Rozlišení člověk/robot na úrovni HTTP použitelné pro omezení DDOS útoků

Marek Aufart, aufarmar@fel.cvut.cz

29.8.2011

Úvod

Chtěl bych vytvořit nástroj pro rozlišování HTTP požadavků podle jejich chování. Rozlišování se bude zaměřovat na normální uživatel/robot, případně na skupiny požadavků, které budou mít podobné chování.
Chováním požadavků mám na mysli jejich časovou náročnost pro server, HTTP kód, kterým jim server odpoví, jejich počet a rozložení. Větší částí této práce bude tyto parametry vybrat a napsat co nejlepší algoritmus pro jejich vyhodnocování.

Řešit budu pouze úroveň protokolu HTTP, tedy validní HTTP požadavky.

Vývoj

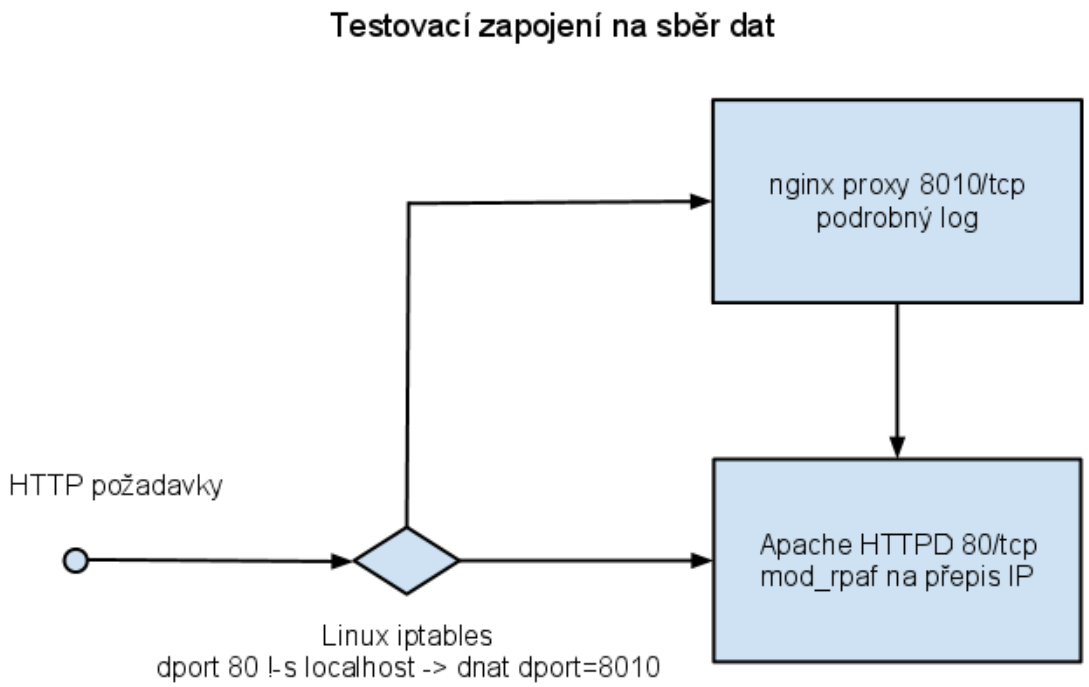
Pro tuto práci potřebuji nějaká data, která budu zpracovávat. Algoritmus jsem se pro začátek rozhodl napsat v jazyce Ruby, protože je to rychlejší a lépe se to mění.

Výsledný program (modul do webového serveru) bude s nejvyšší pravděpodobností napsán v jazyce C až po vyzkoušení funkčnosti ve vývojovém prostředí.

Získání dat

Pro sběr dat jsem potřeboval připravit zvláštní proxy server, protože Apache HTTPD zatím neumí logovat čas zpracování požadavků s větší přesností, než sekundy.

Konfigurace je na obrázku níže. IPtables překládají cílový port na 8010, kde poslouchá reverzní proxy. Pro zachování správné IP adresy klienta v logu Apache bylo třeba použít modul rpaf, který přesouvá ip adresu klienta z hlavičky X-forwarded-for, pokud je použita.



Logy z reverzní proxy měly po nastavení formát cvs a obsahují následující informace:
"$time_local","$remote_addr","$status","$request_length","$body_bytes_sent","$request_time","$connection","$pipe","$host","$request","$http_referer","$http_user_agent"

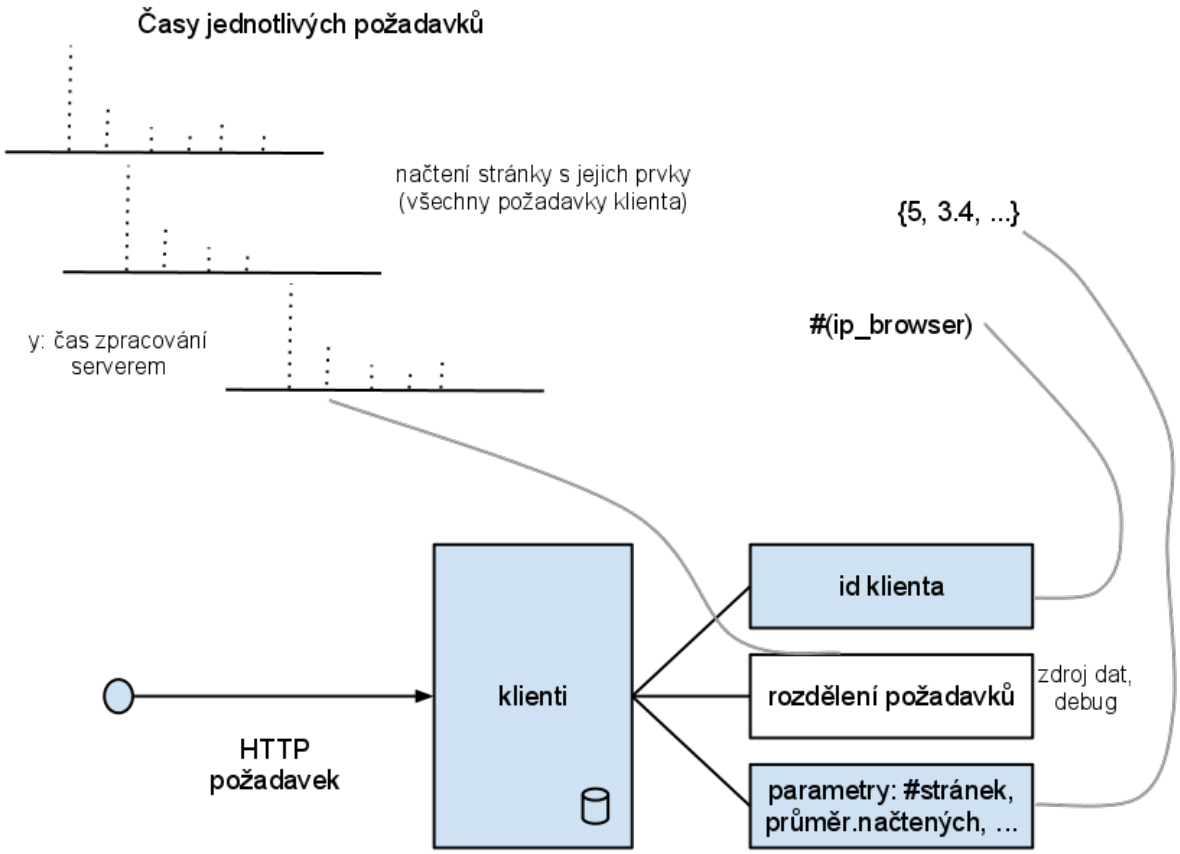
Například tedy:
"02/Apr/2011:17:30:41+0200","217.196.113.136","200","864","3182","0.570","1",".","dev.aurem.cz","GET /uvod HTTP/1.1","http://dev.aurem.cz/user","Mozilla/5.0 (X11; U; Linux i686; en-US) AppleWebKit/534.16 (KHTML, like Gecko) Chrome/10.0.648.127 Safari/534.16"

Algoritmus

Na vyzkoušení algoritmu jsem si napsal skript v Ruby, který čte log a postupně posílá požadavky. Vývojová verze je na https://gist.github.com/1178950.
V paměti se drží hash tabulka s klienty a parametry jejich požadavků a podle těchto parametrů zkoumám, které jsou pro jejich rozlišení výrazné a které ne. Parametry jsou blíž popsány v odstavci Parametry.

Př.: `"217.196.113.136_Mozilla/5.0(X11;U;Linuxi686;en-US)AppleWebKit/53... @request_count=17, @notmod_count=0, @avg_time=1.373, @avg_time_count=17>`

Klient má být identifikován IP adresou a hlavičkou jeho prohlížeče. Tím chci omezit zablokování všech klientů, kteří jsou za jedním NATem. Počítám s tím, že hlavičku může útočník podvrhnout, takže v algoritmu nebude její důležitost velká, aby se v některých případech IP adresa zablokovat dala.



Nejlepší prostředí, nad kterým bych takovou funkčnost postavil, se mi jeví lehký webový server nginx. Narozdíl od Apache HTTPD nevytváří pro každé spojení vlastní vlákno a tak je rychlejší. Navíc se používá i jako reverzní proxy nebo na poskytování statického obsahu.

Modul webového/proxy serveru by se skládal ze dvou částí - detekční a filtrovací. Detekční by byla prováděna po vyřízení požadavku (protože algroritmus potřebuje i informace, které nejsou známé z požadavku klienta, například doba, kterou vyřízení trvalo, stavový kód HTTP). Infomace o požadavku by byly zapisovány do hash tabulek (podle IP a hlaviček).

Filtrovací část by naopak byla prováděna hned při přijetí požadavku a zajišťovala by jeho zahození, pokud by klient byl označen jako ”špatný”.
Pokud bychom nechtěli filtrovat i podle některých hlaviček a stačilo by filtrování podle IP adresy, můžeme použít některé existující způsoby (Deny from IP, firewall). Který způsob bude lepší se ukáže až při testování a ladění algoritmu.

Parametry

Výběr parametrů, podle kterých se požadavky budou filtrovat, je asi nejdůležitejší část této/budoucí práce.
Kromě počtu požadavků se mi nejzajímavějším parametrem zdá čas, který serveru trvalo

požadavek obsloužit. Zejména poměr mezi “výpočetní” a “paměťovou” prací. Rozdíl času zpracování dynamického a statického obsahu - víc čtení z disku (obrázky) x víc výpočtů (php, j2ee, ruby, .net) by měl pomoci “značkovat” požadavky a pomoci s rozlišením.

Další zajímavé parametry mohou být

- použití refereru aktuálního webu - css, js, obrázky (?)
- počet dotazů
- počet dotazů za minutu
- počet poslání 304 “not changed” (+poměr k výše uvedenému)
- min, max a medián nebo jiné funkce doby obsluhy požadavku

Obecně nelze spoléhat na informace, které může klient sám ovlivnit, tedy hlavičky (referer, ..). Otázkou je také manipulace a následná kontrola e-tagu a času poslední změny dokumentu odeslaného klientovi.

Existující řešení

Pro Apache HTTPD existuje mod_evasive, který má bránit (D)DOS útokům. Ten je založen čistě na počítání požadavků za časový interval z nějaké IP adresy, kterou je schopen zablokovat.
Můj projekt nebude primárně založen na počtu požadavků z jedné IP adresy, ale na jejich náročnosti (časové), rozložení a dalších parametrech, které se budou jevit jako použitelné.

Komerční/uzavřená řešení (IDS, F5 BIG-IP) mají podobnou funkčnost pouze jako okrajovou nebo se jedná o službu, např. blacklotus.net.

Zdroje

- Detecting Attacks on Web Applications from Log Files, SANS Institute
- A Detection Model Based on Statistical against DDoS Attack, Yoon-Ju Kook, Yong-Ho Kim, Jeom-Goo Kim, Kiu-Nam Kim
- Preventing Denial of Service Attacks on the Web: A Progress Report, M.Copenhafer, P.Mukhopadhyay, F.Schneider
- http://wiki.nginx.org/HttpLogModule
- http://www.zdziarski.com/blog/?page_id=442 mod_evasive