Memory onderzoek : Private browsing

Door Tycho Veltmeijer

Inhoudsopgave

1. Inleiding

Het memory onderzoek is begonnen door structureel te zoeken in het virtuele geheugen van elke browser. We zorgden ervoor dat in elke browser een aantal cookies geplaatst waren en dat een aantal websites bezocht waren, dit uiteraard terwijl de private modus van de browser aanstond. In de dump van het virtuele geheugen zochten we dan naar de URL's en cookies die we daar zouden verwachten. Als we deze vonden keken vervolgens naar de bytes die hier omheen stonden. Dit proces herhaalde we enkele keren en de bytes die we vonden zetten we onder elkaar. Door het analyseren van deze bytes hoopten we reeksen bytes te vinden die altijd een vaste waarden hebben. Met deze vaste waardes is het namelijk mogelijk om geautomatiseerd cookies en URL's uit een stuk geheugen te halen. Als dit gelukt was probeerde we erachter te komen wat de andere bytes betekenden, die tussen de vaste waarde en de cookie/URL stond.

Eenmaal geautomatiseerd gingen we op grote schaal websites bezoeken in zowel private modus als gewone modus om vervolgens het gemaakte filter los te laten op het virutele geheugen van een webbrowser, maar ook op een dump van het volledige fysieke geheugen en/of het wisselbestand. Door de uitkomst van het filter te bekijken konden we het filter weer aanscherpen, maar ook concluderen of we met het filter alleen bezochte websites meekregen of ook 'ruis'. Ruis, alsin webistes die niet bezocht zijn maar om andere redenen in het geheugen staan.

De cookies die geplaatst zijn in de eerste fase van het onderzoek zijn twee cookies (naam waarde en geldigheidsduur gescheiden door een nieuwe lijn):

- koekje_30_dagen
  houdbaarheid 30 dagen
  Moet verwijderd worden na 30 dagen na de dag dat deze geplaatst is.
- koekje_sessie
  houdbaarheid hele sessie
  Moet verwijderd worden zodra de browser gesloten is.

De pagina's die bezocht zijn in de eerste fase van het onderzoek:

- browser.test.tvsoftware.nl/
- browser.test.tvsoftware.nl/cookie/
- browser.test.tvsoftware.nl/flash/66000000.swf
- browser.test.tvsoftware.nl/flash/
- browser.test.tvsoftware.nl/img/6780687_700b_v1.jpg
- browser.test.tvsoftware.nl/img/test.js
- browser.test.tvsoftware.nl/img/
- browser.test.tvsoftware.nl/java/karakters.txt
- browser.test.tvsoftware.nl/java/plaatsen-rb.txt
- browser.test.tvsoftware.nl/java/vuurpokemon.txt
- browser.test.tvsoftware.nl/java/WordSearchNL.class
- browser.test.tvsoftware.nl/java/WSButton.class
- browser.test.tvsoftware.nl/java/WSGrid.class
- browser.test.tvsoftware.nl/java/WSList.class
- browser.test.tvsoftware.nl/java/WSWord.class
- browser.test.tvsoftware.nl/java/

2. Chrome

2.1 Het onderzoek

In Chrome is net zo als in andere browsers gezocht naar structuren, echter is Chrome hier een speciaal geval en noemt niet netjes in het geheugen wat elk stukje betekend . Om deze reden zijn we gaan zoeken naar pointers, in de hoop een mooi lijstje tegen te komen met cookies of URL's. De bevindingen staan hieronder

Het zijn twee cookies waar naar gezocht zijn, Koekjes_30_dagen (A) en koekjes_sessie (B). Zoals zichtbaar kan er

- Een pointer naar het domein gevonden worden, waar de cookie geldig is
- Een pointer naar de waarde van de cookie
- De naam van de cookie, samen met de locatie van waar de cookie geldig is, zijn in het geheugen blok zelf terug te vinden.

Dit lijkt een mooie vondst, echter was het verwijderen van alle cookies, herstarten van Chrome en het opnieuw plaatsen van de cookies genoeg om deze structuur nergens in het geheugen meer terug te vinden. Hierna is er tevergeefs gezocht naar andere structuren, die ook steeds van vorm schenen te veranderen. Helaas is dit onderzoek dan ook gestrand zonder resultaat.

A Koekjes_30_dagen
B koekjes_sessie

	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
A)	30	EC	89	09	26	98	FC	FF	00	02	88	09	00	00	00	00	20	00	00	00	2F	00	00	00	00	00	00	00
B)	B0	61	89	09	FF	FF	FF	FF	21	00	00	00	0C	00	00	00	22	00	00	00	2F	00	00	00	00	00	00	00

Pointer naar domein waar cookie geldig is

Grootte van de buffer hierna tot gap

Gap

	1D-2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
A	Koekjes_30_dagen	00	0F	00	00			00	0F	00	00	00	FF	FF	FF	FF	E0	AD	C9	08
B	koekjes_sessie	00	54	00	0D	00	00	00	0F	00	00	00	00	00	00	00	A0	10	88	0A

Naam van cookie

Pointer naar de waarde van de cookie

	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56
A	FF	FF	FF	FF	07	00	00	00	1A	00	00	00	15	00	00	00	1F	00	00	00	21	00	00
B	0F	00	00	00	00	00	00	00	00	00	00	00	18	00	00	00	1F	00	00	00	00	00	00

	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D
A	00	60	89	CF	08	00	00	00	00	FF	FF	FF	FF	00	00	00	00	1A	00	00	00	1F	00
B	00	E0	10	88	0A	0F	00	00	00	00	00	00	00	00	00	00	00	1A	00	00	00	1F	00

```
   6E 6F 70 71 72 73  74-7D
A 00 00 00 00 00 00 /cookie
B 00 00 01 00 00 00 /cookie
```

Locatie op domein waar cookie geldig is

3. Firefox

3.1 URL's

Dit onderzoek is bedoeld om te kunnen achterhalen naar welke URL's een gebruiker van Firefox heeft bezocht, deze URL's moeten uit het geheugen van Firefox gehaald worden. Er zijn meerdere methodes ontwikkeld, allemaal met eigen voor- en nadelen.

Methode 1: Het eerst dat opviel tijdens het bekijken naar de geheugen dump van Firefox was dat Unicode (UTF-16) strings bijna altijd begonnen met 4 bytes die altijd dezelfde waarde bevatte: '0x00000001'. Na deze 'handtekening' komen 4 bytes die als waarde de lengte van de string bevat, na deze waarde komt de string. Veel van deze strings bevatten URL's, interessant genoeg dus om dit te automatiseren en kijken of dit het gewenste resultaat geeft.
Geautomatiseerd ziet dit er als volgt uit:

- Eerst wordt er gezocht naar 0x00000001.
- De waarde die hierna komt wordt uitgelezen (wat de lengte van de string zou moeten voorstellen)
- Vervolgens wordt de lengte van de string berekend, komt deze overeen met de waarde hierboven dan is de kans groot dat we een string te pakken hebben.
- Vervolgens wordt er nog een filter op toegepast die controleert of de string die we te pakken hebben wel een URL is. Ook verwijderd dit filter chrome:// adressen, dit is een ingebouwd protocol in Firefox die standaard pagina's weergeeft, zoals de 'niet gevonden' pagina.

Eigenschappen van deze methode:

- Het betreft veel URL's, niet alleen URL's die door de gebruiker bezocht zijn. Dit kan mogelijk verholpen worden door dezelfde methode toe te passen op een net opgestarte Firefox en de twee lijsten te vergelijken.
- Na het afsluiten van de private sessie vind deze methode geen URL's meer die tijdens de private sessie bezocht zijn.
- De URL's zijn in unicode (UTF-16) formaat

Methode 2: Tijdens de private sessie worden URL's die door Firefox worden opgevraagd in het geheugen opgeslagen met de string 'HTTP-memory-only-PB:' voor de URL. Deze filter is makkelijk toe te passen. Als de string 'HTTP-memory-only-PB' gevonden wordt weet je zeker dat deze bezocht is tijdens de private sessie.
Eigenschappen van deze methode:

- Geen ruis, er worden alleen URL's gevonden die bezocht zijn tijdens de private sessie.
- Alle URL's worden na de private sessie uit het virtuele geheugen verwijderd, deze methode heeft dan ook geen resultaat als de private sessie is afgesloten.
- De URL's kunnen wel nog achterblijven in het Fysieke geheugen / wisselbestand
- De URL's zijn in ASCII formaat

Methode 3: Tot slot is er nog gezocht naar structuur. Het was de bedoeling om naar pointers te zoeken die naar URL's verwezen. Mogelijk konden we op deze manier lijsten vinden, of verwijzingen naar lijsten, die bezochte URL bevatten. Dit onderzoek heeft helaas geen resultaat opgeleverd.

3.2 Cookies

Dit onderzoek is bedoeld om cookies geautomatiseerd te kunnen gaan zoeken in het virtueel geheugen van Firefox. Zoekend naar de cookies in het virtuele geheugen kwamen we erachter dat elke cookie een header heeft. In eerste instantie dachten we dat deze header herkenbaar was aan vier bytes '98 FE C6 52'. Voor de zekerheid hebben we dezelfde test gedaan op een ander systeem, daar bleek uit dat de handtekening van de header maar uit twee bytes bestaat. Nu de header gevonden is kunnen we gaan onderzoeken wat de waardes betekenen, het resultaat van dit onderzoek staat onderaan de pagina.

Na de header komen de volgende waardes, in deze volgorde, in het virtueel geheugen voor (alle tekenreeksen zijn ASCII, afgesloten met een nul-byte):

- De naam van de cookie.
- De waarde van de cookie.
- Het domein waar de cookie geldig is.
- De locatie op het domein waar de cookie geldig is.

De cookie kan automatisch gevonden en gevalideerd worden. Volg de onderstaande stappen om een cookie te valideren:

- Allereerst moet er gezocht worden naar het begin van de header “98 FE” (in Firefox 21 is deze waarde verandert naar “40 07”).
- Het *header*, zoals hieronder beschreven, is 64 bytes groot, er moet gecontroleerd worden of het geheugen dat gecontroleerd wordt nog wel deze grootte heeft.
- Na de header komt de *naam* van de cookie (afgesloten met een nul byte), de lengte van deze string is nooit nul bytes groot, bereken dus hoe groot deze string is.
- Dezelfde controle kan je uitvoeren voor de *domeinnaam* die na de string *waarde* komt.
- De *domeinnaam* kan bestaan uit de tekens: a t/m z (let op geen hoofdletters), 0 t/m 9, een liggend streepje en een punt. Controleer of er vreemde tekens in voorkomen.
- De *locatie* begint altijd met een schuine streep ('/'), controleer of deze er staat.

Na deze stappen is zo goed als zeker dat het stuk geheugen dat je aan het analyseren bent een cookie bevat. Helaas is echter een kleine marge dat geen cookie is en die wel als zo wordt aangemerkt door het filter, vooral als deze filter op een volledige hardeschijf wordt toegepast. Dit kan verbeterd worden door de filters aan te scherpen. Zo kan je de locatie controleren (of er vreemde tekens in voorkomen) of je kan de *waarde* van de cookie controleren op spaties, deze staan er namelijk als een plusje ('+') ipv een spatie en mogen er dus ook niet in voorkomen. De testen die wij uitvoeren is de foutmarge van gevonden cookies echter zo klein dat we het bij de bovengenoemde controles hebben gehouden.

Hieronder is voor elke byte in het header van de cookie beschreven wat deze betekend en/of wat wij denken dat het mogelijk betekend.

Register:
P = De nummering van elke byte beginnend bij 1 in hexadecimaal formaat
A = De header die hoort bij cookie koekje_30_dagen
B = De header die hoort bij cookie koekje_sessie

```
P:  01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20
A:  98 FE C6 52 01 00 00 00 00 0E BA 0B 10 0E BA 0B 26 0E BA 0B 41 0E BA 0B 49 0E BA 0B 00 00 00 00
B:  98 FE C6 52 01 00 00 00 90 0E BA 0B 9E 0E BA 0B B7 0E BA 0B D2 0E BA 0B DA 0E BA 0B 00 00 00 00
A:  98 FE D9 6E 01 00 00 00 B0 FD F6 08 C0 FD F6 08 D6 FD F6 08 F1 FD F6 08 F9 FD F6 08 00 00 00 00
B:  98 FE D9 6E 01 00 00 00 40 FE F6 08 4E FE F6 08 67 FE F6 08 82 FE F6 08 8A FE F6 08 00 00 00 00
A:  98 FE D9 6E 01 00 00 00 60 36 E9 08 70 36 E9 08 86 36 E9 08 A1 36 E9 08 A9 36 E9 08 00 00 00 00
B:  98 FE D9 6E 01 00 00 00 C0 52 99 09 CE 52 99 09 E7 52 99 09 02 53 99 09 0A 53 99 09 00 00 00 00
A:  98 FE D9 6E 01 00 00 00 C0 69 07 08 D0 69 07 08 E6 69 07 08 01 6A 07 08 09 6A 07 08 00 00 00 00
B:  98 FE D9 6E 01 00 00 00 B0 26 2B 0A BE 26 2B 0A D7 26 2B 0A F2 26 2B 0A FA 26 2B 0A 00 00 00 00
A:  98 FE C6 52 01 00 00 00 00 0E BA 0B 10 0E BA 0B 26 0E BA 0B 41 0E BA 0B 49 0E BA 0B 00 00 00 00
B:  98 FE C6 52 01 00 00 00 90 0E BA 0B 9E 0E BA 0B B7 0E BA 0B D2 0E BA 0B DA 0E BA 0B 00 00 00 00
```

Lijkt vaste waarde te zijn '98 FE', dit geld voor Firefox 19.0.2 , in Firefox 21 is dit de waarde '40 07'

Nog onbekende waardes (30 bytes)

```
P: 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40
A: 5A 69 82 51 00 00 00 00 B8 DB D5 AC 60 D9 04 00 B8 DB D5 AC 60 D9 04 00 00 00 00 00 00 00 00 00
B: FF FF FF FF FF FF FF 7F B8 DB D5 AC 60 D9 04 00 B9 DB D5 AC 60 D9 04 00 01 00 00 00 00 00 00 00
A: DF 83 82 51 00 00 00 00 18 9E 7C 41 62 D9 04 00 18 9E 7C 41 62 D9 04 00 00 00 00 00 00 00 F0 3F
B: FF FF FF FF FF FF FF 7F 18 9E 7C 41 62 D9 04 00 19 9E 7C 41 62 D9 04 00 01 00 00 00 00 00 00 00
A: 03 A3 82 51 00 00 00 00 38 B0 9F 1C 64 D9 04 00 38 B0 9F 1C 64 D9 04 00 00 00 00 00 00 00 F0 3F
B: FF FF FF FF FF FF FF 7F 38 B0 9F 1C 64 D9 04 00 39 B0 9F 1C 64 D9 04 00 01 00 00 00 00 00 F0 3F
A: D3 A6 82 51 00 00 00 00 10 03 D1 56 64 D9 04 00 10 03 D1 56 64 D9 04 00 00 00 00 00 00 00 F0 3F
B: FF FF FF FF FF FF FF 7F 10 03 D1 56 64 D9 04 00 11 03 D1 56 64 D9 04 00 01 00 00 00 00 00 F0 3F
A: 5A 69 82 51 00 00 00 00 B8 DB D5 AC 60 D9 04 00 B8 DB D5 AC 60 D9 04 00 00 00 00 00 00 00 00 00
B: FF FF FF FF FF FF FF 7F B8 DB D5 AC 60 D9 04 00 B9 DB D5 AC 60 D9 04 00 01 00 00 00 00 00 00 00
```

Cookie is geldig tot deze datum, gerepresenteerd in secondes vanaf 1 januari 1970 00:00:00. Zijn alle waardes op FF (op de laatste na, 7F) dan is de cookie alleen geldig voor de huidige browser sessie. (8 bytes)

Twee reeksen die bijna uniek zijn per browser sessie. De eerste 8 bytes zijn bijna hetzelfde als de 8 bytes die daarop volgen, alleen de eerste byte van de tweede reeks wijkt af. Mogelijk pointers naar virtueel geheugen? (16 bytes)

Mogelijk flags (geldig voor sessie: 1, geldig voor bepaalde tijd: 0)

Of de telling van aantal cookies (4 bytes)

Tot nu toe altijd nul (2 bytes)

Nog geen logica in kunnen vinden, soms '00 00'. soms 'Fo 3F'

4. Internet Explorer

4.1 URL's

Dit onderzoek is bedoeld om te kunnen achterhalen naar welke URL's een gebruiker van Internet Explorer heeft bezocht, deze URL's moeten uit het geheugen van Internet Explorer gehaald worden.

Internet Explorer slaat van elke URL die je bezocht hebt zowel de volledige URL als de gebruiker die deze bezocht heeft op in het geheugen. staat dit vervolgens in het volgende formaat:: “Visited: gebruiker@URL”. Het gaat verder dan alleen URL's van websites, ook bestanden of elk denkbaar protocol dat een gebruiker geopend heeft met Internet Explorer of Explorer worden op deze manier in het geheugen opgeslagen.

Stappen bij het geautomatiseerd zoeken URL's in het geheugen zijn:

- Zoeken naar de string “Visited: ”
- Zoeken naar het karakter “@”, eindigt de string eerder dan dat je het apenstaartje vind dan is het geen URL die bezocht is.
- Vanaf het apenstaartje controleren of het een protocolnaam betreft (tot aan de dubbelepunt). Deze mag bestaan uit kleine letters, cijfers een streepje en een punt.
  Opmerking: Je kan ook alleen controleren of er “http://” staat, dit zorgt er echter voor dat je veel informatie mist aangezien het protocol als “file://” en “magnet:” in de resultaten zal missen.

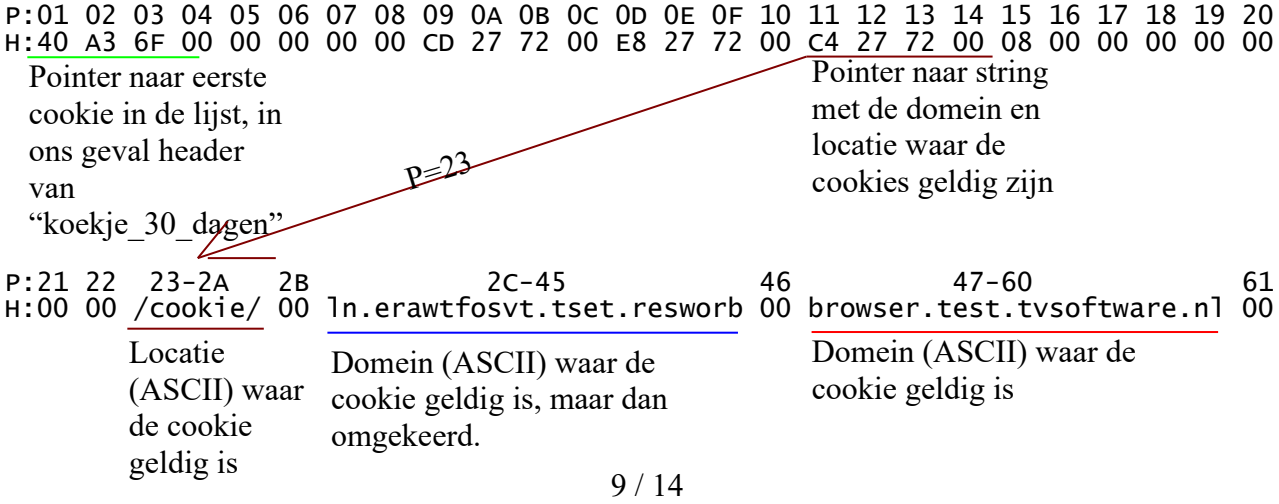
Eigenschappen:

- De volledige string kan in ASSCI en in Unicode (UTF-16) in het geheugen aanwezig zijn (zowel “Visited: “ als de URL)
- Het bevat alle protocollen die geopend zijn en dus niet alleen 'http'
- Private modus is niet van normale modus te onderscheiden
- Het is duidelijk vanuit welk account de URL geopend is

4.2 Cookies

Dit onderzoek is bedoeld om cookies geautomatiseerd te kunnen gaan zoeken in het virtueel geheugen van Internet Explorer. Zoekend naar de cookies in het virtuele geheugen hebben we meerdere headers gevonden die met elkaar verbonden zijn, de headers staan hieronder beschreven.

Overkoeplende header:



Header koekje_30_dagen:

```
P:01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18
H:FF FF FF FF FF FF FF 7F 3A 55 3A 9C BE 69 CE 01 02 04 00 00 00 A3 6F 00
```

Geldig tot deze datum, echter staat deze in de inPrivate modus altijd op "tot einde sessie"

Dit is een pointer naar een volgende cookie. In dit geval een pointer naar de header van koekje_sessie

```
p:19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28
H:68 A3 6F 00 C8 E4 71 00 0F 00 15 00 03 00 00 00
```

Pointer naar de naam van de cookie als ASCII string (in dit geval "koekje_30_dagen")

Pointer naar de waarde van de cookie als ASCII string (in dit geval "Houdbaarheid+30+dagen")

Header koekje_sessie:

```
P:01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18
H:FF FF FF FF FF FF FF 7F 5B 79 41 9C BE 69 CE 01 02 04 00 00 00 00 00 00
```

Geldig tot deze datum, echter staat deze in de inPrivate modus altijd op "tot einde sessie"

Dit is een pointer naar een volgende cookie. Er is geen volgende cookie, deze waarde is dus nul

```
P:19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28
H:28 A3 6F 00 B8 8B 74 00 0D 00 18 00 03 00 00 00
```

Pointer naar de naam van de cookie als ASCII string (in dit geval "koekje_sessie")

Pointer naar de waarde van de cookie als ASCII string (in dit geval "houdbaarheid+hele+sessie")

Om het overzichtelijker te maken heb ik de headers uitgewerkt in C syntax.

```
typedef struct header {
	header_cookie* firstCookie;
	char unknown[12];
	valid_location* ValidAt;
} header;

typedef struct valid_location {
	char location[?];
	char domainReverse[?];
	char domain[?];
} valid_location;
```

```
typedef struct header_cookie {
	long long valid_date;
	char unknown[12];
	header_cookie* next;
	char* name;
	char* value;
	char unknown2[8];
} header_cookie;
```

Uitleg headers
Zoals uit de uitwerkentingen hierboven duidelijk mag zijn worden alle cookies gegroepeerd d.m.v domeinnaam en locatie waar deze geldig zijn (structuur header). De header verwijst naar de eerste cookie (structuur header_cookie). Dit is ook de eerste die geplaatst is voor deze domein en locatie, in ons geval is dit van koekje_30_dagen. In deze header staat een verwijzing naar de naam en een verwijzing naar de waarde van de cookie. Ook staat er een verwijzing naar een volgend cookie (in ons geval koekje_sessie). Deze heeft weer dezelfde indeling, alleen op de plek waar normaal een verwijzing staat naar de volgende cookie staan nu alleen maar nullen. Dit betekend dat dit de laatste cookie is voor deze domein en locatie .

De header verwijst ook naar een string met de locatie en domein waar de cookies geldig zijn (valid_location). Het vreemde is dat de domein er twee keer in staat en één daarvan in omgekeerde richting. Zo staat er bijvoorbeeld “ln.un” en “nu.nl”. We zijn er niet achtergekomen waarom dit zo is, maar gebruiken dit wel om geautomatiseerd te zoeken naar cookies.

Automatiseren
De stappen voor het automatisch zoeken (Live en static forensics):

- Zoeken naar een string die groter is dan 3 bytes
- Controleren of deze alleen bestaat uit letters, cijfers, punt en een streepje
- Controleren of de string die daarna komt dezelfde string is, maar dan in omgekeerde richting geschreven
- Controleren of er nog een string voor staat
- Het begin van deze string zoeken door naar de nul-byte te zoeken in tegenovergestelde richting (hij staat er immers ervoor, niet erachter)
- Controleren of deze string begint met '/' (dit is namelijk de locatie waar de cookies geldig zijn)
- In dit stadium hebben we header “valid_location” gevonden

Met static forensics kan je helaas alleen maar op zoek gaan naar de domeinen en bijbehorende locaties waar de cookies geldig zijn. De daadwerkelijke cookies die erbij horen zijn alleen terug te vinden met live forensics. Dit komt omdat alle verwijzingen in het geheugen gebaseerd zijn op de virtuele geheugen adressering. Bij static forensics is deze vaak niet meer te achterhalen, indien deze wel te achterhalen valt kan je natuurlijk verder zoeken naar de cookies.

De stappen om de cookies te vinden die bij de gevonden domein / locatie horen (alleen live forensics en, indien de virtuele geheugen adressering nog terug te halen valt, ook static forensics):

- Zoeken in het hele geheugen of de verwijzing (pointer) naar het gevonden header “valid_location” voorkomt.
- Indien gevonden moet er gecontroleerd worden of de 8 bytes die *min* 20 bytes vanaf de gevonden pointer staan, leeg zijn (alle bytes op nul). Dit is tot nu toe bij al onze testen zo geweest en is dus een mooie controle of we structuur “header” gevonden hebben. Indien dit niet het geval is kan er verder gezocht worden met stap 1 totdat de pointer niet meer voorkomt in het geheugen.
- Op *min* 16 bytes vanaf de gevonden verwijzing (pointer), staat een verwijzing (pointer) naar de eerste cookie (“header_cookie”).
- Vanaf dit punt kunnen we de cookies één voor één uitlezen.

5. Safari

5.1 URL's

Dit onderzoek is bedoeld om te kunnen achterhalen naar welke URL's een gebruiker van Safari heeft bezocht, deze URL's moeten uit het geheugen van Safari gehaald worden. Zoekend naar de URL's in het virtuele geheugen bleek de URL's vaker voor te komen met verschillende headers, de resultaten staan hieronder per header.

Header 1:
8C 07 00 01 xx

Hiermee is xx de grootte van de string die achter deze reeks aan bytes komt. De string hoeft niet persé, maar kan wel, een URL bevatten. Ook hoeft deze URL niet bezocht te zijn en tot slot is de maximale lengte van de URL gelimiteerd aan 255 bytes. Deze header is dus niet interessant genoeg.

Header 2:
05 00 00 00 00 00 00 xx xx xx xx 00 00 00 00

Vaste waarde	Lengte van de string die na de header volgt	Tot nu toe altijd nul, mogelijk verlengde van de lengte van de string

Eigenschappen:

- De string in ASCII formaat
- Betreft vaak een URL, soms ook een andere string
- Indien het een URL betreft is deze bezocht
  - Direct
  - Indirect (bijvoorbeeld afbeelding, script die op een pagina staan)
- Het betreft zowel private als normaal gebrowsde URL's
- De string die hier op volgt eindigt **niet** met een nul-byte
  - Controleer bij automatiseren of
    - de string gelijk of groter is dan opgegeven grootte in de header om het gros van foute headers eruit te filteren.
    - het een URL betreft
      - domein controleren (a-z, 0-9 – en .) tot de '/' of '\'
      - De rest van de url controleren op nieuwe lijnen en spaties die er niet in voor mogen komen

Header 3:

```
4C 00 69 00 6E 00 6B 00 55 00 52 00 4C 00 00 00 10 00 00 00
xx xx xx xx xx xx xx xx 00 00 00 00
```

String “Link”		String “URL”	Onbekende (vaste) waarde	Onbekende variërende waarde
Aantal karakters in string (unicode)	Grootte van string in bytes	Tot nu toe altijd nul, mogelijk verlengde van de grootte van de string		

Eigenschappen:

- De URL is in unicode (UTF-16)
- De string “LINK” hoeft niet aanwezig te zijn en er kan dus alleen “URL” staan.
  - Indien “LINK” **niet** aanwezig is,dan is er bewust geklikt op een link door de gebruiker. De URL die achter de header staat is waar de link heen verwees op het moment dat de gebruiker er op klikte.
  - Indien “LINK” **wel** aanwezig is lijkt het een URL te zijn die ooit op een pagina heeft gestaan, dit hoeft echter **niet** te betekenen dat de gebruiker op deze link heeft geklikt of er heen is geweest.
- Deze vorm verdwijnt snel uit het geheugen en je zal ook niet veel resultaten vinden bij een geheugen onderzoek.

5.2 Cookies

Dit onderzoek is bedoeld om cookies geautomatiseerd te kunnen gaan zoeken in het virtueel geheugen van Safari. Zoekend naar de cookies in het virtuele geheugen bleken ze vaker voor te komen met verschillende headers, de resultaten staan hieronder per header.

- `8C 07 00 01 xx` {naam}={waarde}; expires={date: Wed, 26-Jun-2013 17:32:35 GMT},
  Na de handtekening van 4 bytes komt de grootte van de string (1 byte). Vervolgens kom de naam van de cookie en daarachter de waarde met daartussen het teken '='. Hierna komen de eigenschappen van de cookie, gescheiden door een punt-komma (zoals de optie expire). Hierna volgt een komma waarna een nieuwe cookie wordt genoemd. De handtekening van 4 bytes was echter te klein waardoor er veel meer hits bestonden dan alleen cookies, om deze reden is deze structuur niet geautomatiseerd.
- `53 65 74 2D 43 6F 6F 68 69 65 xx xx xx xx xx xx 07 00 00 00 00 00 00 00 00 05 00 00 00 00 00 00 xx xx xx xx 00 00 00 00`
  De handtekening begint met de string “Set-Cookie” (10 bytes), hierna komen 6 bytes die vaak maar niet altijd op nul staan, deze zijn nog niet geïdentificeerd. De 16 bytes die daarna komen lijken vast te zijn. De 4 bytes die daarop volgen zijn de grootte van de string die na het header volgt, de vier bytes daarna komen lijken altijd op nul te staan. Hierna komt de string die hetzelfde formaat heeft als bij de handtekening die hierboven beschreven staat.

Het moet opvallen dat er niet gesproken wordt van een domein waar deze cookie bij hoort, dit komt omdat er geen verbanden gevonden konden worden tussen de cookie en de domein waar de cookie geldig is. In sommige gevallen staat er bij de eigenschappen die bij de cookie horen (zoals eigenschap 'expire') een eigenschap 'domain' en 'path'. Dit is echter niet bij elk cookie aanwezig.

Wel hebben we een verband kunnen maken tussen de cookie en de pagina die deze cookie geplaatst

heeft (hieruit kan ook een domein opgemaakt worden waar deze geldig is). Dit hebben we ook kunnen automatiseren, echter komt hier niet altijd het juiste resultaat uit. Onder de cookie zijn meerdere strings aanwezig die verband hebben met de geplaatste cookie, deze hebben allemaal een zelfde soort structuur. Door vanaf locatie van de cookie naar beneden te zoeken naar deze structuur en controleren of de gevonden structuur een URL bevat kan geautomatiseerd gezocht worden naar de URL die de cookie geplaatst heeft. Dit geeft echter niet altijd het juiste resultaat, er moet altijd kritisch naar gekeken worden of het resultaat van deze zoek methode wel de juiste URL oplevert.

Hoe deze structuur is opgebouwd:

```
05 00 00 00 00 00 00 xx xx xx xx 00 00 00 00
```

Vaste waarde	De lengte van de string	Tot nu toe nul, mogelijk hoort dit bij de lengte

Achter deze structuur komt de string, door te controleren of deze en URL bevat kan negen van de tien keer (uit onze testen) automatisch de URL achterhaald worden die de cookie geplaatst heeft.

Om deze documentatie makkelijker te begrijpen is hier een voorbeeld van een cookies zoals die in het geheugen van safari staat (deels hex deels string):

Voorbeeld van het header in het geheugen:

```
P: 01 02 03 04 05 06 07 08 09 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
H: 53 65 74 2D 43 6F 6F 6B 69 65 00 37 00 00 00 07 00 00 00 00 00 00 00 00 05 00 00 00 00 00 00
```

De string “Set-Cookie”.	Onbekende random bytes.	(lijkt) een vaste range aan bytes.

```
P: 30 31 32 33 34 35 36 37                         38-7F
H: 74 00 00 00 00 00 00 00 koekje_30_dagen=houdbaarheid+30+dagen; expires=Wed, 26-jun-2013 20:06:13
```

De grootte van de string	Tot nu toe steeds nul (mogelijk hoort het bij grootte van string)

```
P:            80 - AB
H:  GMT, koekje_sessie=houdbaarheid+hele+sessie
```