DDoS-skydd för webben etapp3 (under utveckling)

Status

Utreda och implementera Sunet Frontend som DDoS-skydd.

Bakgrund och förutsättningar

Behovsbilden kretsar kring att det idag sker skyddsarbetet i huvudsak reaktivt; utöver de fasta filter som implementerats. Riktade attacker är svåra att upptäcka om de inte är volymetriska.

Projektleveranser

1. Pilotmiljö installerad för SUNET frontend som DDoS-skydd internt samt hos minst ett lärosäte
2. Definierat en tjänstemodell om tidigare punkter anses lyckade för SUNET DDoS-skydd med möjligtvis två tjänstenivåer. Första för stöd och analys, andra för skydd av exponerade system.
3. Etablerat ett första tjänstekontrakt och genomfört handover till Sunet NOC

Leverabler etapp 1

1. Projektplan
2. Projektdeltagare rekryterade och projektet organiserat
3. Kommunikationsinsatser
4. Kravspecifikation-underlag till upphandling/ utveckling

Etapp 3

Test implemenetation SUNET Frontend interna system.

Pilot installation Sunet Frontend hos Sunetansluten organsiation

Beskrivning av aktiviteter etapp 3

Nr	Aktiviteter
1	Definiera tekniska förutsättningar och dokumentera skiss av arkitektur för nya användningsområdet. Definiera steg för steg hur implementationen av Sunet Frontend hanterar externt tillgängliga resurser.
2	Dokumentera risker och juridiska förutsättningar för etablering av tjänsten
3	Förankring och försäkring om nätkapacitet och tekniska resurser för tjänsten
4	Initial test av SUNET Frontend som DDoS-skydd för utpekade Sunet tjänster (definieras under projektet vilka som bör ingå www.sunet.se exempelvis)
5	Pilot-implementation hos Sunetansluten organisation (KTH?)
6	Initial rapport och delgivande till Styrgrupp, ITCF, Säkerhetcenter Öppet hus / alt Sunetdagarna
7	Beslutspunkt om arbetet har förutsättningar att fortsätta inför en tjänsteetablering och förvaltning
8	Definiera tjänstemodell och avtal i samverkan med deltagande pilot-organisationer och ref samt styrgrupp

Leverabler etapp 3

1. Lyckad installation internt
2. Lyckad installation externt
3. Etablerad utkast tjänsteavtal (innan tjänsteetablering beslutas)

Designval

Maskinlayout

Tanken är att bygga lastbalanseringslösning för att kunna skala ut till många cache-maskiner vid behov. Vi kommer använda BGP med ECMP-routing för att dela ut trafik till flera maskiner. Ett klassiskt problem i samband med detta är att ECMP-routingen kastas om helt när man plockar i eller ur maskiner för maintenance. Detta är ett mindre problem vid enkelt frågasvar protokoll som DNS, men i fallet HTTP så kan det vara tråkigt att skicka RST-paket till alla klienter oavsett om de råkade prata med den specifika noden man tog ur drift eller inte.

För att lösa detta vill vi bygga en multi-tier lösning, vilket betyder att det finns ett främre lager maskiner som terminerar BGP, men som sedan har som uppgift att se till att oavsett vem av dem som får ett IP-paket adresserat till en viss adress, så ska alla skicka denna vidare till samma bakomliggande server. I fallet att man plockar ur en sådan maskin ur BPG så ska det alltså inte spela roll att all trafik kastas om, för kvarvarande maskiner kommer se till att det är samma bakomliggande server som tar emot dem. En illustrativ beskrivning av detta koncept finns här: https://vincent.bernat.ch/en/blog/2018-multi-tier-loadbalancer

Tanken är ju sen att vi ska skydda origin-servern från att överlastas, så detta betyder att vi behöver hålla en cache av webdata lokalt.

Detta medför att det kommer finnas två servertyper i projeketet, de främre som terminerar BGP och ser till att skicka vidare paketen till samma bakomliggande cacheserver. Dessa kallas för "**l4lb**". Sedan kommer det finnas de bakomliggande servrarna som faktiskt håller cache av webdata, dessa kallas helt enkelt för **cache**.

Protokoll

Vi kommer stödja HTTP över port 80, och HTTPS över 443. Tanken är att båda protokoll förmedlas bakåt till origin-server, och att vi sen cache:ar resultatet. Det är ännu inte planerat att stödja QUIC.

Certifikatshantering

Första tanken var att använda Caddy och dess inbyggda clusterstöd för att sköta certinhämtning från Lets Encrypt och sedan förmedla dessa mellan sig. Detta fungerar bra så länge man håller sig på en site, men eftersom detta projekt förväntas vara multi-site så uppstår problem med storagelösningen när vi vill ha oberoende siter. Den enklaste lösningen skulle vara att ha flera siter som oberoende av varandra hämtar cert för samma namn från Lets Encrypt, men detta springer in i begränsningar som Duplicate Certificate Limit.

För att ha en lösning som minimerar risken att springa in i sådant är rimligtvis rätt väg att gå att det bara finns en aktiv certförnyare, och att certen sen får spridas från denna till webserver-noderna. Skulle en site försvinna kommer det förmodligen vara ett manuellt handgrepp att ändra vilken site som står för inhämtandet av cert. Med övervakning ska vi kunna uppmärksamma problem i god tid innan certen går ut.

Vidare hade det varit trevligt att inte behöva lagra kunders privata TLS-certifikat på maskinerna alls, men just nu ser det ut som detta är ett krav för att kunna ha prestanda i systemet. Något som verkar lovande på den här fronten är Delegated Credential men det verkar vara på ett experimentellt stadie just nu.

XDP för att hantera stor mängd trafik

För att vara motståndskraftig mot volymbaserad DDOS vill vi använda XDP. Helst vill vi använda detta både för "bra" trafik som ska vidare till cache, men också för blockering av trafik som är elak. För att använda XDP för inledande lastbalansering mellan cache-noder så tittar vi på att använda oss av Cilium. Detta bör ge oss bra lastbalanseringshastighet på l4lb-lagret och åtminstone grundläggnade stöd för att blockera IP-addresser som upplevs aggressiva.

Ett problem som finns är att det bara går att köra ett eBPF-program i XDP per nätverkskort (det finns saker som https://github.com/xdp-project/xdp-tools/tree/master/xdp-loader som simulerar att man kan ladda flera separata program, men det bygger då på att man har ett övergripande program som löser detta, och detta kräver att man använt libxdp, vilket inte är kompatibelt med Cilium, se https://github.com/cilium/ebpf/discussions/940). Det innebär att vi inte enkelt kan kombinera Ciliums paketforwardering med andra defensiva eBPF program, exempelvis för att effektivt kunna svara på en ICMP ping flood eller hantera SYN-flood med SYN-cookies i XDP samtidigt.

Cache

Hjärtat i DDOS-skyddet är att vi cache:ar HTTP-data så klienter inte ens når origin-servern, tanken är att detta görs med Varnish. Den ger via sitt VCL-språk mycket stor kontroll över vad som ska cache:as och hur. En utmaning i det här projektet är att man vanligtvis cache:ar en HTTP GET, men inte inte en POST, vilket kanske kan vara en väg att sänka origin-servern den vägen om man vet att det går att kliva rakt genom skyddet genom att skicka massa POST-förfrågningar istället. I den bästa av världar så är websidan man skyddar bara ett ställe att läsa från, inte att skriva till, men med tanke på att sådant kan vara olika från system till system så borde varnish vara ett sätt att skräddarsy exakt hur cachen ska bete sig för respektive kund.

Varnish finns förutom sin opensource-version även i en enterprise version. Denna har massa utökad funktion, exempelvis stöd för att hantera TLS (HTTPS) både från klienter och vid backend-förfrågningar till origin-servern samt olika moduler för att göra intressanta saker. Vi kommer inledningsvis försöka se hur långt det går att komma med opensource-versionen, men det kommer kräva mer arbete.

Det största problemet med opensource-versionen av varnish är just att den inte stödjer inkommande HTTPS, eller utgående HTTPS. För att lösa det är tanken att använda HAproxy både som TLS-terminerare för inkommande trafik, samt som "TLS onloader" för trafik riktat till origin-server. Det finns fungerande exempel på hur man använder HAproxy som onloader här: https://varnish-cache.org/docs/trunk/users-guide/vcl-backends.html#connecting-through-a-proxy

Cache-invalidering

Sekunden efter man aktiverar en cache kommer någon undra varför en ändring man gjort inte syns på hemsidan. Vi behöver därför stödja invalidering av cache på beställning. Antingen genom att man skickar en HTTP PURGE request till den URL som ska invalideras, eller, om applikationen på origin-servern är mer cache-aware, kan tagga sidor med headers och sedan använda exempelvis varnish modulen xkey för att kasta alla relaterade sidor. En sådan request behöver spridas över alla körande cache-noder. Tanken är att sköta detta purge-spridande via MQTT-meddelanden som kan spridas både inom ett DC och mellan DC. Förmodligen kommer vi använda en MQTT-broker per datacenter, och sen bygga bryggor så varje MQTT-broker ansvarar för att pusha sina dc-specifika topics till alla andra brokers i andra datacenter, det leder i så fall till en full mesh koppling mellan alla brokers. Work-in-progress kod för MQTT-prataren finns här: https://github.com/SUNET/sunet-cdnp.

Inkoppling i nätverket

Vi vill ha fristående fysiska interface för internet-exponerade portar som vi serverar tjänsten på, både managementtrafik (SSH) och förmodligen även routeannonsering (BGP) bör ske på separat interface för att inte störas av överbelastningsattacker. Vi bör även se till att ha så mycket filter som möjligt i framförvarande router så bara sådant som behövs för tjänsten kan ta sig fram till l4lb-maskinerna. Allt som kan filtreras innan det når oss är en vinst. Vi vill göra silent drop så tidigt som möjligt på alla annan trafik.

Offernod

Det finns en tanke om att adresser förutom att annonseras på olika platser i sverige också ska annonseras på något enskilt ställe utanför landet. Tanken med detta är att omfattande DDOS-attacker som kommer från utlandet ska attraheras till den nod som finns utanför landet, och därför inte störa ut inhemska frågor.

Kundlayout

Vi vill förmodligen köra på modellen att varje kund passerar gemensam l4lb men har egen IP-address, egen haproxy och egen varnish. Det gör det lättare att hålla nere konfigurationsstorleken och en enskild kund med problem bör vara lättare att lokalisera och hantera när man har direkt påverkan på övriga kunder.

Loggar

Det finns en tanke om att vi ska kunna skicka loggar uppdelade per kund från systemet, det kan försvåra felsökning en hel del om man har en vägg framför sin origin-maskin som man inte har insikt i. Det hade varit väldigt praktiskt om kunder kan ta del av vad cachen faktiskt serverar till folk, inte minst för att hålla koll på vilken material som efterfrågas osv.