BOLETIM DE SEGURANÇA

Falha crítica do Tinyproxy expõe mais de 50.000 hosts expostos

heimdall
security research
A DIVISION OF ISH

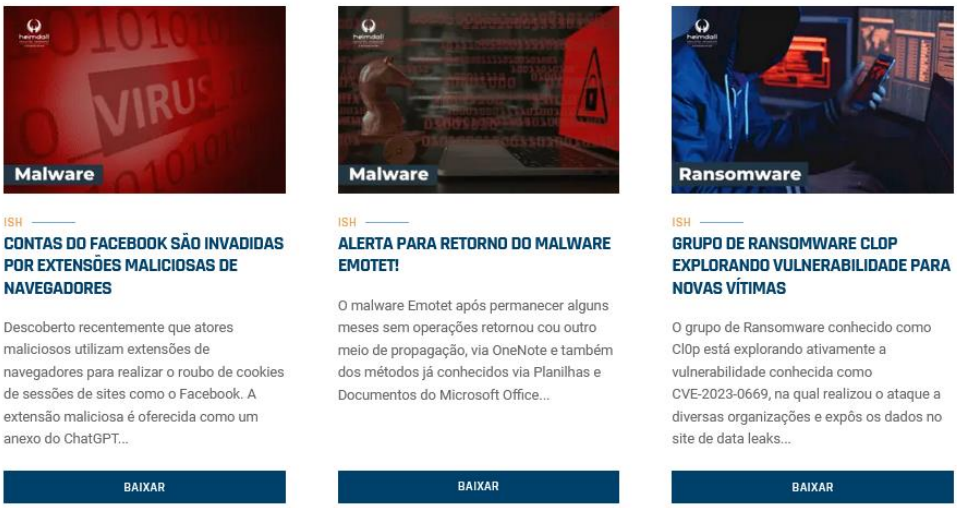
TLP: CLEAR

Receba alertas e informações sobre segurança cibernética e ameaças rapidamente, por meio do nosso **X**.

[Heimdall Security Research]

Acesse boletins diários sobre agentes de ameaças, *malwares*, indicadores de comprometimentos, TTPs e outras informações no *site* da ISH.

[Boletins de Segurança – Heimdall]



SUMÁRIO

LISTA DE FIGURAS

1 SUMÁRIO EXECUTIVO

A Cisco Talos divulgou agora em maio detalhes sobre a vulnerabilidade identificada como CVE-2023-49606. Esta falha, encontrada nas versões 1.11.1 e 1.10.0 do Tinyproxy, permite que atores maliciosos executem um use-after-free de memória, resultando em comportamentos inesperados do software.

Figura 1 – Mapa de hosts expostos com Tinyproxy.

2 DETALHES SOBRE A VULNERABILIDADE

O Tinyproxy, é um serviço de proxy HTTP/S de código aberto adequado para sistemas UNIX, ideal para pequenas redes, como as utilizadas por usuários individuais ou pequenas empresas. Entretanto, organizações maiores que utilizam este serviço para testes ou desenvolvimento é crucial que não o deixem acessível pela Internet pública. O impacto dessa vulnerabilidade é significativo, pois um atacante não autenticado pode causar uma corrupção de memória por meio de um cabeçalho de conexão HTTP manipulado, levando a uma negação de serviço (DoS) e, em alguns casos, até mesmo à execução remota de código (RCE).

Conforme a especificação HTTP, o cabeçalho **Connection**, quando enviado pelo cliente, indica quais cabeçalhos HTTP devem ser excluídos pelo proxy na requisição final. O proxy, então, elimina esses cabeçalhos da requisição, realiza a solicitação ao servidor remoto e retorna a resposta ao cliente. Essa operação assegura que apenas os cabeçalhos necessários sejam mantidos, otimizando a comunicação entre cliente e servidor.

```
static int remove_connection_headers (orderedmap hashofheaders)
{
        static const char *headers[] = {
                "connection",
                "proxy-connection"
        };

        for (i = 0; i != (sizeof (headers) / sizeof (char *)); ++i) {
                /* Look for the connection header.  If it's not found, return. */
                data = orderedmap_find(hashofheaders, headers[i]);                    (1)

                if (!data)
                        return 0;                                                     (2)

                                ...

                ptr = data;

                while (ptr < data + len) {
                        orderedmap_remove (hashofheaders, ptr);                       (3)
                                                ...
                }

                /* Now remove the connection header it self. */
                orderedmap_remove (hashofheaders, headers[i]);                        (4)
                }

                return 0;
}
```

Figura 2 – Cabeçalho HTTP Connection.

Inicialmente, é importante notar que os cabeçalhos HTTP que o cliente envia estão armazenados no '**hashofheaders**', que é uma estrutura de armazenamento chave-valor. O código identifica os cabeçalhos **Connection** e **Proxy-Connection** dentro de 'hashofheaders' e extrai seus valores em (1), que são listas de cabeçalhos HTTP designados para exclusão. Em (3), procede-se à remoção de cada cabeçalho HTTP especificado pelo cliente. De fato, cada cabeçalho HTTP mencionado nos valores dos cabeçalhos Connection e Proxy-Connection serve como chave para a sua eliminação do 'hashofheaders'. Por fim, em (4), os próprios cabeçalhos HTTP Connection e Proxy-Connection são removidos.

A função `orderedmap_remove()` opera da seguinte maneira:

```
int orderedmap_remove(struct orderedmap *o, const char *key) {
        htab_value *lv, *v = htab_find2(o->map, key, &sk);                    (5)
                ...
        sv = sblist_get(o->values, v->n);                                     (6)
        free(*sv);
                ...
        htab_delete(o->map, key);                                             (7)
                ...
}
```

Figura 3 – Função orderedmap_remove().

Para uma dada chave específica, o processo inicia com o cálculo do seu hash em (5). Utilizando esse hash, o ponteiro correspondente ao valor da chave é encontrado e desalocado em (6). A etapa final envolve a remoção da chave do hashmap em (7).

Analisando o cenário onde o cliente envia o cabeçalho HTTP Connection, diferenciados aqui como ConnectionA e ConnectionB para exemplificação, o valor de ConnectionA é obtido em (1), que é ConnectionB. Posteriormente, em (3), ConnectionB é utilizado como a chave na função 'orderedmap_remove()'. O hash de ConnectionB é calculado em (5), sendo idêntico ao de ConnectionA, e o sistema não faz distinção entre maiúsculas e minúsculas. Com o hash, o ponteiro para o valor de ConnectionA, que é ConnectionB, é recuperado e liberado em (6). Neste momento, a memória alocada para ConnectionB foi liberada. Em (7), a chave, que agora contém um ponteiro obsoleto para ConnectionB, é reutilizada, resultando em um cenário de uso após liberação (Use-After-Free). Essa vulnerabilidade é um risco evidente, pois pode ser explorada para corromper a memória e ganhar privilégios de execução de código. É importante notar que uma situação semelhante pode ocorrer com o cabeçalho Proxy-Connection. Contudo, a função 'remove_connection_headers()' verifica primeiro o cabeçalho Connection e, na sua ausência, o código é encerrado em (2). Portanto, como alternativa ao Connection: Connection, o cliente também pode enviar o cabeçalho Proxy-Connection para desencadear o problema.

```
Connection: [AnyValueHere]
Proxy-Connection: Proxy-Connection
```

Figura 4 – Proxy-Connection.

Abaixo está o resultado do ASAN em uma prova de conceito realizada pela Talos:

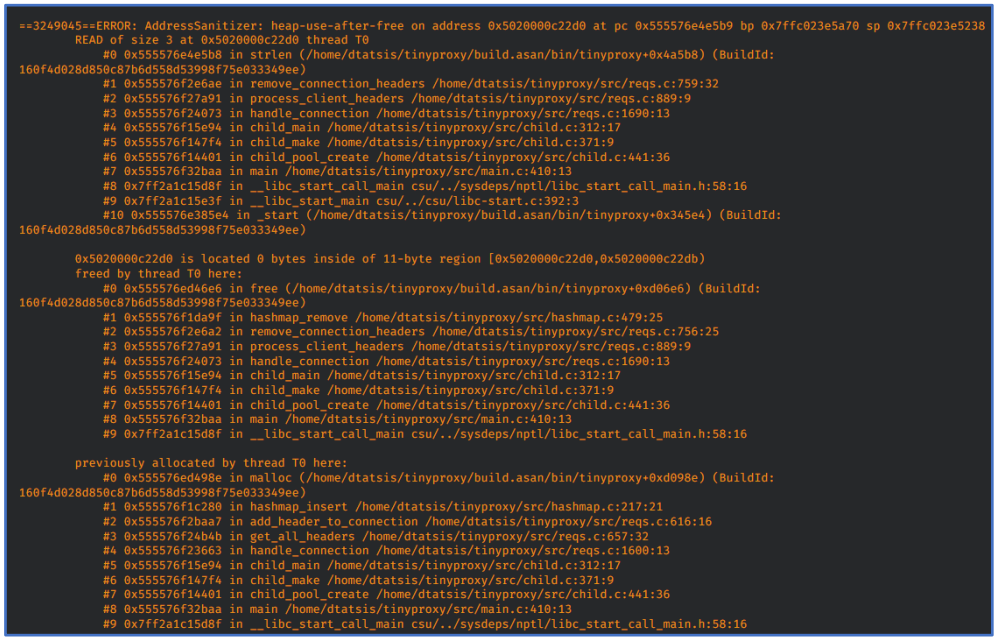


Figura 5 – Resultado do ASAN.

Durante a compilação padrão do daemon, ocorre uma falha devido a uma desalocação dupla na função 'orderedmap_destroy()'. Essa falha de segurança consiste em um pedido HTTP extremamente básico.

```
free(): double free detected in tcache 2
IOT instruction  ./src/tinyproxy -d -c ../tinyproxy-fuzz/tinyproxy.config
```

Figura 6 – Compilação do daemon.

```
GET / HTTP/1.1
Connection: Connection
Host: 192.168.86.166:8000
```

Figura 7 – Variação do http.

```
GET / HTTP/1.1
Connection: keep-alive
Proxy-Connection: Proxy-Connection
Host: 192.168.86.166:8000
```

Figura 8 – Alternativa utilizada do http.

3 RECOMENDAÇÕES

De acordo com o mantenedor, nenhum patch encontra-se disponível. No entanto a Censys, informa que para remediar este problema, é recomendável garantir que o usuário não exponha um serviço Tinyproxy à Internet pública, principalmente se ele estiver em uso em um ambiente de desenvolvimento ou teste.

4 REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia**
- Cisco_Talos
- Censys
- Thehackernews
- NVD

5 AUTORES

- **Leonardo Oliveira Silva**

heimdall
security research
A DIVISION OF ISH