Upotreba metode segmentnog hešovanja iniciranog sadržajem za identifikovanje homogenih fajlova

Nenad Ristić
Fakultet za računarstvo i informatiku
Univerzitet Sinergija
Bijeljina, Bosna i Hercegovina
nristic@sinergija.edu.ba

Aleksandar Jevremović, Mladen Veinović
Univerzitet Singidunum
Beograd, Srbija
ajevremovic@singidunum.ac.rs
mveinovic@singidunum.ac.rs

Sadržaj - **Homogeni fajlovi dele identičan niz bitova, raspoređenih u istom redosledu. Pošto ovi fajlovi nisu potpuno identični, tradicionalne tehnike kao što je hešovanje ne mogu se koristiti za njihovo pronalaženje. U ovom radu primenjuje se nova tehnika za građenja heš potpisa koja koristi kombinovanje nekoliko tradicionalnih heš vrednosti čije su granice određene ulaznim sadržajem. Originalno, razmatrana tehnika se koristi za identifikovanje neželjene pošte, a u ovom radu se po prikazuje upotreba u forenzici. Ovi potpisi mogu se koristiti za identifikovanje izmenjenih varijanti poznatih fajlova, čak iako je određeni podatak dodat, izmenjen ili uklonjen iz nove verzije fajla. Opis ove metode praćen je kratkom analizom njenog dejstva, kao i primerom aplikacije za računarsku forenziku.**

Ključne riječi- Segmentno hešovanje, forenzika, spamsum, ssdeep

I. UVOD

Ovaj rad opisuje metodu za upotrebu pomičnog hešovanja na osnovu sadržaja u kombinaciji sa klasičnim heš algoritmom zbog pronalaženja poznatih fajlova u koje je ubačen, promenjen ili uklonjen neki deo fajla. Na početku se analizira klasično hešovanje fajlova za pronalaženje poznatih fajlova kao i slabosti koje ta metoda ima u forenzičkoj istrazi. U nastavku se predstavlja koncept segmentnog hešovanja. Na kraju je opisan algoritam pomičnog hešovanja (engl. *rolling hash*), pomoću kog se dobija pseudo-slučajni niz na osnovu ulaznog sadržaja. Upotrebom pomičnog hešovanja za određivanje granica klasičnog segmentnog hešovanja (engl. *piecewise hash*), dobija se pomično hešovanje određeno okidačima u sadržaju. Ovakvi heš nizovi mogu se koristiti za identifikovanje homogenih nizova između nepoznatih ulaznih fajlova i poznatog fajla, čak iako je nepoznati fajl izmenjena varijanta poznatog fajla. U radu će biti predstavljen *spamsum* algoritam, implementacija pomičnog hešovanja određenog okidačima u sadržaju i analiza performansi upotrebom programa pod nazivom ssdeep koji dokazuje koncept.

Algoritam objašnjen u radu preuzet je iz detektora neželjene pošte pod nazivom *spamsum*, a napisan je od strane dr. Endru Tridgel [1]. Algoritam *spamsum* ima mogućnost prepoznavanja neželjene pošte koja je slična ali ne i identična poznatim uzorcima. Zasnovan na *rsync* proveri koji je takođe kreiran od strane dr Tridgela 1999. godine.

II. SEGMENTNO HEŠOVANJE INICIRANO SADRŽAJEM

Originalno kreirano od strane Nikolas Harbura za dcfldd aplikaciju [2], segmentno hešovanje koristi algoritam za proizvoljno hešovanje da bi kreirao više kontrolnih vrednosti. Umjesto kreiranja jedne heš vrednosti za jedan fajl, heš se generiše za više segmenata određene veličine. Na primer, jedan heš se generiše za prvih 512 bitova ulaza, sledeći za narednih 512 bitova i tako dalje. Ova tehnika je razvijena da bi se ublažile greške tokom forenzičkih snimanja podataka. Ako se dogodi greška tokom snimanja, samo jedan segmentni heš neće biti ispravan. Ostatak segmentnih heš vrednosti kao i integritet podataka biće očuvani.

Segmentno hešovanje može da koristi bilo koji od šifrarskih algoritama, kao što je MD5 u dcfldd [2] ili neki od drugih klasičnih algoritama kao što je Flower/Nool/Vo [3] (*FNV*) heš. Ne vezano za algoritam, za potrebe ovog rada segmentno hešovanje se naziva klasičnim hešom da bi se napravila razlika od pomičnog hešovanja koje je objašnjeno u nastavku rada.

A. Pomični heš

Pomični heš algoritam proizvodi pseudo-slučajnu vrednost zasnovanu na kontekstu trenutnog ulaza. Pomični heš radi na osnovu održavanja prethodnog stanja i na osnovu stanja samo nekoliko poslednjih bita ulaza. Svaki bit je dodat stanju, obrađen i zatim uklonjen nakon što se određen broj drugih bita obrađuje.

Pretpostavimo da imamo ulaz od n karaktera gde je n-ti bit označen sa b_n. Dakle, ulaz se znači sastoji od bita $b_1, b_2, ..., b_n$. Na bilo kojoj poziciji p na ulazu, stanje pomičnog heša će zavisiti samo od poslednjih s bitova u fajlu [4]. Na osnovu ovoga, vrednost pomičnog heša može biti predstavljena kao funkcija poslednjih nekoliko bita kao što je prikazano u jednačini (1)

$$r_p = F(b_p, b_{p-1}, \ldots, b_{p-s}) \quad (1)$$

Funkcija pomičnog heša F je koncipirana tako da je moguće ukloniti uticaj jednog od uslova. Tako, za dato r_p je moguće izračunati r_{p+1} uklanjanjem uticaja b_{p+1}, predstavljeno kao funkcija $X(b_{p-s})$ i dodavanjem uticaja b_{p+1} predstavljeno funkcijom $Y(b_{p+1})$, kao što se vidi na u jednačinama (2) i (3)

$$r_{p+1} = r_p - X\left(b_{p-s}\right) + Y(b_{p-1}) \quad (2)$$

$$r_{p+1} = F(b_{p+1}, b_p, b_{p-1}, \ldots, b_{(p-s)+1}) \quad (3)$$

B. Kombinovanje heš algoritama

Dok programi koji koriste segmentno hešovanje, kao što je dcfldd koriste fiksne dužine za određivanje početka i kraja klasičnog heša, algoritam segmentnog hešovanja iniciranog sadržajem (*SHIS*) koristi pomični heš. Kada izlaz pomičnog heša dâ specifičan izlaz ili vrednost koja je okidač klasični heš se pokreće. To znači, da dok se obrađuje ulazni fajl, paralelno se vrši računanje klasičnog heša za fajl. Istovremeno, mora se proračunati pomični heš fajla. Kada pomični heš proizvede vrednost koja je okidač, vrednost klasičnog heša snima se u SHIS potpis i klasični heš se resetuje.

Na kraju, svaka od snimljenih vrednosti u SHIS potpisu odgovara delu ulaza, i promene u ulazu prouzrokovaće samo lokalizovane promene na SHIS potpisu. Ovo znači da u slučaju izmene jednog ili dva bajta, samo jedna od vrednosti klasičnog heša će biti promenjena, a samim time i najveći dio SHIS potpisa će ostati ne promenjen. Zbog toga što najveći deo potpisa ostaje isti, fajlovi sa promenama i dalje mogu da se povežu sa potpisima poznatih fajlova.

U nastavku ovog rada biće prikazana implementacija SHIS pod nazivom *spamsum* algoritam. Ovaj algoritam je implementiran u programu ssdeep koji je objavljen na Internet lokaciji proizvođača [5].

III. SPAMSUM ALGORITAM

Algoritam spamsum koristi FNV hešove za klasično hešovanje, koje obezbeđuje 32 bitni izlaz za bilo koji ulaz. U spamsum algoritmu smanjen je FNV heš snimanjem samo osnove šifrovanja poslednjih šest najmanje važnih bita (*P6NB*) svake heš vrednosti.

Algoritam za pomični heš zasnovan je na Alder32 proveri vrednosti i pseudokoda prikazanog u nastavku.

Originalni spamsum algoritam imao je dodatno logičko I (AND) sa 0xffffffff uz pomeranje u desno koje je izbrisano. Ova upotreba "logičkog I" prvobitno je ograničavala veličinu u algoritmu na 32 bitne vrednosti. Pošto je autor eksplicitno naznačio da su sve vrednosti u algoritmu neoznačene 32 bitne vrednosti, funkcija „logičkog I" nema efekta i zbog toga je izuzet.

Prije obrade ulaznog fajla, mora se izabrati okidač vrednost za pomični heš. U *spamsum* algoritmu, kao i u ovom radu, okidač vrednosti će biti označena veličina bloka. Dve konstante, minimalna veličina bloka, $b_{min,}$ i dužina spamsum, S, se koriste za iniciranje početne veličine bloka za ulaz n bita kao što je prikazano u jednačini (4). Veličina bloka se može promeniti nakon ispunjenja određenih uslova, koji će biti opisani u nastavku. Veličina bloka koja proračunava prije čitanja bloka naziva se inicijalna veličina bloka b_{int}.

$$b_{int} = b_{min} 2^{\log_2\left(\frac{n}{sb_{min}}\right)} \quad (4)$$

Nakon obrade svakog bajta, pomični heš se menja i rezultat se poredi sa veličinom bloka. Ako pomični heš kao rezultat daje vrednost koja je jednaka modulu veličine bloka u odnosu na veličinu bloka umanjenu za jedan, onda je pomična provera vrednosti dostigla graničnu vrednost. U ovom trenutku bazna šifrovana vrednost šest najmanje važnih bajtova klasičnog heša dodaje se prvom delu konačnog potpisa. Slično, kada pomična provera vrednosti dostigne vrednost koja je jednaka, dvostrukom modulu veličine bloka, bazna šifrovana vrednost šest najmanje važnih bajta (*6NVB*) klasičnog heša dodaje se drugom delu spamsum heša. Važno je da se održe dva razdvojena stanja klasičnog heša, po jedan blok sa svake strane jednačine [1].

Nakon obrade svakog bajta, konačni potpis se pregleda. Ako prvi deo potpisa nije dovoljno dug nakon što su svi ulazi obrađeni veličina bloka se prepolovi i ulaz se obrađuje ponovo.

Konačni spamsum potpis se sastoji od veličine bloka, dva paketa 6NVB, i naziva fajla unutar navodnika. Prvi paket 6NVB se računa sa veličinom bloka b i drugog $2b$.

IV. POREĐENJE SPAMSUM POTPISA

Dva spamsum potpisa se mogu porediti da bi se utvrdilo da li potiču od fajlova koji su homogeni. Provera prvo gleda veličinu bloka, eliminišu se sve sekvence, izračunava se razmak između blokova. Razmak između izmena se meri da bi se dobila ocena podudaranja, ili podudarnost poređanih homogenih nizova u oba fajla.

Okidači u klasičnim hešovima su bazirani na ulaznom fajlu i veličini bloka, pa se mogu porediti samo potpisi sa identičnim blokovima. Spamsum algoritam generiše potpise za svaki ulaz a kao osnovu koristi veličinu bloka b i $2b$, tako da je moguće porediti dva potpisa ako je dati blok u osnovi ima dva. Na primer, dva potpisa, prvi sa veličinom bloka b_x i drugi sa veličinom bloka b_y, prvi potpis ima SHIS vrednosti za blok veličine b_x i $2b_x$, drugi b_y i $2b_y$. Možemo uporediti ova dva potpisa ako je $b_x = b_y$, $2b_x = b_y$, ili $b_x = 2b_y$.

Kada se utvrdi veličina bloka, uklanjaju se nizovi koji se ponavljaju. Ovi nizovi ukazuju na šablone u ulaznom fajlu i uglavnom ne pokazuju mnogo informacija o sadržaju fajla. Na kraju, uz pomoć dinamičkog programiranja proračunava se razmak između dva heša. Uzmimo dva niza karaktera *s1* i *s2*, razmak izmene između ta dva niza se definiše kao minimalan broj mutacija potreban da se *s1* promeni u *s2*, gdje se mutacija odnosi na menjanje, umetanje, ili uklanjanje karaktera. *Spamsum* algoritam za merenje razmaka koristi izmenjenu verziju algoritma izvorno kreiranog za USENET čitač vesti. U ovoj verziji, svako ubacivanje ili brisanje meri se kao jedna razlika, ali svaka izmena se meri kao tri i svaka zamena (*npr. Isti karakteri ali u obrnutom rasporedu*) meri se kao pet. Ova izmerena dužina izmene se definiše kao $e(S_1,S_2)$ za potpise S_1 i

S_2 dužine l_1 i l_2 . Kao što se može videti u nastavku prikazano jednačinama (5), (6) i (7). U ovim jednačinama, *i* je broj ubacivanja, *d* je broj brisanja, *c* je broj izmena, i *w* predstavlja broj zamena.

$$e = i + d + 3c + 5w \tag{5}$$

$$c + w \leq \min(l_1, l_2) \tag{6}$$

$$i + d = |l_1 - l_2| \tag{7}$$

Razmak između izmena se preračunava sa od 0 do 64 na od 0 do 100 i primenjuje tako da nula predstavlja nehomogene fajlove dok 100 predstavlja rezultat za skoro identične fajlove. Konačni rezultat podudaranja, *M*, za nizove teksta dužine l_1 i l_2 mogu se izračunati upotrebom jednačine 8 prikazane u nastavku.

Rezultat poklapanja predstavlja izmereni procenat koliko S_1 i S_2 imaju poređanih homogenih sekvenci, tj. meri se koliko dva potpisa imaju identičnih bita u istom rasporedu. Što je veći rezultat poklapanja, veća je verovatnoća da potpisi potiču sa istog izvora što znači da je veća verovatnoća da izvorni fajlovi potiču sa istog izvora. Veći rezultat poklapanja ukazuje na veću mogućnost da izvorni fajlovi imaju zajedničke blokove vrednosti koji su i u istom rasporedu.

$$M = 100 - \left(\frac{100\,Se(S_1, S_2)}{64(l_1 + l_2)} \right) \tag{8}$$

Program ssdeep, koji je dokaz koncepta, identifikuje fajlove koji imaju rezultat poklapanja veći od nule kao podudarajuće. Tokom testiranja autor nije pronašao različite fajlove koji su imali rezultat poklapanja veći od nule, u daljem istraživanju trebalo bi razmotriti i istražiti ovo područje.

Moguće je da dva fajla imaju identične SHIS potpise a da budu različiti. Ako se na fajlu izvrši izmena koja ne utiče na okidače pomičnog heša dok kreira klasične heš, vrednost segmentnog može biti drugačija. Pošto se vrednost klasičnog heša umanjuje sa 32 bita na 6 bita, po principu golubinjaka (*engl. Pigeonhole Principle*), mnogo vrednosti klasičnog heša mogu imati istu vrednost u potpisu. Preciznije, postoji 2^{-6} verovatnoća da će novi blok imati isti SHIS potpis kao originalni fajl. Znači, čak i ako dva fajla imaju identične potpise, nema dokaza da su to fajlovi koji imaju identičan svaki bit i istražitelj mora dalje pregledati fajlove. Ova problematika se može prevazići upotrebom heš algoritma za šifrovanje kao što je MD5. Upotreba ovakvih algoritama može sa sigurnošću identifikovati identične fajlove i funkcioniše brže od SHIS, zbog toga bi trebalo prvo koristiti algoritme kao što je MD5.

V. Problemi sa performansama

Pošto se ulazi nekad moraju obraditi više od jednom da bi se dobio SHIS, vreme potrebno za formiranje SHIS može biti znatno veće od sličnih heš programa. Za potrebe ovog rada kreirano je nekoliko fajlova koji sadrže pseudo-slučajne podatke, za kreiranje fajlova korišten je alat *dd* i fajl/uređaj */dev/urandom* na virtuelnom računaru sa Fedora distribucijom Linux sistema. Generisani fajlovi su hešovani upotrebom navedenih heš programa za šifrovanje i upotrebom ssdeep programa. Vreme koje je bilo potrebno za rad je prikazano u tabeli 1 u nastavku, vreme je prikazano u milisekundama.

Tabela 1: Prikaz vremena potrebnog za hešovanje.

Algoritam	***1 MB***	***10 MB***	***50 MB***
MD5	9	49	223
SHA256	24	184	897
ssdeep	71	669	6621
Whirlpool	156	1505	7518

Kao što se vidi u tabeli 1 ssdeep je značajno sporiji od heš programa za šifrovanje sa izuzetkom Whirpool algoritma. Takođe je važno napomenuti da dva fajla iste dužine mogu imati različito vreme obrade u programu koji vrši SHIS zbog broja interakcija obrađivanja koje se moraju izvršiti na ulazu. Pod pretpostavkom da jedna SHIS obrada traje *O(n)* vremena, veličina bloka mora se prilagodi *O(log n)* puta, formirajući vreme trajanja na *O(n log n)*.

Kada se vrši poređenje para SHIS potpisa, program mora porediti potpis nepoznatog fajla sa sa svakim poznatim potpisom. Šifrovani potpisi mogu se sortirati ili staviti u heš tabelu da bi se smanjilo vreme potrebno za poređenje poznatog sa nepoznatim hešom na *O(log n)*, gde je n dužina heša koja je uglavnom konstantna. Pošto se poređenje mora izvršiti između svakog poznatog SHIS potpisa i nepoznatog heša, razmak između izmena se računa za svaki par. Računanje razmaka između izmena traje $O(l^2)$ vremena, gde *l* predstavlja promenljivu koja se može i ograničiti. Vreme potrebno da se uporedi jedan SHIS potpis sa blokom od n poznatih potpisa je *O(n)* [6].

VI. Primena

Tehnika segmentnog hešovanja iniciranog sadržajem (*SHIS*) opisana u ovom radu je izuzetno primenjiva u računarskoj forenzici. U ovom delu obrađeni su primeri primene kao i analiza rezultata. Program ssdeep se u nastavku koristi za generisanje i davanje rezultata.

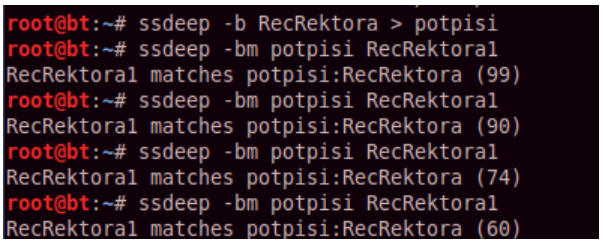
SHIS se može koristiti za identifikovanje dokumenata koji su veoma slični ali ne identični. Kao primer kreiran je tekstualni dokument koji sadrži 292 reči, koje predstavljaju reč rektora preuzete sa sajta Univerziteta Sinergija. Kreirani dokument je tekstualni fajl sa .txt ekstenzijom. Zatim je kopiran postojeći fajl RecRektora.txt i kreiran je fajl RecRektora1.txt, u novom fajlu dodat je jedan simbol (zarez). Poređenjem klasičnom metodom MD5 hešovanja dobijena su dva potpuno različita heš potpisa, što se može videti na slici 1.

```
root@root:~# md5deep RecRektora RecRektora1
9d66f016a41056557f8973d963803ed6  /root/RecRektora
aa75ac7017ac25b3a31659e9225596b5  /root/RecRektora1
```

Sl. 1. MD5 hešovi fajlova

Zatim je upotrebljena metoda SHIS koja je kao rezultat

indeksa poklapanja dala 99. U nastavku u fajlu RecRektora1 izbrisan je kompletan pasus kao rezultat je dobijen indeks poklapanja od 90 zatim 2 pasusa i rezultat je bio 74. Kao dodatni test izvršeno je ubacivanje dodatnog teksta u vidu pasusa od 109 reči rezultat poklapanja nakon ove izmene je bio 60. Prikaz rezultata se može videti na slici 2.



Sl. 2. Prikaz rezultata poređenja fajlova

Još jedna primena tehnologije *SHIS* je poređenje nepotpunih fajlova. Za fajl od *n* bita, kreiran je novi fajl koji sadrži samo *n/3* prvih bita originalnog fajla. SHIS potpis originalnog fajla se može koristiti za poređenje i pronalaženje podudarnosti sa drugim fajlom. Ovakvom upotrebom mogu se porediti poznati fajlovi sa ostacima pronađenih fajlova. Dodatno, tehnika SHIS može se iskoristiti za poklapanje dokumenata čak i ako su samo dostupna podnožja (*engl. footer*) dokumenta. Ako se novi fajl kreira sa samo n/3 poslednjih bita originalnog fajla, novi fajl se može identifikovati kao povezan sa originalnim fajlom.

Poređenje nepotpunih fajlova, posebno podnožja, je važno jer daje nove mogućnosti u forenzičkom ispitivanju. JPEG fajl (slika), na primer, u slučaju nedostatka zaglavlja fajla neće biti moguće prikazati sliku. Metodom *SHIS* mogu se porediti čak i takvi fajlovi koji se ne mogu pregledati konvencionalnim metodama.

VII. ZAKLJUČAK

Segmentno hešovanje inicirano sadržajem je izuzetno efikasna metoda u računarskoj forenzici. Omogućava forenzičarima da tokom pregleda identifikuju fajlove koji bi inače mogli biti izgubljeni u velikoj količini podataka. Pronalaženjem povezanosti između homogenih ali ne identičnih fajlova, veoma brzo i jednostavno se mogu pronaći važni delovi materijala potrebni za istragu. Iako *SHIS* nije nova tehnologija, njegova primena u forenzici predstavlja korak dalje od upotrebe klasičnih algoritama za hešovanje.

U ovom radu prikazana je nova tehnika građenja heš potpisa kombinovanjem nekoliko klasičnih heš vrednosti čije su granice određene ulaznim sadržajem. Ovi potpisi mogu se koristiti za identifikovanje izmenjenih verzija poznatih fajlova, čak iako je određeni podatak dodat, izmenjen ili uklonjen iz nove verzije fajla. Pored upotrebe na fajl sistemu takođe postoji mogućnost upotrebe u optimizaciji Web performansi u sistemima kao što je DMS [7].

LITERATURA

[1] http://samba.org/ftp/unpacked/junkcode/spamsum/README; dostupno: maj 2012.
[2] http:// dcfldd.sourceforge.net; dostupno: maj 2012.
[3] http://www.isthe.com/chongo/tech/comp/fnv/index.html; dostupno: jun 2012
[4] Kornblum Jesse: Identifying almost identical files using context triggered piecewise hashing, ScienceDirect, 2006.
[5] http://ssdeep.sourceforge.net/; dostupno: jun 2012.
[6] Breitinger Frank, Baier Harald: Performance Issues about Context-Triggered Piecewise Hashing, Center for Advanced Security Research Darmstadt, 2011
[7] A. Jevremović, R. Popvić, D. Živković, M. Veinović, G. Shimic, DMS – Improving Web Performance, Serbia, Belgrade, Novembar 22-24, 2011

ABSTRACT

Homogeneous files share an identical sequence of bits, arranged in the same order. Since these files are not completely identical, traditional techniques such as hashing can be used for finding them. In this paper we apply a new technique for building hash signature that uses a combination of several traditional hash values whose boundaries were defined by input content. Originally this technique was used to identify spam messages. This paper is showing the use in computer forensics. These signatures can be used to identify modified versions of known files, even though the data is added, changed or removed from the new version of a file. Description of the method is followed by a short analysis of its effects, as well as example applications for the computer forensics.

IDENTIFYING HOMOGENEOUS FILES USING PIECEWISE HASHING INITIATED BY CONTENT

Nenad Ristić, Aleksandar Jevremović, i Mladen Veinović