URL 필터링에 대한 정규식 지침 및 성능 고려 사항

목차

소개

사전 요구 사항

- 요구 사항
- 사용되는 구성 요소

배경 정보

- 핵심 사항

피해야 할 패턴

권장 모범 사례

- 호스트 이름에서 항상 점 이스케이프
- 앵커 패턴 및 문자 제한
- 가능한 경우 중첩, 무제한 반복 방지
- PCRE2 호환 테스터의 테스트 패턴

HTTP 및 HTTPS에 대한 URL 일치의 차이점

- HTTPS(TLS) 트래픽
- HTTP(암호화되지 않은) 트래픽
- 컨피그레이션에 미치는 영향

다음을 확인합니다.

- 디버그 로깅 사용

컨피그레이션 예

- 호스트 기반 일치
- HTTP 호스트/경로 일치

관련 정보

소개

이 문서에서는 UTD 엔진을 통한 URL 필터링에서 정규식을 사용하기 위한 지침 및 성능 고려 사항에 대해 설명합니다. UTD 엔진의 URL 필터링에서는 PCRE2 정규식 라이브러리를 사용합니다.

기고자: Eugene Khabarov, Cisco Engineering.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- 정규식(regex) 구문

- URL 필터링 개념
- UTD(Unified Threat Defense) 컨피그레이션
- HTTPS/HTTP 프로토콜 차이

사용되는 구성 요소

이 문서는 특정 소프트웨어 및 하드웨어 버전으로 한정되지 않습니다.

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

배경 정보

PCRE2는 강력하지만 특정 복합 또는 '그리디' 표현식은 과도한 역추적을 유발할 수 있으며 Regex 엔진의 내부 제한에 도달할 수 있습니다. 이러한 현상이 발생하면 패턴은 처리하는 데 너무 많은 시간이 소요되어 결국 '일치하지 않음'으로 취급될 수 있습니다.

핵심 사항

- PCRE2는 시스템 리소스를 보호하기 위해 역추적 단계 또는 일치 시간에 대한 내부 제한을 적용합니다.
- 일부 패턴은 구문상 유효하지만 계산상 안전하지 않으며 '치명적인 역추적'을 트리거할 수 있습니다.
- 이러한 제한을 초과하면 URL이 패턴과 논리적으로 일치하는 경우에도 regex 엔진은 처리를 중단하고 일치하지 않는 결과를 반환할 수 있습니다.

피해야 할 패턴

다음과 같이 결합된 regex 구문은 사용하지 마십시오.

- 중첩된 수량자(예: (...+)*, (.*)*, (.+)+ 등
- 와일드카드(.)가 특히 패턴의 끝 부분에서 문자열의 많은 부분에 반복되었습니다.
- 반복과 함께 사용할 때 도메인 이름에 이스케이프되지 않은 점

예를 들어, 여기서 패턴은 구문상 유효하지만 처리하는 데 비용이 많이 들 수 있습니다.

```
^([a-zA-Z0-9-]+.)*portal.example.com$
```

참고: 이 경우 ([a-zA-Z0-9-]+.)*는 중첩된 한정자(+ inside *)와 와일드카드(.)가 있는 그룹입니다. 일부 일치하지 않는 입력에서 regex 엔진은 매우 많은 수의 백트래킹 경로를 탐색할 수 있습니다.

권장 모범 사례

호스트 이름에서 항상 점 이스케이프

리터럴 점을 일치시키려면\.를 사용합니다. 예를 들면 다음과 같습니다.

```
^([a-zA-Z0-9-]+\.)*portal\.example\.com$
```

앵커 패턴 및 문자 제한

백트래킹을 줄이기 위해 ^ 및 $를 사용하고, 호스트 레이블의 경우 [a-zA-Z0-9-]와 같이 필요한 문자로 제한합니다.

가능한 경우 중첩, 무제한 반복 방지

하나의 regex에서 모든 것을 다루려고 하는 복잡한 패턴보다는 단순한 구조를 선호한다. 하나의 매우 광범위한 표현 대신 몇 개의 특정 항목을 고려하십시오.

PCRE2 호환 테스터의 테스트 패턴

구축 전에 PCRE2 호환 환경에서 regex 패턴을 테스트하고 '치명적인 역추적' 또는 유사한 경고를 발생시키는 패턴을 방지합니다.

참고: regex 패턴이 PCRE2 엔진의 내부 제한에 도달하면 URL 필터링 엔진에서 'no match'로 처리할 수 있습니다. 이러한 경우 URL 분류는 화이트리스트/블랙리스트 regex 결과가 아니라 카테고리 또는 평판으로 돌아갑니다. 정확한 제한은 구현별로 다르며 릴리스 간에 변경될 수 있습니다. 당신은 보수적으로 디자인해야 합니다.

HTTP 및 HTTPS에 대한 URL 일치의 차이점

UTD 엔진은 HTTPS 및 HTTP 트래픽에 대해 URL을 서로 다르게 검사합니다. 이는 URL 필터링에 정규식을 디자인하는 방법에 영향을 줍니다.

HTTPS(TLS) 트래픽

암호화된 HTTPS 트래픽의 경우 UTD 엔진은 기본적으로 페이로드를 해독하지 않습니다.

- URL 필터링은 TLS(Transport Layer Security) ClientHello의 SNI(Server Name Indication)를 사용합니다.
- regex 패턴은 다음과 같이 SNI 호스트 이름에만 적용됩니다. api.example.com

이 경우 호스트 이름 기반 패턴은 다음과 같은 호스트 이름 문자열 api.example.com과 일치합니다.

```
^([a-zA-Z0-9-]+\.)*example\.com$
```

HTTP(암호화되지 않은) 트래픽

일반 HTTP 트래픽의 경우 UTD 엔진이 전체 HTTP 요청(요청 라인 및 헤더)을 볼 수 있습니다.

구현에 따라 regex 엔진에 제공되는 문자열에는 다음이 포함될 수 있습니다.

- 전체 URL 또는 요청 라인(예: GET /path?param=value HTTP/1.1) 또는
- 경로와 결합된 Host 헤더(예: api.example.com/path)

따라서 HTTP에 대한 regex 입력에는 기본 호스트 이름뿐만 아니라 /, ? 및 쿼리 문자열과 같은 추가 문자가 포함될 수 있습니다.

컨피그레이션에 미치는 영향

호스트 이름용으로 설계된 regex(예: 일치하는 api.example.com만 해당)는 HTTPS(SNI)와 정확히 일치할 수 있지만 전체 URL 또는 host+path 문자열이 포함된 HTTP 요청과는 일치하지 않습니다.

HTTP 및 HTTPS 트래픽을 동일한 패턴으로 필터링하려면 다음을 수행해야 합니다.

- 주로 호스트 이름을 중심으로 패턴 설계
- UTD 로그에서 HTTP 및 HTTPS 모두에 대한 동작 확인

다음을 확인합니다.

디버그 로깅 사용

1단계. 디버그 로깅을 활성화하려면 debug utd engine standard url-filtering level info 명령을 실행합니다.

2단계. show logging process ioxman module utd 실행 | 로그를 확인하기 위해 api.example.com 명령을 포함합니다.

출력 예:

```
2025/11/27 11:45:28.195000350 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF event->server_
2025/11/27 11:45:28.195001873 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF   URL: api.exa
2025/11/27 11:45:28.195009216 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF Regex matched
2025/11/27 11:45:28.195022442 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF URLF whitelist
2025/11/27 11:45:33.530605572 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF   URL: api.exa
2025/11/27 11:45:33.530606333 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF Regex not matc
2025/11/27 11:45:33.530614980 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF URLF whitelist
```

컨피그레이션 예

호스트 기반 일치

example.com의 모든 하위 도메인을 허용하려면 다음 권장 호스트 이름 중심 패턴(기준)을 사용하십시오.

```
^([a-zA-Z0-9-]+\.)*example\.com$
```

이 패턴:

- example.com, api.example.com, foo.bar.example.com 등과 일치
- HTTPS(SNI) 매칭에 적합
- 엔진에 표시되는 문자열이 베어 호스트 이름인 경우에도 HTTP와 일치시킬 수 있습니다

HTTP 호스트/경로 일치

HTTP에 host/path가 포함되어 있고 경로를 무시하려는 경우 호스트 이름 접두사를 일치시키고 regex가 후행 대신 단어 경계에서 중지되도록 할 수 있습니다. *예:

```
^([a-zA-Z0-9-]+\.)*example\.com\b
```

참고: 여기서 \b(단어 경계)는 명시적인 .* 와일드카드 없이 호스트 이름을 따르도록/또는 ?와 같은 문자를 효과적으로 허용합니다. 일반적으로 마지막에 .*를 추가하는 것보다 비용이 저렴하며 추가 무제한 와일드카드를 방지하기 위해 지침에 더 잘 맞습니다.

주의: HTTP 요청을 위해 regex 엔진에 전달되는 정확한 문자열은 구현에 따라 다르며 진화할 수 있습니다. 의심스러운 경우 랩 환경에서 HTTP 및 HTTPS 트래픽에 대한 패턴을 테스트하고 프로덕션에 구축하기 전에 UTD 로그에서 일치 여부를 확인합니다.

관련 정보

- Cisco Catalyst SD-WAN 보안 컨피그레이션 가이드, Cisco IOS XE Catalyst SD-WAN 릴리스 17.x
- Cisco 기술 지원 및 다운로드

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.