Exemple de script Keepalive pour connecter/déconnecter un serveur Web SSL s'exécutant avec une négociation non chiffrée

Contenu

[Introduction](#)
[Conditions préalables](#)
[Conditions requises](#)
[Composants utilisés](#)
[Exemple de script](#)
[Informations connexes](#)

Introduction

Ce script connectera au Protocole SSL (Secure Socket Layer) un web server exécutant la version 3.0 SSL. Connectez au serveur, faites la prise de contact non chiffrée, et la déconnectez. Ce document adresse également l'implémentation des keepalives à base de script. Cette méthode de script le plus étroitement est liée à la fonctionnalité, qui est présente dans des clients distants du Remote Access Server (RAS), des programmes de terminal, et des utilitaires généraux de script. Cette caractéristique utilise le langage de script riche de WebNS.

Terminez-vous avec une interface de programmation simple de socket (API) (connectez/débranchement/send/receive), une keepalive à base de script donnera à l'utilisateur la capacité de travailler leur propre protocole, ou écrivez leur propre ordre des étapes pour fournir un `ACTIF` ou un état d'indisponibilité fiable d'un service. Sans fonctionnalité de keepalive à base de script, vous êtes actuellement limité au FTP, au HTTP, à l'ICMP, et au TCP. Avec des keepalives à base de script, cependant, vous pouvez rester sur les protocoles en cours à côté d'écrire vos propres scripts. Par exemple, vous pouvez développer un script spécifiquement modifié la tonalité pour se connecter à un serveur POP3 sans exiger de WebNS d'établir un type de keepalive POP3. Cette caractéristique permet à des clients pour créer leur propre Keepalives fait sur commande pour satisfaire à leurs exigences spécifiques. Bien que ce soit un composant du Commutateur de services de contenu (CSS), des scripts personnalisés ne sont pas pris en charge par le centre d'assistance technique Cisco (Cisco TAC).

Les keepalives à base de script ci-dessous ne sont pas officiellement prises en charge par TAC, mais ont été testées, et sont disponibles pour l'usage à votre propre discrétion.

Conditions préalables

Conditions requises

Connaissance de langage de script de riches de WebNS.

Composants utilisés

Les informations de ce document sont basées sur les versions de logiciel et matériel suivantes :

- Versions 3.x et ultérieures de WebNS
- Gamme 11x00 CSS

Les informations contenues dans ce document ont été créées à partir des périphériques d'un environnement de laboratoire spécifique. Tous les périphériques utilisés dans ce document ont démarré avec une configuration effacée (par défaut). Si votre réseau est opérationnel, assurez-vous que vous comprenez l'effet potentiel de toute commande.

Exemple de script

Le script ci-dessous peut être utilisé pour se connecter et déconnecter à une exécution de serveur Web SSL avec une prise de contact non chiffrée.

```
!--- No echo. !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! !--- Filename: ap-kal-ssl-port !--- Parameters: HostName ! !--- Description: !--- This script will connect to an SSL Web server running SSL !--- version 3.0. Connect to the server, do the non-encrypted !--- handshake, and disconnect. ! !--- Parameters: !--- SSL-IP: Address of the SSL Accellerator !--- SSL-Port: Port for the SSL Accellerator ! !--- Failure Upon: !--- 1. Not establishing a connection with the host. !--- 2. Not receiving a positive authentication. ! !--- Author: KGS !--- Last Tested: 9/27/01 ! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! if ${ARGS}[#] "==" "0" echo "Usage: ap-kal-ssl-port \'SSL-IP [SSL-Port]\'" exit script 1 endbranch !--- Defines. set HostName "${ARGS}[1]" set SSL-Port "443" if ${ARGS}[#] "GT" "1" set SSL-Port "${ARGS}[2]" endbranch !--- Connect to the remote host. set EXIT_MSG "Connection Failure for ${HostName}:${SSL-Port}" socket connect host ${HostName} port ${SSL-Port} tcp 2000 !--- Send the GET request for the Web page. set EXIT_MSG "Send: Failed" !--- Send over the hex for the fields: !--- [Handshake: 0x16] [Version: 0x03 0x00] [Length: 0x00 0x59] !--- [Client Hello: 0x01] [Length: 0x00 0x00 0x55] [Version: 0x03 0x00] !--- [Random (32bit) #: 0x39 -> 0xff] [Session Length: 0x20] !--- [Session ID (32bit): 0x3a -> 0x5d] [Cipher Length: 0x00 0x0e] !--- [Cipher Suite: 0x00 -> 0x00 (Last Byte in stream)] !--- Break the request into two send requests, as there is a 128 byte !--- max on quoted text parameters. socket send ${SOCKET} "16030000590100005503003 92ae5530da35d89041b4beaa42891470e49 351c3bfeb7631296139928dd7fff203a" raw socket send ${SOCKET} "9a0ed92a4e4f66d75ecce24c3a361efc26ab86310c4b9e7271a1317d9 7635d000e0004ffe0000a00640062000300060100" raw !--- Wait for a good status code. set EXIT_MSG "Waitfor: Failed" !--- Wait for a handshake message (0x16), paired with the version !--- of SSL (0x03 0x00). socket waitfor ${SOCKET} "160300" 2000 raw !--- Wait for the specific server hello (0x02). socket waitfor ${SOCKET} "02" 2000 raw !--- Wait for the version again (it appears twice: 0x03 0x00). socket waitfor ${SOCKET} "0300" 2000 raw no set EXIT_MSG socket disconnect ${SOCKET} exit script 0
```

Informations connexes

- Support matériel de Commutateurs de services satisfaits de gamme 11000 CSS
- Support matériel pour les commutateurs de services de contenu de la gamme CSS 11500
- **Téléchargement logiciel pour CSS11500 (clients enregistrés seulement)**
- Support et documentation techniques - Cisco Systems