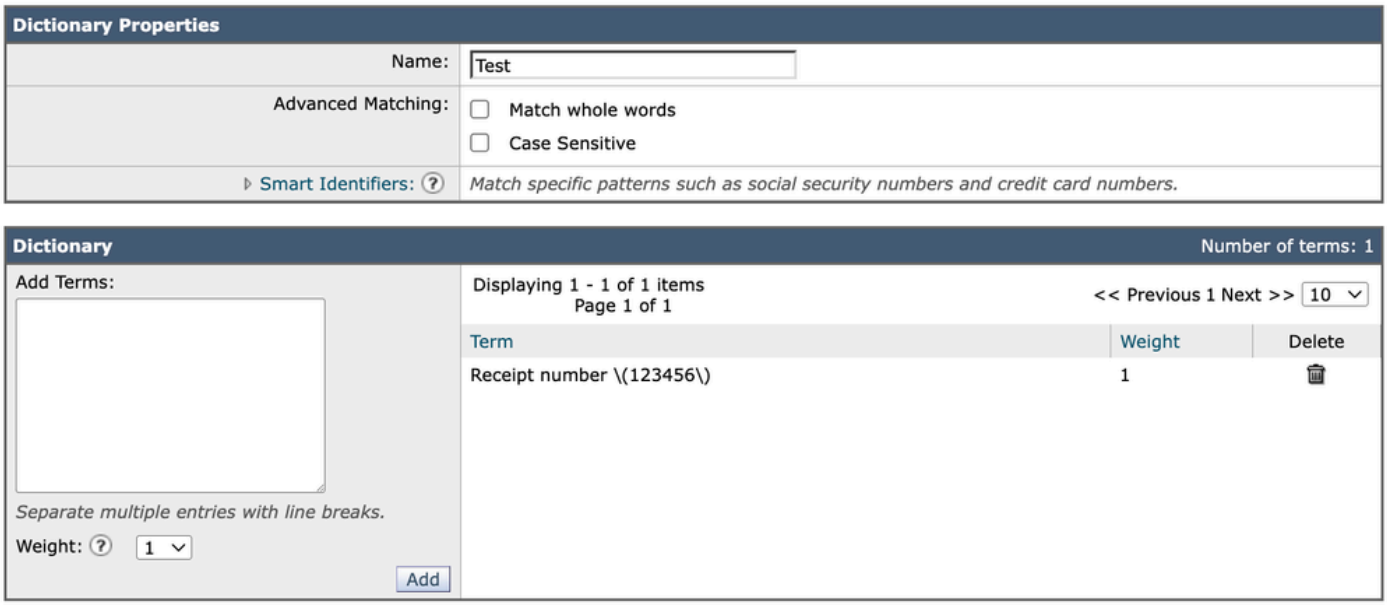
이는 컨피그레이션 출력에서 이스케이프되는 문자열의 문제일 뿐입니다.

사전의 식 소개

동일한 표현식을 사전에 추가하면 다음과 같이 항목이 직접 도입됩니다.

```
Receipt number \(123456\)
```

이 경우 작성된 그대로 계속 표시됩니다. 콘텐츠 필터 GUI 표현과 달리 사전은 시각적 구성 형식으로 추가 이스케이프 레이어를 필요로 하지 않습니다.



사전

각 사전 항목은 구조에 따라 일반 텍스트 또는 정규식으로 평가됩니다. 특수 문자가 포함된 경우(이 경우 괄호 등) 입력 시 식이 이미 제대로 이스케이프되어 있어야 합니다.

"전체 단어 일치" 정보

사전을 구성할 때 "전체 단어 일치"라는 옵션이 있습니다. 대부분의 경우 정규식으로 작업할 때 이 설정에 의존하지 않는 것이 좋습니다.

그 이유는 regex 앵커를 사용하여 워드-경계 거동을 보다 정밀하게 제어할 수 있기 때문이다.

예를 들면 다음과 같습니다.

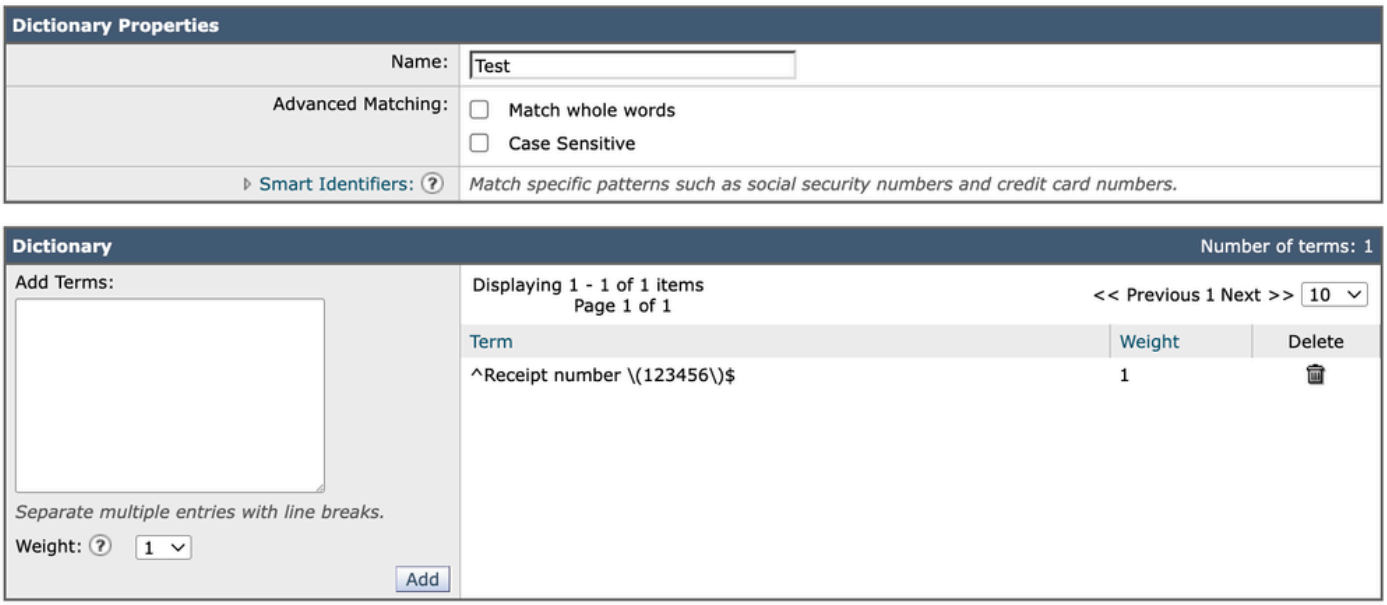
^는 일치가 처음부터 시작되도록 합니다.

$ 는 일치가 끝에 끝나도록 보장합니다.

다음과 같은 앵커 사용:

```
^Receipt number \(123456\)$
```

정확한 일치 동작에 대한 명확하고 예측 가능한 제어를 제공합니다. 이 접근 방식은 특히 다국어 또는 비 ASCII 환경에서 단어 경계가 해석되는 방식과 관련된 잠재적 모호성을 방지합니다.



사전 2

따라서 일반적으로 "전체 단어 일치" 옵션에 의존하지 않고 정규식 내에서 직접 일치 정밀도를 관리하는 것이 좋습니다.

컨텐트 필터와 사전 간의 이러한 미묘한 차이점을 이해하면 식이 일관되게 작동하고 구현 중에 구성 오류가 발생할 위험을 줄일 수 있습니다.

Cisco ESA의 Regex 비용 순위

Cisco ESA에서 정규식으로 작업할 때 성능에 미치는 영향은 엔진이 스캔해야 하는 텍스트 양과 수행해야 하는 백트랙킹 양에 따라 크게 달라집니다. ESA는 전체 메시지 본문, MIME 부분 및 디코딩된 첨부 파일까지 평가해야 하므로 비효율적인 패턴으로 인해 CPU 사용량이 크게 증가할 수 있습니다.

가장 높은 연산 비용부터 가장 낮은 연산 비용까지 실질적인 순위이다.

가장 비싼 고위험 패턴

이러한 표현은 특히 대형 메시지의 성능에 큰 영향을 미칠 수 있습니다.

중첩된 수량자(최악의 경우)

예:

```
(.*)+
(.+)+
(\S+)+
```

이는 기하급수적인 역추적 시나리오를 만들기 때문에 매우 위험합니다.

또 다른 한정자 안의 한정자는 regex 엔진이 많은 조합을 시도하도록 강제한다.

실제 트래픽에서는 이 경우 CPU 스파이크가 심각해질 수 있습니다.

권장 사항: 경계 없는 모호한 중첩 수량자는 사용하지 마십시오.

Greedy.* 다음에 필요한 패턴

예:

```
.*text
.*\/\?text
```

이 패턴은 먼저 전체 메시지를 사용한 다음 필요한 하위 문자열을 찾을 때까지 문자별로 문자를 백트랙합니다.

패턴이 없거나 끝 근처에 나타나는 경우 엔진은 많은 위치에서 필요한 토큰을 백트랙하고 테스트하므로 CPU 비용이 증가합니다.

본문이 크고 MIME 콘텐츠를 포함할 수 있는 ESA에서는 이 비용이 매우 빨리 듭니다.

권장 사항: 하위 문자열을 탐지하려면 .* 앞에 추가하지 마십시오. ESA는 평가된 콘텐츠를 이미 검색하며 선행 와일드카드는 백트랙킹 및 CPU 사용량만 늘립니다.

```
text$
\/\?text$
```

공유 접두사가 있는 큰 대체

예:

```
(a.*b|a.*c|a.*d)
```

여러 대안이 구조를 공유할 때 엔진은 각 브랜치를 순차적으로 평가합니다.

초기 브랜치가 거의 일치하지만 늦게 실패하면 엔진은 광범위하게 재시도합니다.

이렇게 하면 평가 시간이 크게 늘어납니다.

중간 비용 - 신중하게 사용

이러한 패턴은 재앙적이지 않지만 여전히 비효율적일 수 있습니다.

광범위한 * 사용

예:

```
https://.*\?text
```

기하급수적이지는 않지만 .*은 무제한 매칭을 허용합니다. 필요한 하위 문자열이 빠르게 나타나지 않으면 엔진은 메시지의 많은 부분을 검사합니다.

ESA에서는 이메일 본문에서 피싱 URL을 스캔할 때 흔히 사용됩니다.

지연 수량자(+?, *?)

예:

```
\S+?
.*?
```

지연 수량자는 일치 전략을 변경합니다(최단 우선). 일부 패턴에서는 오버매칭을 줄일 수 있지만, 대규모 '검색' 워크로드에서는 종료 토큰이 늦거나 누락되었을 때 시도를 늘릴 수 있습니다.

많은 ESA 활용 사례에서는 실질적인 혜택을 제공하지 않으며 불필요한 내부 재시도가 발생할 수 있습니다.

매우 일반적인 문자 클래스

예:

```
\S+
.+
```

이렇게 하면 일치 범위가 넓어져 잠재적 백트래킹 경로의 수가 늘어납니다.

보다 구체적인 문자 클래스는 항상 선호됩니다.

낮은 비용 — 안전하고 효율적인 패턴

프로덕션 ESA 환경에 권장됩니다.

고정 리터럴

예:

```
text
iw\.adc
```

리터럴 문자열은 가장 효율적인 일치입니다. 엔진은 최소한의 오버헤드로 간단한 비교를 수행합니다.

앵커를 사용하여 검색 범위 제한

특정 위치에서 일치가 예상되면 ^ 또는 $를 사용하여 패턴을 고정할 수 있습니다. 앵커들은 평가를 고정된 위치들로 제한하고, 엔진이 불필요하게 전체 콘텐츠를 스캐닝하는 것을 방지한다. 이를 통해 특히 대형 메시지 본문 또는 구조화된 헤더에서 백트레킹을 줄이고 성능을 개선할 수 있습니다.

```
^Invoice$
```

특정 문자 클래스

```
[A-Za-z0-9.-]+
[^/\s]+
```

이렇게 하면 일치 항목을 제한하여 검색 공간을 크게 줄이고 역추적을 제한할 수 있습니다.

구조화된 패턴 및 제한된 패턴

예:

```
https?:\/\/[A-Za-z0-9.-]+(?:\/[^?\s]*)*\/\?text
```

- 도메인이 수정되었습니다.
- .*을 사용하지 않습니다.
- 치명적인 중첩 패턴(예: (.*)+)을 포함하지 않습니다.
- 불필요한 게으른 연산자는 없습니다.
- 각 섹션은 제한됩니다.

이는 광범위한 와일드카드 매칭에 비해 CPU 영향을 크게 줄입니다.

Cisco ESA를 위한 실용적인 지침

메시지 또는 콘텐츠 필터에 대한 regex를 디자인할 때:

1. 패턴이 구체적일수록 성능이 우수합니다.
2. 꼭 필요한 경우가 아니면 .* 를 피하고, 특히 필요한 토큰을 뒤에 배치하는 것을 피하십시오.
3. 중첩된 수량자를 사용하지 마십시오.
4. 와일드카드보다 명시적 문자 클래스를 선호합니다.
5. 적용하기 전에 항상 모니터 모드에서 새 식을 테스트합니다.

Regex 성능 비교(Cisco ESA 컨텍스트)

패턴	권장	역추적 위험	ESA 영향	권장 대안
https?:\/\/.*/\?text.*	아니요	높음	더 높음	^https?:\/\/[A-Za-z0-9.-]+(?:\/[^?\s]*)*\/\?텍스트
https?:\/\/.*\?텍스트	⚠	보통-높음	보통-높음	^https?:\/\/[^?\s]+\?text$
https?:\/\/.*	아니요	보통-높음	중간	^https?:\/\/[A-Za-z0-9.-]+(?:\/[^\s]*)*
.*비밀번호	아니요	높음	더 높음	암호$
.*텍스트.*	아니요	높음	더 높음	텍스트
.*(송장|지급|이전)	아니요	높음	더 높음	(송장|지급|이전)$
(.+)+	전혀 없음	매우 높음(지수)	심각	중첩된 수량자를 사용하지 않고 재구성(예: .+)
.*@.*	아니요	높음	더 높음	[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}
\S+?	이상적이지 않음	중간	중간	\S+ 또는 [A-Za-z0-9.-]+와 같은 특정 클래스

.*\/관리자	아니요	높음	더 높음	\/admin$
.*(로그인|확인).*	아니요	높음	더 높음	(로그인|확인)
^.*텍스트	아니요	높음	더 높음	text$(또는 위치가 중요한 경우 ^text)

결론

정규식은 Cisco ESA 내에서 강력하고 유연한 도구로서 메시지 필터 및 콘텐츠 필터 모두에서 정확한 콘텐츠 검사 및 고급 정책 적용을 가능하게 합니다. 그러나 이러한 유연성에는 책임이 따릅니다. 잘못 설계되거나 충분히 테스트되지 않은 표현식은 오탐, 탐지 실패, 성능 저하 또는 의도하지 않은 정상적인 이메일 트래픽 중단으로 이어질 수 있습니다.

이러한 이유로 ESA에서 정규식을 사용하는 것은 항상 구조화되고 규율화된 접근 방식이어야 한다. 생성 단계에서는 표현식이 구문적으로 올바르고, 적절하게 이스케이프되고, 효율적이며, 의도한 목적과 논리적으로 정렬되도록 해야 합니다. 외부 툴과 AI 지원 세대는 이 프로세스를 크게 가속화할 수 있지만 신중한 검증을 대체해서는 안 됩니다.

ESA 환경 자체 내에서 검증 단계도 마찬가지로 중요합니다. ESA는 MIME 구문 분석, 헤더 표준화 및 콘텐츠 디코딩을 통해 메시지를 처리하므로 실제 동작은 이론적 기대와 다를 수 있습니다. 로깅 또는 모니터링 모드에서 초기에 추적 및 필터 구축과 같은 툴을 사용하면 관리자는 운영 위험 없이 올바른 동작을 확인할 수 있습니다.

요약하면, 정규식은 최대한 단순하게 유지하고 철저히 테스트하며 신중하게 배포해야 합니다. 잘 설계되고 올바르게 검증된 필터는 정책을 효과적으로 집행할 뿐만 아니라 시스템 안정성을 보호하고 프로덕션 환경에서 예측 가능한 동작을 보장합니다.

문서

Cisco ESA에서 정규식을 구현하고 사용하는 방법에 대한 추가 기술 세부 정보 및 공식 지침은 관리자가 Cisco 제품 설명서를 참조해야 합니다

"규칙의 정규식" 절에서는 규칙 조건 내에서 구문 고려 사항 및 사용을 포함하여 메시지 필터 및 콘텐츠 필터 내에서 정규식을 평가하는 방법에 대한 개요를 제공합니다.

https://www.cisco.com/c/en/us/td/docs/security/ces/ces_16-0-3/user_guide/b_ESA_Admin_Guide_ces_16-0-3/b_ESA_Admin_Guide_12_1_chapter_01000.html#con_1192755

"정규식 사용 지침" 섹션은 정확한 구문, 고정 표현식, 특수 문자 처리, 성능 또는 일치 정확성에 영향을 줄 수 있는 일반적인 실수 방지 등에 대한 실용적인 권장 사항을 제공합니다.

https://www.cisco.com/c/en/us/td/docs/security/ces/ces_16-0-3/user_guide/b_ESA_Admin_Guide_ces_16-0-

3/b_ESA_Admin_Guide_12_1_chapter_01000.html#con_1125696

정규식을 사용하는 필터를 설계하거나 트러블슈팅하는 경우 사용 중인 특정 AsyncOS 버전에 맞는 신뢰할 수 있는 지침을 제공하므로 이러한 공식 리소스를 검토하는 것이 좋습니다.

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.