Cross-Origin Resource Sharing (CORS) begrijpen voor Finesse

Inhoud

Inleiding

In dit document wordt het delen van bronnen over de herkomst heen volledig beschreven, zodat tijdens probleemoplossing de onderliggende processen grondig worden begrepen.

Voorwaarden

Vereisten

Cisco raadt kennis van de volgende onderwerpen aan:

- Cisco Unified Contact Center Enterprise (UCS) release 12.6.x
- Cisco Packaging Contact Center Enterprise (PCE) release 12.6.x
- Cisco FineReader-softwarerelease 12.6.x
- Cisco Unified Intelligence Center (CUC) release 12.6.x

Gebruikte componenten

De informatie in dit document is gebaseerd op de volgende software- en hardware-versies:

- UCS release 12.6.2
- Finesse-softwarerelease 12.6.2
- CUC-release 1.6.2

De informatie in dit document is gebaseerd op de apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde (standaard)configuratie. Als uw netwerk live is, moet u zorgen dat u de potentiële

impact van elke opdracht begrijpt.

Achtergrondinformatie

Wat is CORS

Cross-Origin Resource Sharing (CORS) is een manier voor servers om te controleren welke websites (domeinen, protocollen en poorten) toegang hebben tot hun bronnen. Terwijl browsers normaal verzoeken van verschillende oorsprong blokkeren (het zelfde-oorsprongbeleid), geeft CORS servers de macht om selectief deze beperking te ontspannen. In essentie gebruiken servers speciale HTTP headers om de browser te vertellen welke oorsprongen zijn toegestaan, welke soorten verzoeken zijn toegestaan (zoals GET, POST, etc.), en welke aangepaste headers kunnen worden opgenomen. Dit laat servers beslissen wie toegang kan krijgen tot hun API's en hoe, variërend van volledig open tot strikt beperkte toegang. CORS werkt door de browser en server te laten communiceren via deze HTTP headers om verzoeken van verschillende oorsprong te beheren.

CORS maakt gebruik van HTTP-headers om gecontroleerde cross-origheidsverzoeken mogelijk te maken. De browser en de server communiceren via deze headers, waarbij de server de toegestane herkomst, methoden en headers specificeert. Als de reactiekoppen van de server ontbreken of ongeldig zijn, blokkeert de browser de respons, waarmee hetzelfde beleid van oorsprong wordt afgedwongen. Voor bepaalde verzoeken stuurt de browser eerst een preflight-verzoek naar de server om ervoor te zorgen dat deze het eigenlijke cross-origine verzoek accepteert.

Browsers gebruiken preflight verzoeken om te controleren als een server een cross-origine verzoek toestaat alvorens het echte verzoek te verzenden. Deze preflight verzoeken omvatten details zoals de methode HTTP en de douanekopballen. De Cors-toegelaten servers kunnen dan antwoorden, of het toelaten of het ontkennen van het daadwerkelijke verzoek. Als een server niet is geconfigureerd voor CORS, reageert hij niet correct op de preflight, en de browser blokkeert het eigenlijke verzoek, waardoor de server wordt beschermd tegen ongewenste toegang van verschillende oorsprong.

Cross-Origin Resource Sharing (CORS) is van cruciaal belang voor webbeveiliging en -functionaliteit. Het verleent gecontroleerde toegang tot middelen van verschillende oorsprong (domeinen, protocollen, havens), wat noodzakelijk is omdat browsers een zelfde-Originbeleid afdwingen dat normaal dergelijke toegang blokkeert.

Levenscyclus van een CORS

Een CORS-verzoek bestaat uit twee kanten: de client die het verzoek indient en de server die het verzoek ontvangt. Aan de clientzijde schrijft de ontwikkelaar JavaScript-code om het verzoek naar de server te verzenden. De server reageert op het verzoek door speciale CORS-specifieke kopregels in te stellen om aan te geven dat het verzoek om kruisbestuiving is toegestaan. Zonder de deelname van zowel de klant als de server mislukt het CORS-verzoek.

De belangrijkste spelers in een CORS-verzoek zijn de client, de browser en de server. De klant wil

wat gegevens van de server, zoals een reactie van JSON API of de inhoud van een webpagina. De browser fungeert als de vertrouwde intermediair om te verifiëren dat de client toegang heeft tot de gegevens vanaf de server.

Klant:

De client is een fragment van JavaScript-code dat op een website wordt uitgevoerd en is verantwoordelijk voor het initiëren van het CORS-verzoek

---

Opmerking: Finesse is een webapplicatie. Het is geïnstalleerd op een server, en de agenten hebben toegang tot het eenvoudig door hun webbrowsers te gebruiken, die de behoefte aan client-side installaties of onderhoud van plugins of andere software elimineren. Zoals is aangetoond in het voorbeeld CORS in Action with Cisco Finesse, ondersteunt deze architectuur functies zoals live datarapporten. In deze context fungeert de JavaScript-code van Cisco Finesse live data gadget als de client, terwijl Cisco CUIC fungeert als de server binnen de CORS-levenscyclus. In wezen, de browser-gebaseerde Finesse client interageert met de CUIC server om live data op te halen.

Klant tegenover gebruiker:

Soms worden de woorden client en gebruiker door elkaar gebruikt, maar ze zijn verschillend in de context van CORS. Een gebruiker is een persoon die een website bezoekt of een Finesse-gebruiker (agent of supervisor) die toegang heeft tot Finesse in deze context, terwijl een klant de feitelijke code is die door die website wordt bediend. Meerdere gebruikers kunnen dezelfde website bezoeken en dezelfde JavaScript-clientcode krijgen.

Browser:

De browser, ook bekend als de user agent, host de client-side code. Het speelt een cruciale rol in CORS door extra informatie toe te voegen aan uitgaande verzoeken, waardoor de server de client kan identificeren. Bovendien interpreteert de browser de reactie van de server, waarbij wordt bepaald of de gegevens aan de client moeten worden geleverd of dat een fout moet worden geretourneerd. Deze browserzijacties zijn essentieel voor het behoud van de beveiliging die wordt geboden door hetzelfde oorsprongbeleid. Zonder de handhaving van de CORS-regels door de browser kunnen klanten ongeoorloofde verzoeken doen, waardoor dit cruciale beveiligingsmechanisme in gevaar komt.

Server:

De server is de bestemming van het CORS-verzoek en het is CUIC voor het gadget voorbeeld Live-gegevens met Cisco Finesse. De server slaat de gegevens op die de client wil en heeft het laatste woord over de vraag of het CORS-verzoek is toegestaan of niet.

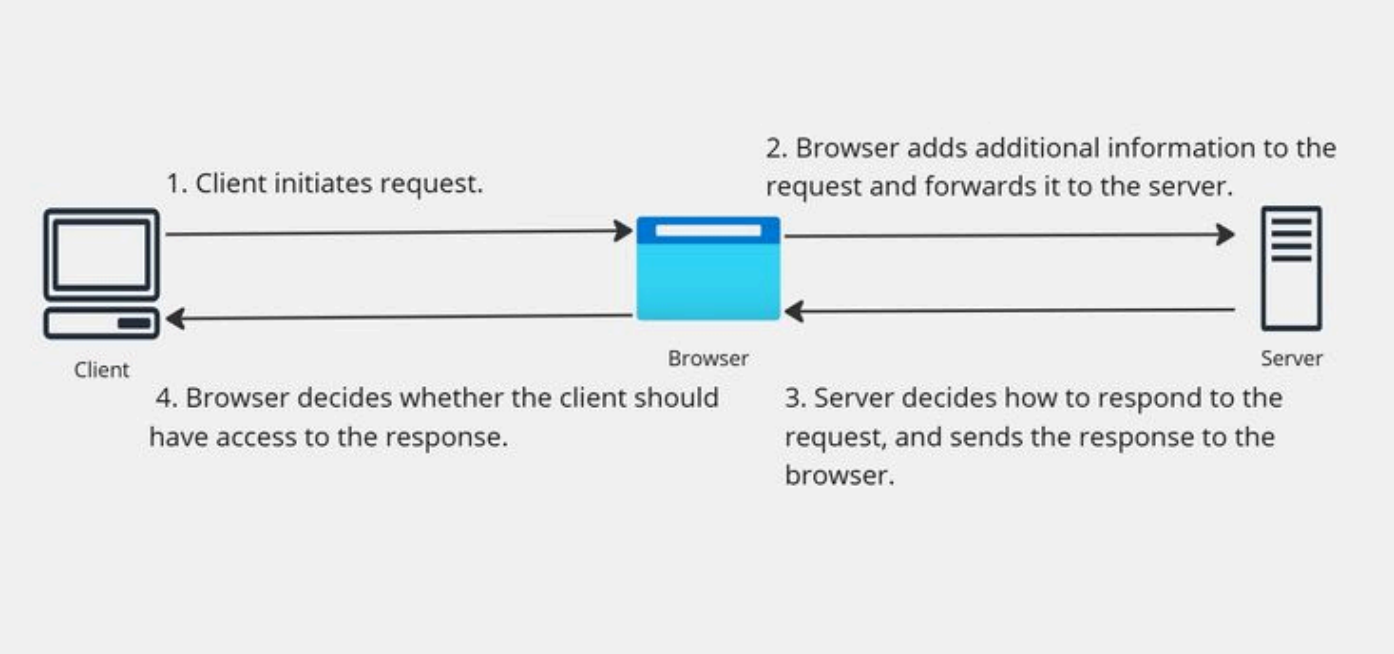
Nu u weet wie betrokken is bij een CORS-verzoek, laten we eens kijken hoe ze allemaal samenwerken. De volgende beelden illustreren de levenscyclus op hoog niveau van CORS:

1. De opdrachtgever initieert het verzoek.

2. De browser voegt extra informatie toe aan het verzoek en stuurt het door naar de server.

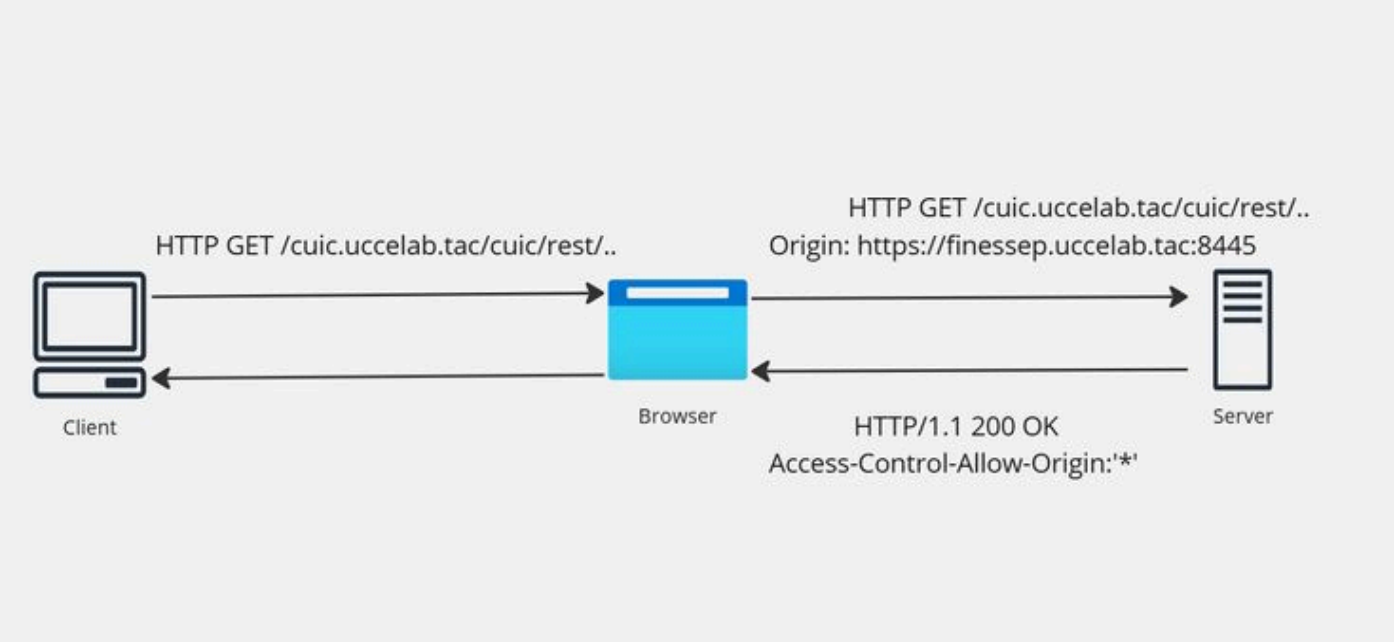
3. De server beslist hoe te reageren op het verzoek, en stuurt het antwoord naar de browser.

4. De browser bepaalt of de client toegang moet hebben tot de respons en of de reactie wordt doorgegeven aan de client of een fout wordt teruggegeven.

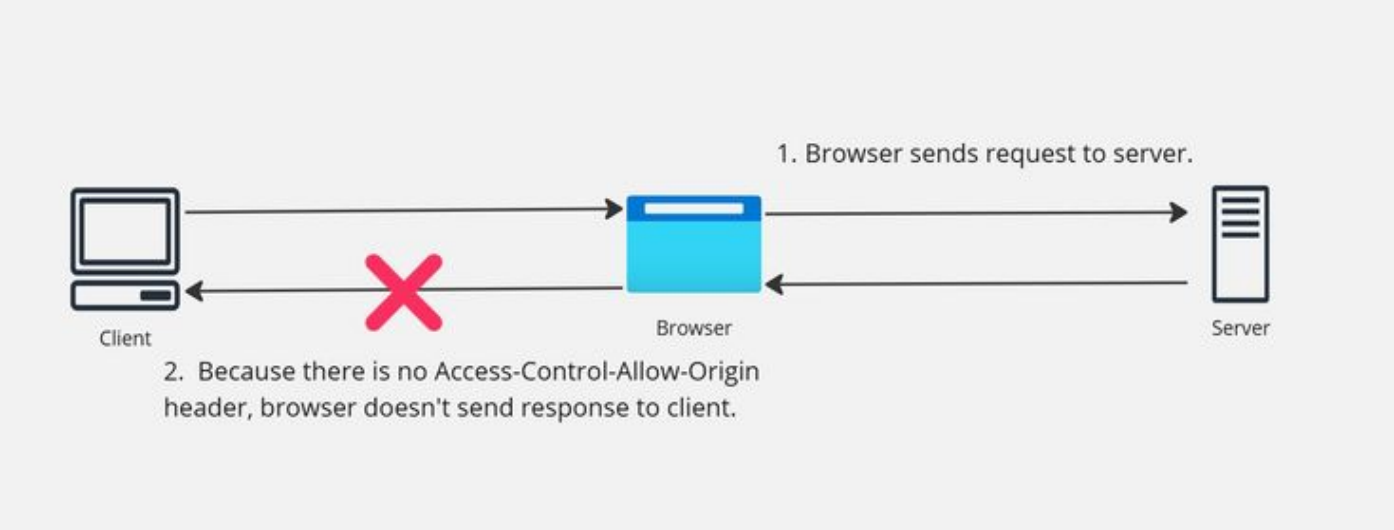


Alvorens een verzoek van de dwars-oorsprong te verzenden, voegt de browser automatisch een oorsprongskader aan het HTTP- verzoek toe. Deze header, die de client niet kan aanpassen, is een cruciaal onderdeel van CORS en dient om de oorsprong van de client te identificeren (dat wil zeggen het domein, protocol en poort waar de client resource is geladen). Deze veiligheidsmaatregel voorkomt dat cliënten zich als een andere herkomst kunnen voordoen. De oorsprong header is fundamenteel voor CORS, omdat het de manier is waarop de client de server vertelt waar het vandaan komt.

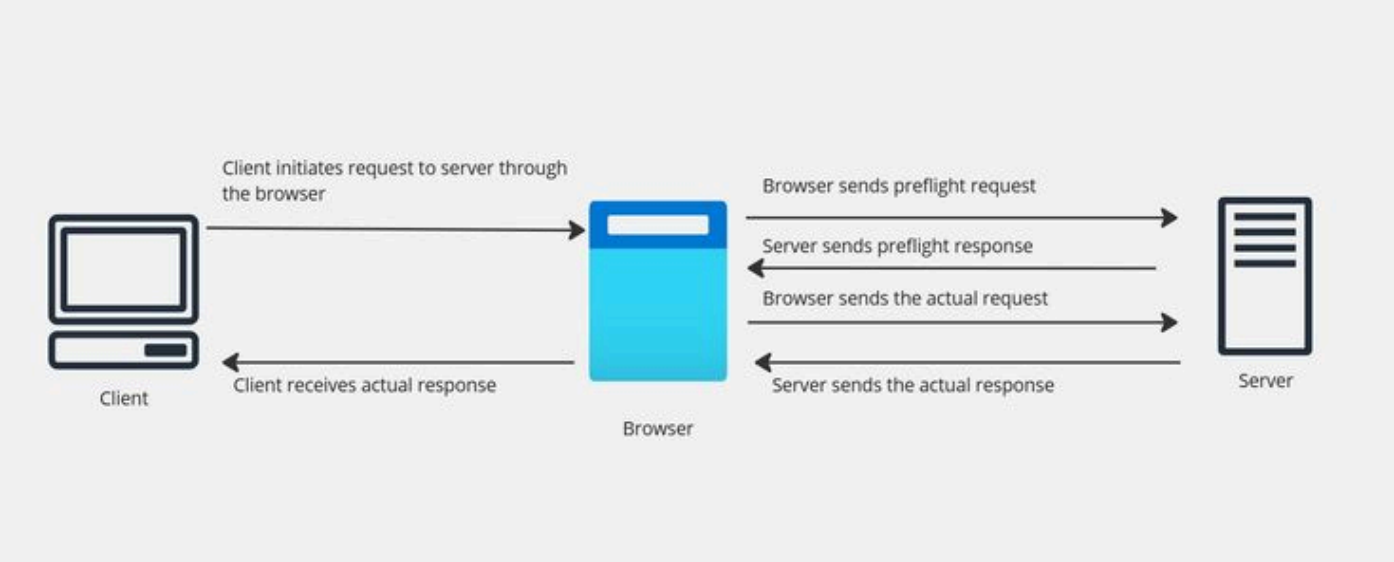
In een Cross-Origin Resource Sharing (CORS)-interactie wordt de oorsprong van de klant geïdentificeerd door de Origin-kop in het oorspronkelijke verzoek. De server gebruikt dan de Access-Control-Allow-Origin header in zijn antwoord om aan te geven of de client toegang heeft tot de gevraagde resource. Deze respons-header is van cruciaal belang; indien niet aanwezig, mislukt het CORS-verzoek. De header Access-Control-Allow-Origin kan een jokerteken (*) bevatten, waardoor toegang mogelijk is vanuit elke oorsprong, of een specifieke oorsprong, waarbij alleen toegang wordt verleend aan die bepaalde klant. Terwijl het beeld Access-Control-Allow-Origin toont: *, het impliceren van CUIC staat alle oorsprong toe, verzendt CUIC typisch deze kopbal met een specifieke oorsprong in real-world scenario's.



Als een browser een CORS-verzoek afwijst, betekent dit dat de client geen informatie ontvangt over de reactie van de server. De klant weet alleen dat er een fout is opgetreden, maar heeft geen details over het specifieke probleem. Dit kan debugging CORS fouten uitdagend maken, aangezien het moeilijk is om een CORS-fout te onderscheiden van andere soorten fouten. Zelfs als het eerste verzoek naar de server wordt verzonden, als de reactie van de server een geldige Access-Control-Allow-Origin header ontbeert, blokkeert de browser de reactie en veroorzaakt een fout aan de clientzijde, waardoor de client nooit de gedetailleerde reactie van de server ziet.



Deze afbeelding verklaart het hele CORS-proces, met bijzondere aandacht voor de preflight-fase, die essentieel is voor de afhandeling van specifieke soorten verzoeken om wederzijdse oorsprong.

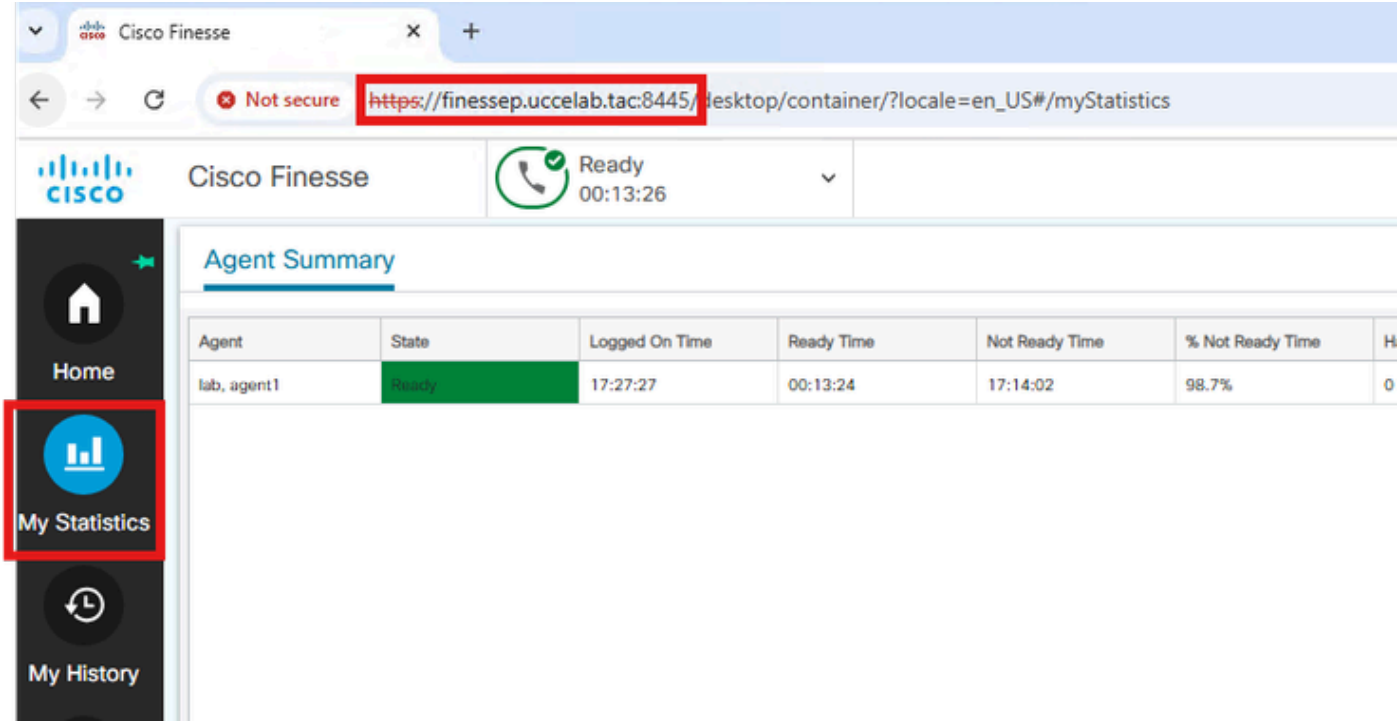


CORS in actie met Cisco Finesse

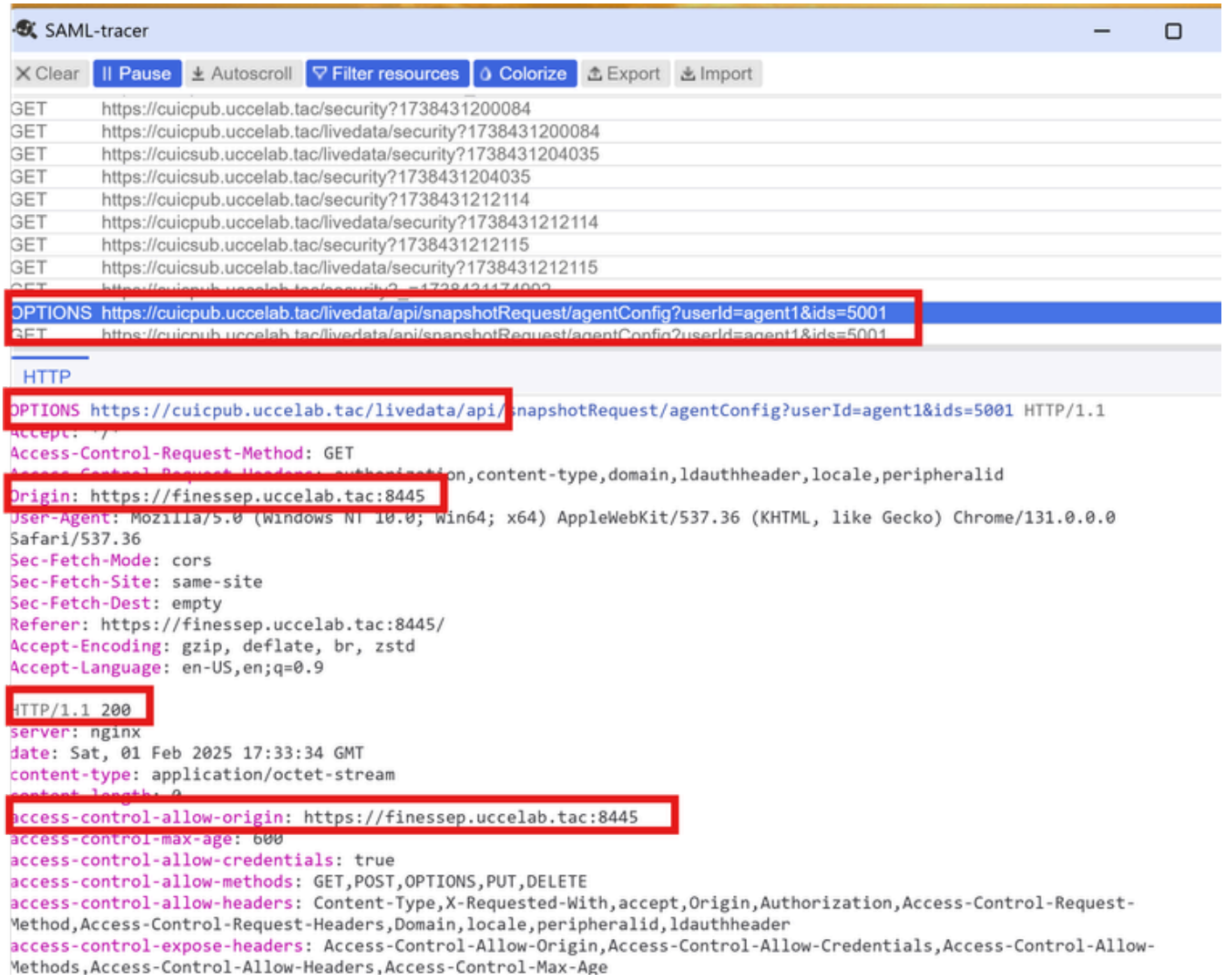
Voorbeeld: CORS-gedrag analyseren met de Live Data Gadget

In deze sectie wordt het typische gebruik van Cross-Origin Resource Sharing (CORS) met Cisco Finesse in contactcenters beschreven. Agents en supervisors gebruiken doorgaans Cisco Finesse voor toegang tot realtime gegevensrapporten (zoals geïllustreerd in de voorbeeldafbeelding).

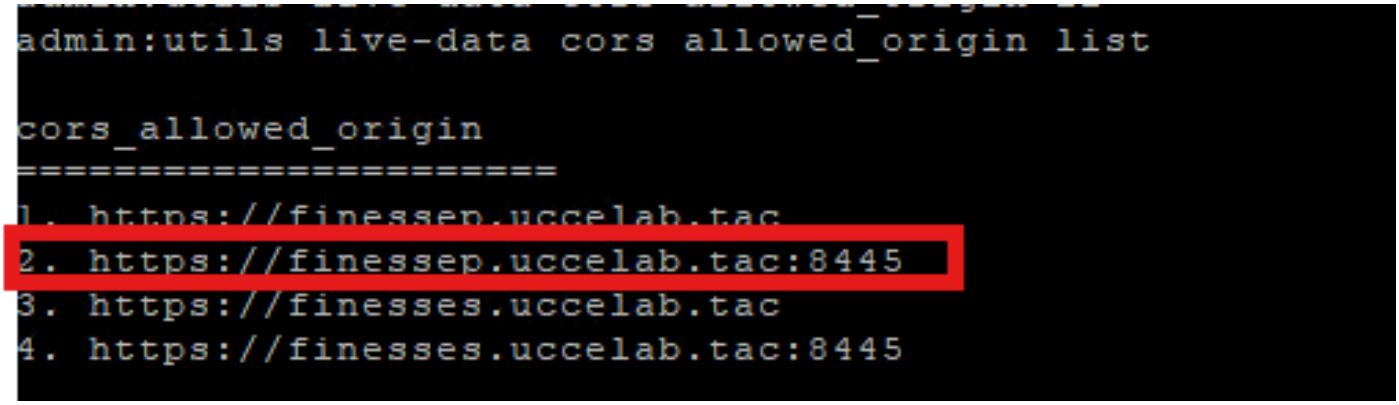
Wanneer een agent of supervisor een rapportgadget klikt, initieert hun actie een verzoek van de gegevensherwinning. Dit verzoek wordt verzonden van de Finesse-applicatie JavaScript-code (fungeert als client) naar de CUIC/Live Data-server met behulp van een GET-methode. Zoals aangetoond in de SAML tracer image, stuurt de browser eerst een preflight verzoek naar de server, de CORS levenscyclus eerder beschreven.

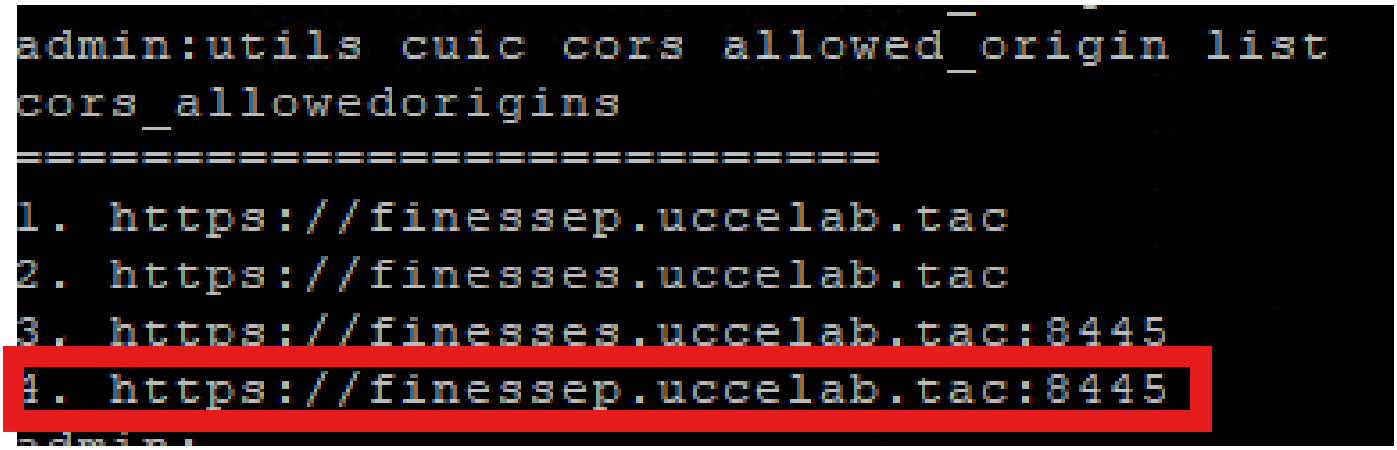


Een HTTP OPTIONS-verzoek (het preflight-verzoek) wordt naar de CUIC/Live Data-server verzonden. Dit verzoek specificeert de oorsprong als de volledig gekwalificeerde domeinnaam (FQDN) van de Finesse-server, inclusief poort 8445. Dit is hetzelfde adres en dezelfde poort als die worden gebruikt door agents voor toegang tot de Cisco FineReader-toepassing.



De commando's van de opdrachtregel interface (CLI) op de CUIC/Live Data server besturen welke origins toegang hebben tot hun live data resources. Als de oorsprong van de Finesse-server (zijn FQDN en poort) is geconfigureerd in deze instellingen, dan kunnen medewerkers de live data gadget details bekijken binnen Finesse.





TAC Tool voor CORS Connection Testing

MISCONFIGURATIES VAN CORS aan de serverkant kunnen soms problemen veroorzaken met gadgets van derden of live gegevens in Cisco Finesse. Dit artikel biedt een link naar een CORS Quick Check gadget, een tool voor probleemoplossing is ontworpen om te helpen bij het diagnosticeren van problemen met Cross-Origin Resource Sharing die van invloed zijn op Finesse gadgets, waaronder live datadisplays en andere integraties van derden.

Technisch gezien werkt dit gadget door preflight verzoeken van de Cisco Finesse-client naar een gespecificeerde doelbron te verzenden. Deze snelle controlefunctie helpt om Cors-gerelateerde problemen snel te identificeren en op te lossen, waardoor het proces van probleemoplossing wordt versneld.

U kunt als volgt het contactcentercors Quick Check 12.6-v1.0 gadget implementeren op het bureaublad van Finesse:

1. Download gadget bestanden van contactcenters CORS Quick Check 12.6-v1.0 map.2.

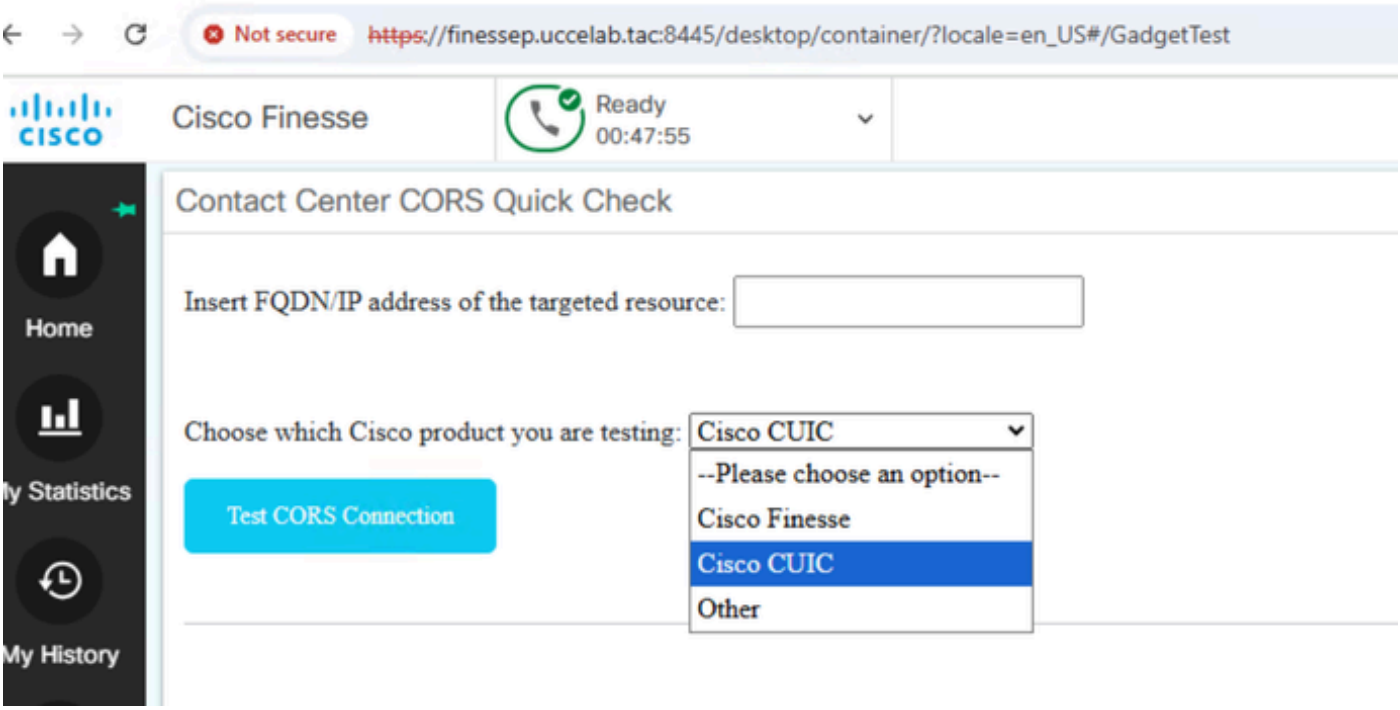
2. Kopieer de inhoud van de map Contactcenter CORS Quick Check 12.6-v1.0 naar de map 3rdpartygadget binnen uw Finesse-installatie.

3. Voeg de gadget toe aan de gewenste gebruikersrol (Agent, Supervisor, enzovoort) in de Finesse desktop lay-out. Het meegeleverde voorbeeld XML toont de juiste configuratie voor het toevoegen van dit gadget.

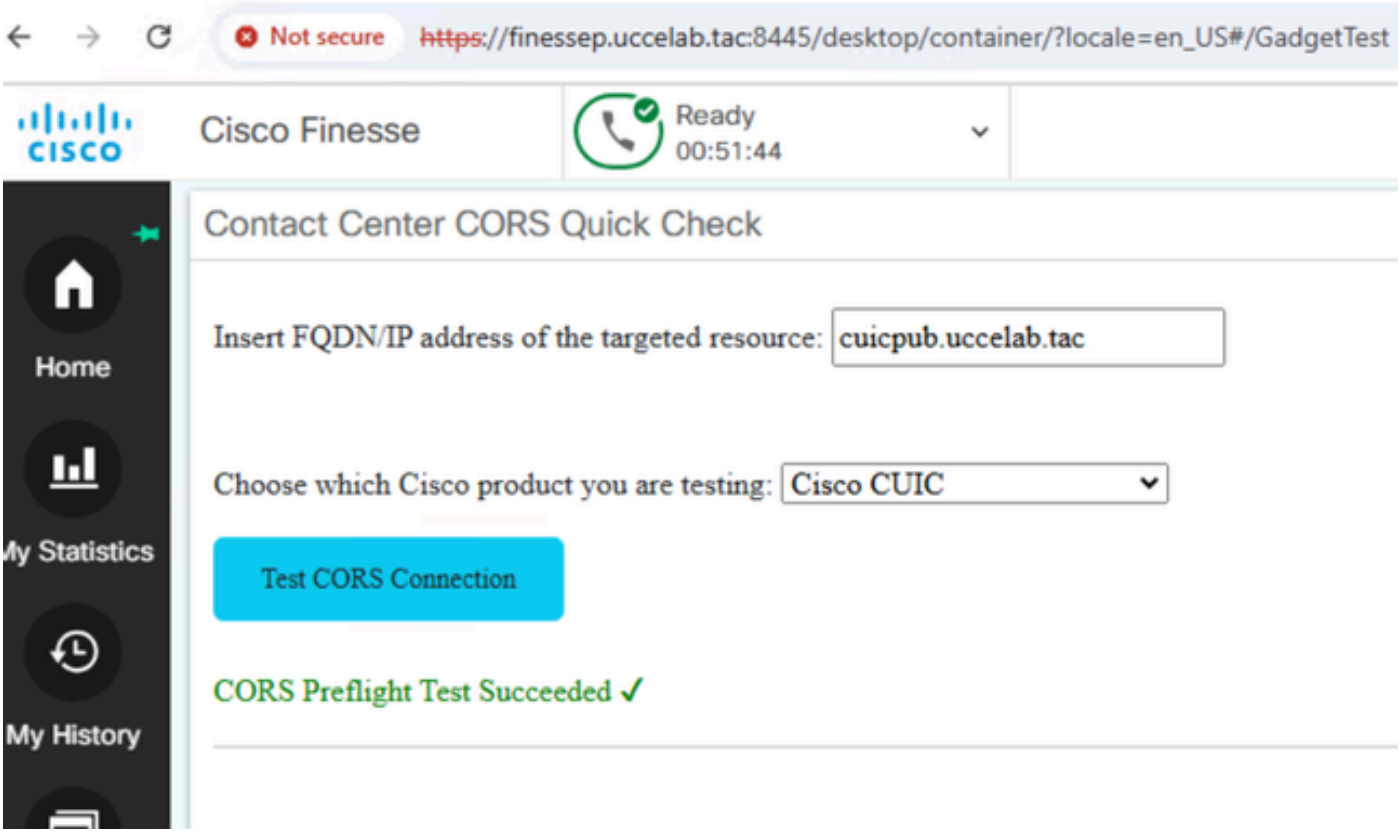
```
<gadget>/3rdpartygadget/files/TestCORSgadget.xml</gadget>
```

Zie het hoofdstuk Gadgets van derden in de Finesse Developer Guideen het hoofdstuk Gadgets van derden beheren in de Finesse Administration Guidevoor meer informatie over het uploaden van gadgets van derden en het toevoegen ervan aan de desktop.

Zodra de gadget files zijn geüpload en de Cisco Finesse Tomcat-service is herstart, is de gadget beschikbaar en geeft deze de grafische gebruikersinterface (GUI) weer.



U kunt CUIC selecteren in de vervolgkeuzelijst bovenaan. Voer in het daarvoor bestemde veld de volledig gekwalificeerde domeinnaam (FQDN) van de CUIC-server in. Een succesvolle test zal zijn zoals hier getoond.



Een succesvolle test betekent dat de CUIC-server correct is geconfigureerd voor Cross-Origin Resource Sharing (CORS) met de Finesse-server. De SAML-tracerlogboeken van de browser tonen aan dat een HTTP OPTIONS-verzoek (het CORS-preflight) naar de CUIC-server is verzonden. Deze aanvraag bevatte het adres van de Finesse-server in de kop Origin. De CUIC-

server reageerde met een 200 OK HTTP-bericht en, belangrijker nog, de Access-Control-Allow-Origin-header bevatte ook het adres van de Finesse-server. Hiermee wordt bevestigd dat de CUIC-server zodanig is geconfigureerd dat verzoeken van de oorsprong van de Finesse-server kunnen worden ingewilligd, waarbij wordt geverifieerd dat CORS correct is geïnstalleerd.

<#root>

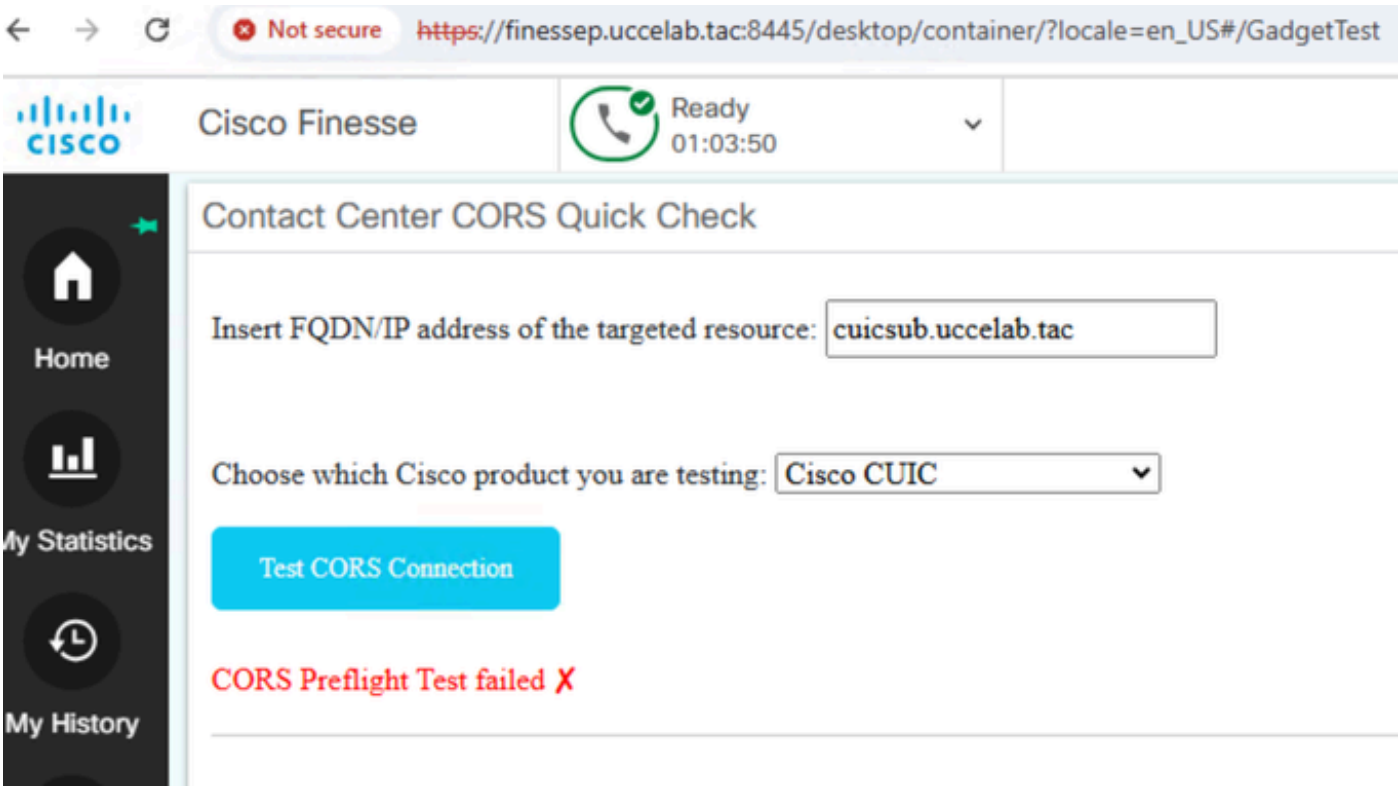
```
OPTIONS https://cuicpub.uccelab.tac/cuic/ HTTP/1.1

sec-ch-ua-platform: "Windows"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome..
sec-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A Brand";v="24"
sec-ch-ua-mobile: ?0
Accept: */*

Origin: https://finessep.uccelab.tac:8445

Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://finessep.uccelab.tac:8445/
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9
```

<#root>

```
HTTP/1.1 200

server: nginx
date: Sat, 08 Feb 2025 01:27:47 GMT
content-length: 0
strict-transport-security: max-age=31536000; includeSubDomains
set-cookie: JSESSIONID=bE73993C4A7C1Fc1b33A7AaF897B8428; Path=/cuic; Secure; HttpOnly; SameSite=Strict
pragma: No-cache
cache-control: no-cache
expires: Thu, 01 Jan 1970 00:00:00 GMT
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
content-security-policy: default-src 'self' ; script-src 'self' data: 'unsafe-inline' 'unsafe-eval' ; st
vary: origin,access-control-request-method,Access-Control-Request-Headers

access-control-allow-origin: https://finessep.uccelab.tac:8445

access-control-allow-credentials: true
access-control-expose-headers: access-control-allow-origin,access-control-allow-credentials,access-cont
access-control-max-age: 600
access-control-allow-methods: DELETE,POST,GET,OPTIONS,PUT
access-control-allow-headers: referer,peripheralid,origin,access-control-request-method,locale,accept,au
allow: GET,POST,OPTIONS,PUT,DELETE
```

In dit scenario toont het gereedschap een niet-werkende configuratie. In tegenstelling tot het vorige voorbeeld is de Finesse-server niet geconfigureerd als een abonnee op de CUIC-server. In plaats daarvan, wordt het slechts gevormd op de uitgever van CUIC. Als gevolg hiervan mislukt

het verzoek om een voorvlucht van CORS en reageert de CUIC-server met een HTTP 403 (Verboden) fout.



<#root>

OPTIONS https://cuicsub.uccelab.tac/cuic/ HTTP/1.1

Accept: */*
Access-Control-Request-Method: OPTIONS

Origin: https://finessep.uccelab.tac:8445

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome..
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Sec-Fetch-Dest: empty
Referer: https://finessep.uccelab.tac:8445/
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9

<#root>

HTTP/1.1 403

server: nginx
date: Sat, 08 Feb 2025 01:54:52 GMT
content-type: text/html;charset=utf-8
content-length: 2143
strict-transport-security: max-age=31536000; includeSubDomains
set-cookie: JSESSIONID=1C7606841B83d7847486c3d18D31cEfD; Path=/cuic; Secure; HttpOnly; SameSite=Strict
pragma: No-cache
cache-control: no-cache

```
expires: Thu, 01 Jan 1970 00:00:00 GMT
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
```

Zoals u kunt zien aan de uitvoer van de opdrachtregel voor CUIC-abonnees (CLI), wordt Cisco Finesse niet in de lijst weergegeven. Dit geeft aan dat Finesse momenteel niet is geconfigureerd als een abonnee op deze CUIC-server.

<#root>

```
admin:utils cuic cors allowed_origin list

cors_allowedorigins
===========================
1. https://finessep.uccelab.tac
2. https://finesses.uccelab.tac
3. https://finesses.uccelab.tac:8445
```

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.