S6 L3

Script dos in python

1.0 Introduzione

In questo esercizio viene analizzato uno script Python progettato per eseguire un attacco dos.
Lo script implementa una tecnica di DoS (Denial of Service) nota come UDP Flood, con l'obiettivo di saturare le risorse di rete o di elaborazione di un target specifico.
L'analisi mira a comprendere il funzionamento dei socket di rete e l'impatto di un traffico massivo non richiesto.

2.0 Analisi del Meccanismo Tecnico

Lo script utilizza la libreria standard socket di Python per interagire con lo stack di rete a basso livello.

2.1 Configurazione e Input

Il programma richiede all'utente tre parametri fondamentali:

- Target IP: L'indirizzo IPv4 della vittima (es. un sistema Windows 10 in laboratorio).
- Target Port: La porta di destinazione (es. 80 o 445).
- Packet Count: Il numero totale di pacchetti da inviare per determinare l'intensità dell'attacco.

2.2 Creazione del Socket e Payload

L'attacco si basa sui seguenti passaggi tecnici:

- Socket UDP: Viene inizializzato un socket utilizzando socket.AF_INET (IPv4) e socket.SOCK_DGRAM (UDP). A differenza del protocollo TCP, l'UDP non richiede una connessione (handshake), rendendo l'invio dei pacchetti estremamente rapido e privo di overhead di verifica.
- Generazione Payload: Viene creato un payload casuale di 1024 byte tramite random._urandom(1024). Questo serve a massimizzare l'occupazione della banda durante il transito dei dati.

2.3 Esecuzione del Flood

Attraverso un ciclo for, lo script esegue il metodo sock.sendto(bytes_payload, (target_ip, target_port)). Questo invia ripetutamente il payload al target senza attendere alcuna risposta, saturando potenzialmente la coda di ricezione del sistema di destinazione.

3.0 Analisi del Rischio e Impatto

Questa attività è classificabile nel framework MITRE ATT&CK come T1498.001 (Network Denial of Service: Direct Exploitation).

- Saturazione della Banda: L'invio massivo di pacchetti da 1KB può consumare rapidamente la capacità della rete locale di laboratorio.
- Saturazione Risorse Host: Il sistema target deve processare ogni pacchetto in arrivo per determinare se esiste un'applicazione in ascolto sulla porta specificata. Questo processo consuma cicli di CPU e memoria.
- Indisponibilità dei Servizi: Se il target è un server che fornisce servizi reali, gli utenti legittimi potrebbero riscontrare latenze elevate o l'impossibilità totale di connettersi.

4.0 Indicatori di Compromissione (IOC) e Difesa

Un analista SOC o un amministratore di rete può identificare questo attacco tramite i seguenti IOC:

- Traffico UDP Anomalo: Picchi improvvisi di traffico UDP verso una singola porta da un unico IP sorgente.
- Packet Size Costante: Flusso di pacchetti con dimensione fissa (es. 1024 byte), tipico di script di attacco non ottimizzati.
- ICMP Destination Unreachable: Se la porta target è chiusa, il sistema vittima potrebbe rispondere con una tempesta di messaggi ICMP Type 3, aumentando ulteriormente il carico di rete.

Contromisure Tecniche

- Rate Limiting: Configurazione di soglie di traffico UDP sui firewall o switch di rete.
- Intrusion Detection Systems (IDS): Regole per rilevare pattern di "UDP Flood" basati sulla frequenza dei pacchetti.
- Disabilitazione Porte Inutilizzate: Ridurre la superficie di attacco chiudendo i servizi non necessari.

5.0 Conclusione

L'esercizio dimostra come poche righe di codice Python possano generare un volume di traffico sufficiente a compromettere la disponibilità di un sistema in un ambiente non protetto. Mentre il contenuto dell'attacco è tecnicamente semplice (non richiede autenticazione o bypass complessi), la sua efficacia risiede nella natura stessa del protocollo UDP. La

resilienza contro tali minacce richiede un monitoraggio attivo dei flussi di rete e l'implementazione di controlli infrastrutturali per prevenire l'esaurimento delle risorse.