Poseidon: Mitigating Volumetric DDoS Attacks with Programmable Switches

Feb 2020

Network and Distributed System Security Symposium (NDSS) ( rank 5)

65 citations

---

# پوسایدن: راهکاری برای رفع تهدید حملات منع‌خدمت توزیع شده حجیم با استفاده از سوییچ‌های برنامه‌پذیر

حملات منع‌خدمت با ظهور دستگاه‌های اینترنت اشیا نمود بیشتری پیدا کرده‌اند. یک دسته از راه‌های رفع تهدید این حملات، استفاده از مراکز شستشوی ترافیک[1] هستند که در آن‌ها مکانیزم‌های دفاعی در نزدیکی مقصد پیاده می‌شوند و از دستگاه‌های میانی— سخت‌افزاری[2]— استفاده می‌شود که هزینه زیادی دارند و علاوه‌براین برای تعریف مکانیزم‌های جدید دفاعی نیز نیاز به ارتقا خود دستگاه سخت‌افزاری دارند که هزینه و مشکلات زیادی دارد. دسته دیگر روشهای مبتنی بر نرم‌افزار هستند. بااینکه این‌روشها انعطاف‌پذیری بالایی دارند اما مشکل تاخیر و سربار زیادی دارند (به دلیل این است که از پردازنده‌های عمومی برای پردازش بسته‌ها به جای سخت افزار اختصاصی استفاده می‌کنند). مقاله فعلی قصد دارد با استفاده از سوییچ‌های برنامه‌پذیر، که به‌تازگی در حال گسترش هستند، محدودیت‌های ذکرشده را بدون نیاز به سخت‌افزار اضافی رفع نماید. با استفاده از پوسایدن، کاربران می‌توانند مکانیزم‌های دفاعی‌شان را در قالب یک مجموعه از اصول دفاعی بیان کنند و در آینده برای حملات جدیدتر بهبود و گسترش دهند. پوسایدن این اصول دفاعی را به سوییچ‌های برنامه‌پذیر بر اساس یک روش بهینه نگاشت و متقبل می‌کند و هر جا که لازم بود نیز می‌تواند برخی از آنها را بر روی نرم‌افزار سرور پیاده سازی و اجرا کند(به‌دلیل اینکه سخت‌افزار این دستگاه‌ها محدود است) . درآخر طبق بررسی‌هایی که انجام می‌شود قابلیت دفاع در برابر حملات حجیم و سازگارپذیری بدون سرباراضافی نیز تایید می‌شود.

سوییچ‌های ASIC سرعت پردازش بسیار بالایی دارند.سوییچ‌های برنامه‌پذیر نرخ‌گذری بسیار بیشتراز بهترین نرم‌افزارهای پردازش بسته دارند و هر کدامشان کارایی برابر چندین سرور دارند که در نتیجه هزینه نهایی را نیز پایین خواهند آورد. همچنین میزان مصرف منابع به نسبت توجیه‌پذیرتری دارند. اما مشکل آنها محدودیت حافظه و منابع است. همچنین این سوییچ‌ها از پردازش بسته statefull به کمک زبان‌های خاص‌منظوره مانند p4 بهره می‌برند، که می‌توانند در خطلوله سوییچ‌ها بسته‌هارا طبق منطقی که کاربر تعریف کرده، با بالاترین سرعت ممکن

---

[1] Scrubbing centers

[2] middleboxes

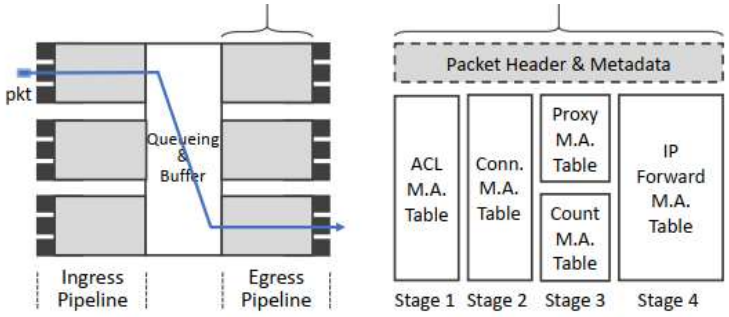
پردازش کنند. به کمک آن می‌توان خط‌مشی‌هایی شامل چندین خط کد نوشت و تمام حملات را پوشش داد. به سرعت خوبی با تغییر خط‌مشی ها سازگاری می‌یابد و تاثیر منفی بر روی ترافیکهای معمولی و مجاز نمی‌گذارد.

روش کار سیستمهای تشخیص حملات منع‌خدمت توزیع‌شده به این صورت هست: در ابتدا تشخیص می‌دهد که آیا یک کاربر در زیر حملات هست. سپس ترافیک را به مسیرهای از پیش تعیین‌شده هدایت می‌کنند، در صورت نیاز خط مشی‌ها را اصلاح می‌کنند، و درآخر اقدام به رفع[3] مخاطره می‌کنند.

روش کار پوسایدن به این صورت هست که یکسری از خط‌مشی‌های دفاعی را از کاربر می‌گیرد و آنها را به منابع موجود (سوییچ‌ها و سرورها) متقبل می‌کند (کاربر نیاز به اینکه این منابع چه هستند ندارد) . پوسایدن به جای این‌که از زبان‌های سطح پایین مثل p4 یا c++ برای تعریف خط‌مشی‌های دفاعی استفاده کند، مجموعه‌ای از اصول دفاعی را به صورت ماژولار پشتیبانی می‌کند که می‌توان آنها را برای ساخت خط‌مشی‌های گوناگون به کار برد و کاربر خودش نیز می‌تواند این مجموعه از اصول را نیز ارتقا دهد.

مدل حمله: فرض می‌شود مهاجم امکان اجرای چندین نوع حمله از بین حملات ممکن را به صورت همزمان داشته‌باشد.

**سوییچ‌های برنامه‌پذیر**: به کمک SDN می‌توان از طریق برنامه‌ریزی سوییچ‌های برنامه‌پذیر ASIC و زبانهای مخصوص شان برای سطح دیتا[4] هوشمندی آورد. هر سوییچ برنامه‌پذیر ASIC از چندین خط[5] ورودی و خروجی تشکیل شده هست و هر خط ورودی شامل چندین مرحله[6] هست که بسته‌ها به ترتیب در آنها پردازش می‌شوند وهر مرحله منابع مخصوص خود را داراست، مثلاً : رجیسترها، جداول تطبیق-عمل[7] که هربسته که فیلدهایش مطابقت داشت، عمل متناظر را بر رویش انجام می‌دهد و همچین Stateful ALUها




(a) Switch data plane architecture (b) Ingress/Egress pipeline architecture

| Match | Action | Action Data |
|---|---|---|
| src_ip:10.0.1.0/24 | set_next_hop() | eport=4 |
| src_ip:192.0.0.0/8, tcp_flag:0x10 | validate_cookie() | cookie_seed=531 |
| ... | ... | ... |
| default | drop() | - |

(c) Match-action table architecture

[3] mitigation
[4] Data Plane
[5] Pipeline
[6] stage
[7] Match-Action tables (M.A)

برای پردازش و انجام و عملیات. هر جدول اطلاعات جدیدی که از بسته ها به دست می‌آورد را با دیگران به اشتراک می‌گذارد. با استفاده از زبان‌هایی مثل p4 می توان هدر بسته ها و محتویات جداول را تغییر داد. کامپایلرهای مخصوص این برنامه‌ها را به باینری تبدیل کرده که در سوییچ ها لود شوند. و همچنین رابط برنامه نویس برای ارتباط با control plane هست.

**زبان تعریف خط مشی پوسایدن**: با این‌که ممکن هست حملات منع خدمت مختلف از ضعف‌های گوناگون سیستم و پروتوکل‌ها استفاده کنند، اما می‌توان همه حملات را بایک سری بررسی‌ها روی هدرهای بسته‌ها و ویژگی‌های آماری آنها تشخیص داد. پس با استفاده از یک زبان با سطح انتزاعی بالا می‌توان خط مشی‌های مناسب را تعریف کرد. به صورت کلی یک خط مشی شامل یکسری از عبارات تطابق/عمل هست که بر روی یک مجموعه از سرایند بسته‌ها تعریف می‌شود.

**Expression**
$E$ ::= v | h | $M(\vec{v})$ | $E \diamond E$
**Predicate**
$P$ ::= $E \circ E$ | $P\&P$ | $P|P$ | $\neg P$
**Monitor**
$M$ ::= count($P, \vec{h}$, every) | aggr($P, \vec{h}$, every)
**Action**
$A$ ::= drop | pass | log | rlimit | sproxy | puzzle
**Policy**
$C$ ::= $A$ | if $P$: $C$ else : $C$ | $(C|C)$

Fig. 3: The syntax for expressing POSEIDON defense policies. $\diamond$ describes calculative operators and $\circ$ describes logical operators.

در زیر اصول دفاعی را کمی بررسی می‌کنیم:

- مانیتورها: جمع‌آوری اطلاعات آماری بسته‌ها. می‌تواند به طور کامل روی سوییچ‌ها پیاده‌سازی شود. در اصل با استفاده از انگاره‌ها[8] آن‌ها را پیاده می‌کند. انگاره‌ها داده‌ساختارهایی بهنیه در مصرف منابع هستند که اطلاعات آماری را با تقریب خوبی نگه داری می‌کنند.
- اعمال: به سه دسته تقسیم می‌شوند:
  - آنهایی که به تنهایی روی سوییچ می‌توانند پیاده شوند
  - نیاز به کمک گرفتن از سرور دارند
  - به طور کامل روی سرور باید اجرا شوند چون سوییچ‌ها توانایی پردازش آنها را ندارند ( مثل puzzle )

[8] sketch

- خط‌مشی‌ها: گاهی لازم است اعمال متفاوتی را برای ترافیک انجام داد. بدین منظور می‌توان از عبارات شرطی استفاده کرد. عملگر ترکیب بدین منظور هست که اجازه می‌دهد که اعمال متفاوتی روی یک ترافیک انجام شود و فرض می‌شود که عمل سخت‌گیرانه‌تر ارجحیت داشته باشد
- قرابت( به هم پیوستگی )جریان‌ها: برای برخی اعمال لازم هست.

TABLE II: Implementation details and resource utilization of POSEIDON primitives.

| Primitives | Switch Component | Switch Resource Usage | Server Component |
|---|---|---|---|
| **monitors** | | | |
| count($P, \vec{h}$, every) | match-action entry + count-min sketch | stages: 2, hash functions: $\lceil \log_{1/2} \delta \rceil$, stateful ALUs: 6, SRAM: for the $\phi$ biggest elements in a set, in order to achieve a relative error bound of $\varepsilon$ with probability $\delta$, usage = $\frac{64\lceil \log_{1/2} \delta \rceil}{\varepsilon\phi}$ | N/A |
| aggr($P, \vec{h}$, every) | match-action entry + count-min sketch | stages: 2, hash functions: $\lceil \log_{1/2} \delta \rceil$, stateful ALUs: 6, SRAM: for the $\phi$ biggest elements in a set, in order to achieve a relative error bound of $\varepsilon$ with probability $\delta$, usage = $\frac{64\lceil \log_{1/2} \delta \rceil}{\varepsilon\phi}$ | N/A |
| **actions** | | | |
| drop | flow entry | stages: 1, hash functions: 0, stateful ALUs: 0, SRAM: negligible | N/A |
| pass | flow entry | stages: 1, hash functions: 0, stateful ALUs: 0, SRAM: negligible | N/A |
| rlimit | meter + flow entry | stages: 3, hash functions: 1, stateful ALUs: 0, SRAM: in order to achieve a false positive rate of $\varepsilon$, usage = $\frac{8n}{\ln(1/(1-\varepsilon))}$ | N/A |
| sproxy | handshake proxy + session relay | stages: 3, hash functions: 2, stateful ALUs: 4, SRAM: in order to achieve a false positive rate of $\varepsilon$, usage = $\frac{32n}{\ln(1/(1-\varepsilon))}$ | N/A |
| puzzle | - | - | CAPTCHA |
| log | selecting, grouping | stages: 3, hash functions: 2, stateful ALUs: 2, SRAM: in order to achieve a false positive rate of $\varepsilon$, usage = $\frac{32n}{\ln(1/(1-\varepsilon))}$ | aggregation |
| **branches** | | | |
| if . . . else . . . | tag-based match action | stages: 1, hash functions: 0, stateful ALUs: 0, SRAM: negligible | N/A |

**تعیین مکان اصول دفاعی**: از خط‌مشی تعریف‌شده یک گراف می‌سازد که گره‌های آن اصول دفاعی و یالهای جهت‌دار نشان‌دهنده مسیر عبوری ترافیک هستند. و سپس می‌آید از روی این گراف یک لیست مرتب‌شده می‌سازد. سپس بر اساس این‌که هر اصل چه تعداد منابع می‌خواهد نود ها را جایگذاری می‌کند. با این هدف که تا آنجاکه میتواند این کار را به ASIC ها بسپارد. و با توجه به محدودیت منابع آنها می‌تواند از منابع چندین سوییچ استفاده کند. سپس مسئله تعیین مکان را به مثابه یک ILP مدل می‌کند وآن را حل می‌کند و در نهایت به عنوان نتیجه بیان می‌کند که کدام یک از اصول باید در سوییچ اجرا شوند و چه مقدار منابع نیز لازم هست.

**مدیریت حملات پویا در لحظه اجرا**: پوسایدن به این صورت عمل می‌کند که هرگاه لازم شد که برنامه یک سوییچ را دوباره تنظیم کند، به صورت موقت اطلاعات آن در یک سرور مرتبط ذخیره می‌کند و اگر ترافیکی آمد به سوی آن سرور هدایت می‌شود. بدین‌منظور پوسایدن از یک کنترل‌کننده مرکزی برای هماهنگی میان سرورها و سوییچ‌ها استفاده می‌کند. سرورها همه اصول‌دفاعی را از قبل دارند، اما آنهایی که به تنهایی بر روی سوییچ ها می‌توانند اجرا شوند غیرفعال هستند، مگر این که وضعیت ذکر شده رخ دهد. اما انجام این کار در لحظه چالش‌هایی را به همراه دارد که پوسایدن برای رفع آنها از این روش استفاده می‌کند:هنگامی که اپراتورها یک خط‌مشی را مشخص می‌کنند، پوسایدن حالتی را که نیاز به کپی‌کردن دارند شناسایی می‌کند. در زمان اجرا، اگر حالت‌ها

ایجاد/به‌روزرسانی/حذف شوند، پوسایدن بسته‌های کپی حالت تولید می‌کند و ترافیکی شامل این بسته‌ها را به مجموعه‌ای از سرورها هدایت می‌کند (یعنی آن حالات را میخواهد بین این سرورها به اشتراک بگذارد) .همچنین تناظر بین آدرس آی‌پی سرور و کلید (شامل اطلاعات سرایند بسته‌ها) موجود در کنترلر را ثبت می‌کند. هنگامی که اپراتورها خط‌مشی دفاعی را برای مدیریت حملات جدید تغییر می‌دهند، پوسایدن مسیریابی بالادست[9] را مطابق تناظر موجود به روز می‌کند تا اطمینان حاصل شود که ترافیک قانونی به سرورهای صحیح هدایت می‌شود. همچنین برنامه‌های جدید P4 را دوباره کامپایل و بارگذاری می‌کند. نکته‌ای که باید توجه‌کرد اینست که با شروع پوسایدن، کل رویه به طور خودکار اجرا می‌شود و اپراتورها (کاربران) فقط در صورت وجود حملات پویا نیاز به ایجاد تغییرات در خط‌مشی‌های سطح بالا دارند.

**پیاده سازی** :

[9] upstream