Forensic Science International: Digital Investigation 36 (2021) 301091

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

AIBFT: Artificial Intelligence Browser Forensic Toolkit

Hyunmin Kim [b], InSeok Kim [b], Kyounggon Kim [a, b, *]

[a] Department of Forensic Sciences, Naif Arab University for Security Sciences, Riyadh, Kingdom of Saudi Arabia
[b] School of Cybersecurity, Korea University, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 12 April 2020
Received in revised form
12 October 2020
Accepted 27 October 2020
Available online 7 November 2020

Keywords:
Artificial intelligence browser forensic artifact
AI Based forensic AIBFT

ABSTRACT

In the modern age of content, web browsers serve as an important bridge between content and users. On the other hand, web browsers are also a link to malicious codes, and new types of malicious codes that are secretly mining through browsers are being created. In recent years, a web browser investigation is essential to find the path of malicious code infection when a security incident occurs. However, the content became vast, and as network performance is getting better, websites are getting bigger, and the number of web pages included in a single website has increased exponentially. It is almost impossible to manually analyze all of the thousands of sites due to time limitations. In this paper, we propose a method to apply machine learning to web browser forensics to solve these problems. Also, we propose AIBFT: Artificial Intelligence Browser Forensic Toolkit, a Proof-of-Concept (POC) tool that provides automatic detection of malicious webpages using AI models, analysis of malicious probability, and timeline visualization. We collected 52,500 benign and malicious web pages for AI models and learning. As a result of applying the randomforest algorithm among AI algorithms, we could achieve 99.8% accuracy.

1. Introduction

In the event of a malware infection breach, the crucial step in performing digital forensics is finding the first infection path. If the infection path is not found, re-infect from the same malware through the same path is possible. Therefore, it is most important to find the first path of infection during the investigation.

Malware is infected through various attack techniques and dissemination methods. With these techniques in place, simply visiting a website without active intervention such as user clicks, or file execution can perform specific actions desired by an attacker and infect malware. Drive-By-Download (DBD) attacks are very threatening and deadly. According to the report from NTTSecurity (2019), DBD attacks can lead to the leakage of key information through the installation of keyloggers. DBD attacks are mainly carried out through web browser vulnerabilities and web browser module vulnerabilities. The exploit code that attacks vulnerabilities is downloaded and executed once a malicious code is inserted into the web page.

DBD attacks have been widely used in Advanced Persistent Threat (APT) attacks using zero-day vulnerabilities. Recently, DBD has been used to distribute cryptocurrency mining malware, which is classified as a new attack type called Drive-By-Mining (DBM). Attackers have developed and marketed exploit kits for drive-by attacks by systemizing and modularizing these attack technologies. Attackers are using exploit kits to infect multitudes of users' PCs, as shown in Fig. 1. According to Mcafee (2019), many new exploit kits such as Spelevo, Fiesta, and Fallout were discovered in 2019. A good deal of these attacks is delivered to the user's PC through the web browser, increasing the necessity and importance of web browser forensics.

Modern websites use various Javascript and Cascading Style Sheets (CSS) libraries, which are large and consist of many files. Large sites, such as portal sites, often link to dozens of sites from a single site, so the number of web pages accessed from a web browser ranges from thousands to tens of thousands. This means that the number of files to be checked by an investigator has increased exponentially in case of an infringement incident, increasing analysis time and becoming major obstacle delaying investigations.

We propose AI-based web browser forensic technology that automatically analyzes numerous web pages stored in the web browser cache folder to find malicious scripts to solve these problems. We also develop the command user interface (CUI) and graphic user interface (GUI) programs that can be used for actual investigation. By deploying, we want to improve the accuracy and

* Corresponding author.
E-mail address: kkim@nauss.edu.sa (K. Kim).

https://doi.org/10.1016/j.fsidi.2020.301091
2666-2817/

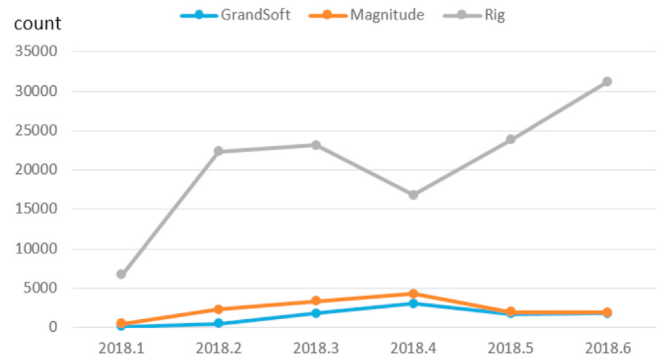


Fig. 1. Exploit-kit activity statistics.

analysis speed of web browser forensics. The contributions of this paper are as follows.

- First, we combined digital forensics and artificial intelligence concepts.
- Second, we developed a visually effective toolkit for forensic analysts.
- Third, we have implemented a reliable AI model that reaches a classification accuracy of malicious codes and normal web pages of 99.8%.

This paper is organized as follows. Section 2 reviews literature papers. In Section 3, we introduce our methodology. Section 4 demonstrates our tool. Section 5 addresses the discussion and future work of our research. Lastly, Section 6 summarizes the conclusion.

2. Literature review

Since web browser investigation is essential for digital forensics, related studies are nothing new, and there are various browser forensic tools distributed for free or commercially.

Existing forensic tools mainly focus on simply extracting data from the browser history, cache, and cookies to display a list of websites that have been visited. Rather than analyzing and processing the contents of the data to provide information, are provided mainly with the function of extracting and listing the raw data such as web browser history, cache, and cookies.

However, as mentioned in the introduction, unlike the past, modern web sites are made up of dozens of pages, so even a single PC will likely access thousands of web pages. It is a very laborious and time-consuming task for investigators to check each page individually. An obfuscated malicious webpage such as an exploit kit consists of many special characters and difficult JavaScript syntax that are difficult to analyze with the naked eye. These obfuscation pages are limited to analysis through existing forensic tools. After all, to check whether a page is malicious, it is necessary to use a separate malicious detection tool or manually with a deobfuscation tool.

Since it is challenging to analyze and detect malicious webpages among many benign webpages, various studies have been conducted to investigate automatic analysis techniques. The methods for detecting malicious webpages can be broadly classified as follows.

2.1. Pattern-based malicious web page detection

First, the pattern-based malicious webpage detection method is a method of extracting previously found malicious webpages' characteristics. This method used a string or a regular expression to determine whether or not the page is malicious through pattern matching on the webpage string. This method, called static analysis, detects a certain malicious pattern. Thus, the detection accuracy is very high when the existing malicious web page is reused, or a pattern from a malicious page is inserted into a new page. Provos, Mavrommatis, Rajab and Monrose (2008) showed that a malicious webpage was detected through malicious string-based patterns such as a hidden iframe HTML tag. Zhang, Seifert, Stokes and Lee (2011) proposed a method to detect Command and Control (C&C) server URL using a regular expression.

2.2. Behavior-based malicious web page detection

Unlike pattern-based detection, various behavior-based malicious webpage detection studies have been attempted to observe and analyze web pages dynamically. The paper Eshete et al. (2012) introduces a behavior-based malicious webpage detection technique through emulation, which produces an accuracy of 97%. However, the detection speed was slow, 3–5 s per webpage. Behavior-based detection operates by loading a page through an emulator and records and checks which functions were executed. This state-of-art technique has the advantage of easily detecting whether the obfuscated web page is malicious. However, it is difficult to detect malicious web pages that require user interaction, such as click or drag, or when multiple pages are linked, which are interactions exploit kits are targeting. Besides, there is also a disadvantage that processing and detection speed is slower than static analysis.

2.3. Machine learning-based malicious webpage detecting

Quite a few studies have been conducted on how to use machine learning to detect malicious web pages. Around 2010, mainly detection research using machine learning was conducted, then around 2016, much research on detecting malicious web pages using neural networks and deep learning was conducted. Verma and Das (2017) introduced a method of analyzing and learning URLs, but ignoring webpages' contents to detect malicious webpages quickly. Zhao and Hoi (2013) proposed a method to improve the AI model's accuracy by performing real-time learning online. In the previous study, the accuracy, which means the ratio of true positive and true negative to the entire dataset, was quite high, mostly 95–97%, but in reality, even 5% false positives have a huge impact on investigators. If there are 10,000 access records per year on one PC, when applying a false positive rate of 10%, the investigator must manually analyze 1000 web pages. Depending on the situation, when a false positive occurs, it may need to review all the positive web pages again to find the error. These issues will be compounded by the growing size of today's Web sites. Also, existing studies have no solution to false negatives, which can confuse analysis. The above-mentioned studies were mainly focused on detecting whether web pages were malicious on the network, such as web server file inspection or network packet inspection, rather than digital forensic research. There is no case where the technologies are applied to digital forensics.

2.4. Machine learning based forensics

In early 2010, the possibility of using AI in forensics began to be presented, and recently, studies on using AI in forensics in various fields are underway. A paper by Ariu et al. (2011) stated that "In real-world cases, it is difficult to apply machine learning to forensics because the environment is so diverse. However, if machine learning and investigators' opinions are properly combined, it can be very helpful."

Yeow et al. (2014) proposed an expert system to classify event types using a machine-trained model of collected information and event data. The system showed an accuracy of 75% when there were more than 200 case-specific learning data. However, in real-world events, it is difficult to classify and learn because the types of events are very diverse, and the classification accuracy is low, indicating that further research is needed to improve accuracy.

Currently, research is being conducted to apply AI to a more specialized field of forensics. Koopman, Rodriguez and Geradts (2018) proposed a method to detect manipulated video using AI. This paper proposed dividing and inspecting the frames of a video and applied AI to the video forensics. The paper by Carriquiry et al. (2019) introduces an example of applying AI to image identification and shows an example of using AI to verify the similarity of a bullet-casing in a gun crime case.

As in the above studies, there are stand-alone applications of AI technology to digital forensics. However, AI forensics is still in its infancy, and in the case of web browser forensics, there have been no cases or studies of applying AI. Although several studies have been conducted to detect and classify malicious webpages at the network stage, no AI has been applied to web browser forensics, and no AI web browser forensics tools can currently be used in the investigation of malicious webpages.

3. AI for browser forensic

When security incidents occur, the investigator can check the list of websites accessed and whether a site is malicious through a browser forensic. Finding and analyzing malicious web pages can help analyzers investigate the initial path and time of infection. However, manually finding malicious web pages in thousands of website access lists is very inefficient. To solve this problem of efficiency, we propose a malicious web page detection system that automatically identifies and classifies whether the browser's access web pages are malicious using machine learning.

3.1. Browser forensic

Various traces remain when users access a website using a web browser. Traces are left in specific ways for each browser, and there are typical access history and cache files. The access history does not store the web page's content due to being used to record the access URL and display the visited website. The browser cache speeds up webpage loading by storing webpage files in local temporary storage. When the user reconnects to the same web page, the web browser checks whether the web page has been changed and if not, the web page is not downloaded again from the website, and the file in the local storage is read and displayed. Since the web page at the time of access is stored in the cache file, it is possible to check whether the accessed web page is malicious.

It is essential from a forensic perspective that the web page at the time of access is stored in its original form. In general, malicious web pages are stored on the web server in an unusual way, such as exploitation or account stealing using vulnerabilities. Therefore, it is very likely that important content has already been deleted or modified by the administrator at the time of the investigation. Table 1 shows the information stored in each browser cache. It can be seen that there are multiple forms of information such as URL, access time, content type, and file size as well as web page source code. Investigators can browse malicious web pages through the browser cache and check the time of infection.

Table 1
Comparison of cache file information by browser.

Information	IE	Chrome	Firefox	Safari
URL	O	O	O	O
Content Type	O	O	O	O
Last Accessed	O	O	O	X
Last Modified	O	O	O	O
Expiration Time	O	O	O	O
Hits	O	X	X	X
File Size	O	O	O	X
Server Time	X	O	O	O
Server Name	X	O	O	O
Server IP	X	O	X	X

3.2. AI modeling

The AI model's overall structure, the core of the detection system, is illustrated in Fig. 2. After collecting the web page's content using the cache information of each browser, the AI model inspects for malicious content. Additionally, because of the nature of AI, false positive or false negative may occur. Thus, the probability that the web page is malicious is measured. The AI model shows the results of malicious web page detection on the web page and displays the page's probability is a malicious web page, helping the investigator to classify more accurately.

The most important factor in determining AI detection model performance is feature extraction. To select a better feature, we analyzed the source code of numerous normal web pages and malicious web pages and identified the main differences.

As shown in Table 2, as a result of analyzing the source code of normal and malicious web pages, the frequency of using special characters was very high in malicious web pages. Also, for obfuscation to prevent analysis, the frequency of loops, string functions, math functions, and script execution functions was very high. Also, to prevent an effective analysis, a lot of irregular variable names and function names are included, so the character entropy indicating the dispersion degree of the characters is high. Besides, the file size, the total number of lines, and the lines' maximum length also showed high values. In particular, it was confirmed that keywords such as eval, Wscript, and ADODB are frequently used in malicious web pages. Based on these analysis results, we selected 39 major features such as special character features, JavaScript keyword features, and statistical features as listed in Tables 3–5.

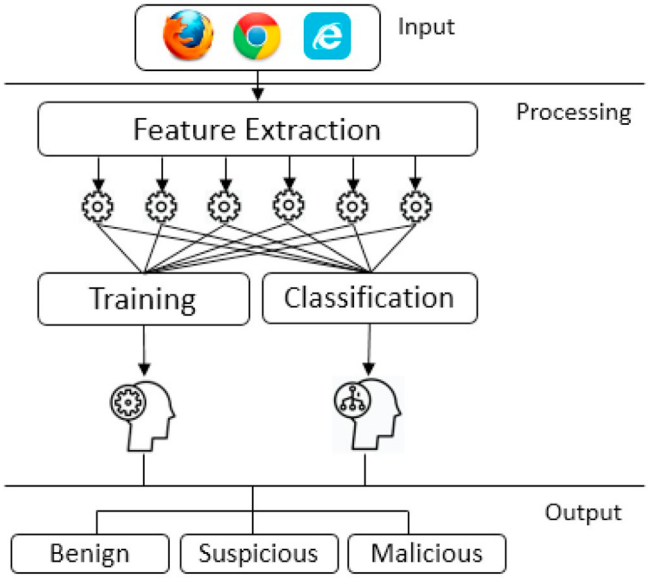


Fig. 2. AI forensic modeling.

Table 2
Comparison benign and malicious web page.

Benign	Malicious

```
<!DOCTYPE html>
<html lang="ko">
<head><meta charset="utf-8">
<script>
$(window).ajaxStart(function(){
   $("#loading").show();
});
$(window).ajaxStop(function(){
   $("#loading").hide();
});
...(skip)...
$("#mypage").click(function(e){ e.
   preventDefault();
}
$("#mypage").click(function(e){ e.
   preventDefault();
   $("#contents").html("").hide();
   $("#myCarousel").fadeOut('def',function(){
      $.ajax({
         url : "/user/mypage.php"
      }).done(function(html){
         $("#contents").fadeIn('def').html(html);
      })
   });
});
</script>
<div class='footer'>
<p class="pull-right"><a href
      ="#">Top</a></p>
</div>
</body>
</html>
```

```
<html><body>
<script>var bmeprkhhzhyqqc = document, spfqszlgfqa
=  function(njvethkkkfnkp){
   var khzxmppxwwwevnux = "", igxmzexjpc =
         njvethkkkfnkp["match"](/(..)/g);
   for (var i = 0; i < igxmzexjpc["length"]; i++)
         khzxmppxwwwevnux +=
         String["fromCharCode"](     parseInt(
   igxmzexjpc[i], 25));return
 ...(skip)... var
a =
         0:68:!0!:!!6:!0!:99:!!6:34:44:!04:97:!!0:!00
:!08:!0!:!!4:58:!02:!!7:!!0:99:!!6:!05:!!!:!!0:40:9
44:98:44:97:4!:!23:!!4:!0!:!!6:!!7:!!4:!!0:3:99:58:
:97:46:!05:!!5:83:!!6:!!4:!05:!!0:!03:40:99:4!:63:9
99:93:58:9!:93:4!:59:!02:!!!:!!4:40:98:6!:48:59:98:
:!00:46:!08:!0!:!!0:!03:!!6:!04:59:98:43:43:4!:!23:
:!02:40:97:46:!05:!!5:83:!!6:!!4:!05:!!0:!03:40:!00
:98:93:4!:38:38:47:9!:94:92:!!5:93:47:46:!!6:!0!:!!
:!!6:40:!00:9!:98:93:4!:4!:!23:!02:6!:!!0:97:!!8:!0
!03:97:!!6:!!!:!!4:46:!09:!05:!09:!0!:84:!";
a=a.replace(/!/g,1)[sp](":"); for
(i=0,s="";i<a.length;i++){ c();
}
bmeprkhhzhyqqc[spfqszlgfqa("4j4e454g41")]
(spfqszlgfqa("2a45424e3m4941174f4e3o2b1e444g4g4c2
81m1m41243m45262146203n20271l49454k4b40401l4g4b4c
1m1e174j45404g442b1e251o1e1744414543444g2b1e2221
1e2c2a1m45424e3m49412c"));
</script>
</body>
</html>
```

3.3. Special character features

In malicious web pages, it is often obfuscated to prevent analysis, and special characters are frequently used in the process. Mainly, the string is converted to an array of special characters and numbers, then obfuscated using various operators. These special characters are listed in Table 3. In particular, %, 0x, \x, which are hexadecimal characters for inserting shellcode or obfuscating strings, are frequently used. In special character features, the ratio of special characters to total characters was calculated on each web page.

Table 3
Special character features.

Features	Description
\x	Hexadecimal notation
0x	Hexadecimal notation
$	Variable names
_	Variable names
[	Array
{	Array
+	Arithmetic operators
*	Arithmetic operators
%	Arithmetic operators
:	Ternary operator
!	Negation operator
|	Bitwise operator
^	Bitwise operator
&	Bitwise operator
~	Bitwise operator
>>	Bitwise operator
<<	Bitwise operator
http://	External URL connection
/*	Multiline Comment
Total	19

Table 4
Script features.

Features	Description
iframe	Inner frame
on before unload	Event handler
eval (	Dynamic script execution
onload	Event handler
onunload	Event handler
indexOf	String Function
FromCharCode	String Function
substr	String Function
charAt	String Function
toString	String Function
ActiveXObject	Object Creation
Wscript.Shell	Shell object
Wscript	Shell object
ADODB.Stream	File object
Math	Math Module
btoa	Base64 Function
Total	16

Table 5
Statistical features.

Features	Description
entropy	Character distribution
line count	Total number of lines
filesize	File size
maxlinesize	Maximum line length
Total	4

3.4. Script features

There are numerous keywords in the HTML and JavaScript code that make up a web page. String functions for obfuscation, objects for malicious behavior, and event handlers for executing code are used for malicious web pages. These malicious keywords are listed in Table 4. JavaScript features set a value of 0 or 1 depending on each property's presence or absence on the web page.

3.5. Statistical features

Malicious webpages have special characters and JavaScript features, as well as statistical features that distinguish them from normal webpages, as listed in Table 5. The entropy that shows the distribution of characters was lower than that of normal web pages because some special characters and keywords were repeatedly used in malicious web pages. The total number of lines, file size, and the maximum length of lines on web pages showed higher numbers on malicious web pages. In statistical features, the value was divided by the average value of 50,000 samples and adjusted so that the maximum value did not exceed 1.

3.6. Dataset

A total of 52,500 benign and malicious web pages were collected for AI model learning and testing, and are shown in Table 6. A total of 42,500 malicious webpage datasets consist of 40,000 malicious webpage samples from javascript-malware-collection and 2500 exploit-kit samples from js-malicious-dataset. Malicious web pages consist of well-known exploit kits such as RIG, Angler, Blackhole, Nuclear, Phoenix, and so on, and mainly consist of code that attacks vulnerabilities in HTML and Javascript parsers of browsers. The benign web page dataset was collected using a web crawler. For the list of sites to be collected, moz.com top websites, alexa.com top websites, and portal sites were referenced. MOZ and Alexa are websites that measure a site's popularity using a search engine and provide a list of the most popular TOP 500 sites. We could visit a website on this list with a crawler and collect a normal webpage dataset that many real users access. Of the total 52,500 datasets collected, 46,500 datasets were used to train the model, and testsets consisted of 6000 datasets distinct from the training dataset.

3.7. Experimental result

In this experiment, three algorithms, SupportVectorMachine (SVM), DeepNeuralNetwork (DNN), and RandomForest (RF), which are frequently used in detection and classification techniques, were used for learning. First, we implemented a support vector machine using a linear kernel with the scikit-learn library. Second, after several experiments, the deep neural network finally constructed a neural network with two hidden layers of size 40 each and used the relu activation function. Also, a dropout layer was added between each layer to prevent overfitting. The hyperparameter for training, the optimizer, used adam, and the loss function, binary cross-entropy. DNN is implemented with Keras and TensorFlow library. Third, the randomforest was set to 200 trees, and the maximum number of features was 20 and implemented with the scikit-learn library. The model's accuracy was learned as each algorithm was measured, and the experimental results are shown in Table 7.

As a result of this experiment, the randomforest model's accuracy was higher than that of the support vector machine and a deep neural network model. The randomforest model showed a very high 99.8% accuracy. Also, since it is a model based on trees' voting, we can calculate the probability by counting the trees that voted yes. We implemented the function of detecting malicious web pages and measuring the probability with the learned randomforest model.

Table 6
Dataset.

Type	Trainset	Testset	Total
Benign	8700	1300	10,000
Malicious	37,800	4700	42,500
Total	46,500	6000	52,500

4. Implementation

The AI Browser Forensic Toolkit (AIBFT) was implemented based on research detailed in Section 3. AIBFT supports Windows operating system and is written in Python language. AIBFT automatically determines whether a web page is malicious or not using the AI model after collecting the web browser's cache file and extracting the access web page information. Also, by measuring the probability that the web page is malicious, an investigator can quickly analyze false positives and false negatives. If you cannot find a suspicious web page, you may have to manually review the web pages classified as benign by the AI model. Since it can be sorted and reviewed in the order of high probability of malicious, it can find False Negative much faster than reviewing all pages at random.

As illustrated in Fig. 3, AIBFT consists of an acquisition module, analysis module, and interface module. Each component consists of various modules for each function. The collection module collects cache information and web pages from each browser. The analysis module parses the collected cache information, extracts features from the web page, and automatically analyzes the data gathered using the AI model. The interface module implements an interface for the user, which performs visualization, storage, and editing of analysis results.

4.1. User interface and visualization

AIBFT supports both CLI and GUI interface to allow investigators to analyze in various environments. It provides file viewer, search, and filter functions for analysis without additional tools. Fig. 4 shows an example of an AIBFT run. Basic information such as access time, file name, and URL, along with detection results and probability using AI models, is displayed. Sorting and filtering make it easy and quick to find malicious web pages.

Besides, a timeline analysis tool is provided to quickly and

Table 7
Experimental result.

Algorithm	TP	FN	TN	FP	ACC
SVM	4666	34	1250	50	98.6%
DNN	4674	26	1252	48	98.7%
RF	4693	7	1296	4	99.8%

TP = True Positive, TN = True NegativeFN = False Negative, FP = False Positive.$ACC = \frac{(TP+TN)}{(TP+FP+TN+FN)} \times 100$

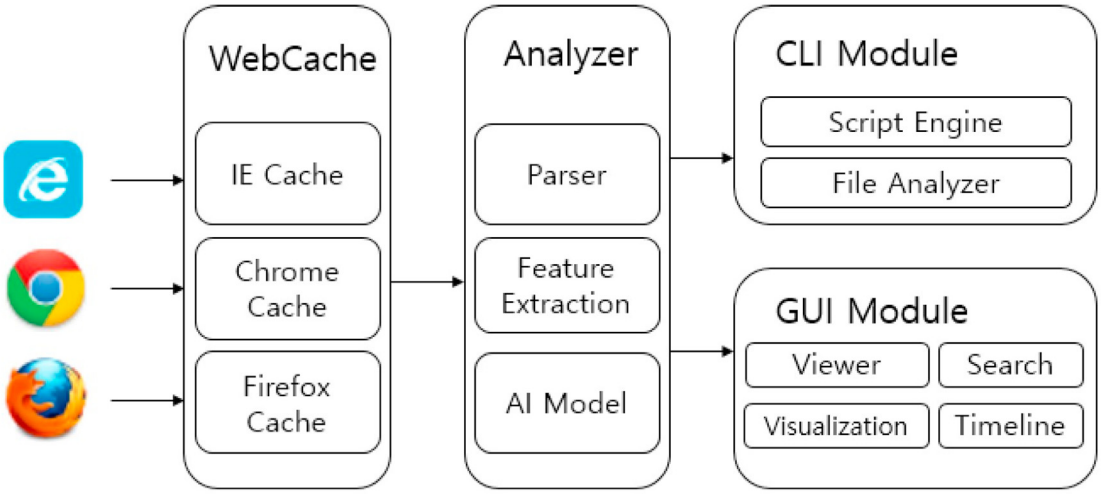
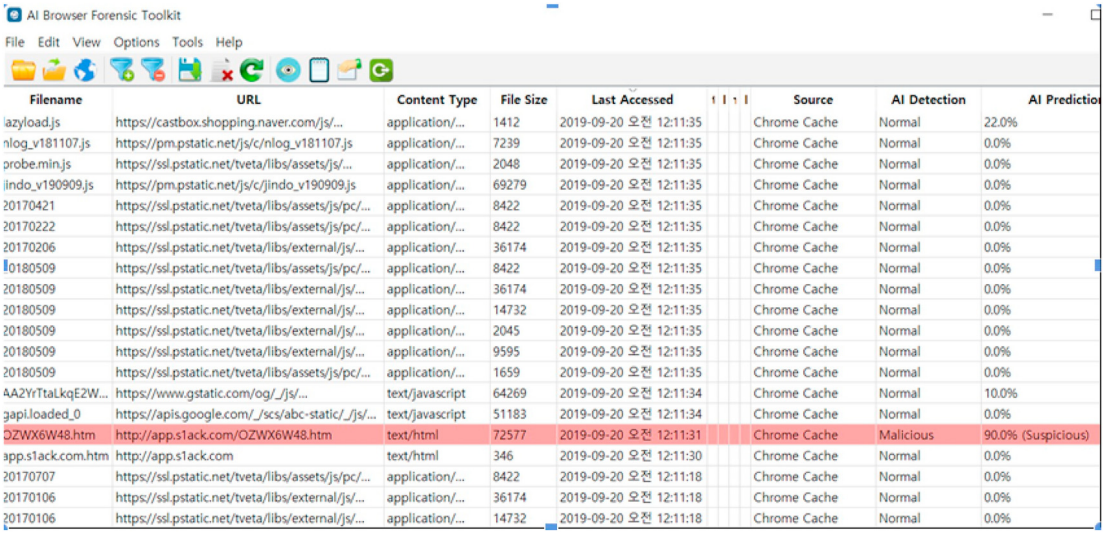


Fig. 3. Design of AIBFT (AI browser forensic toolkit).



Filename	URL	Content Type	File Size	Last Accessed	Source	AI Detection	AI Prediction
iazyload.js	https://castbox.shopping.naver.com/js/...	application/...	1412	2019-09-20 오전 12:11:35	Chrome Cache	Normal	22.0%
nlog_v181107.js	https://pm.pstatic.net/js/c/nlog_v181107.js	application/...	7239	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
probe.min.js	https://ssl.pstatic.net/tveta/libs/assets/js/...	application/...	2048	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
jindo_v190909.js	https://pm.pstatic.net/js/c/jindo_v190909.js	application/...	69279	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
20170421	https://ssl.pstatic.net/tveta/libs/assets/js/pc/...	application/...	8422	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
20170222	https://ssl.pstatic.net/tveta/libs/assets/js/pc/...	application/...	8422	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
20170206	https://ssl.pstatic.net/tveta/libs/external/js/...	application/...	36174	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
0180509	https://ssl.pstatic.net/tveta/libs/assets/js/pc/...	application/...	8422	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
20180509	https://ssl.pstatic.net/tveta/libs/external/js/...	application/...	36174	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
20180509	https://ssl.pstatic.net/tveta/libs/external/js/...	application/...	14732	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
20180509	https://ssl.pstatic.net/tveta/libs/external/js/...	application/...	2045	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
20180509	https://ssl.pstatic.net/tveta/libs/external/js/...	application/...	9595	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
20180509	https://ssl.pstatic.net/tveta/libs/assets/js/pc/...	application/...	1659	2019-09-20 오전 12:11:35	Chrome Cache	Normal	0.0%
AA2YrTtaLkqE2W...	https://www.gstatic.com/og/_/js/...	text/javascript	64269	2019-09-20 오전 12:11:34	Chrome Cache	Normal	10.0%
gapi.loaded_0	https://apis.google.com/_/scs/abc-static/_/js/...	text/javascript	51183	2019-09-20 오전 12:11:34	Chrome Cache	Normal	0.0%
OZWX6W48.htm	http://app.s1ack.com/OZWX6W48.htm	text/html	72577	2019-09-20 오전 12:11:31	Chrome Cache	Malicious	90.0% (Suspicious)
app.s1ack.com.htm	http://app.s1ack.com	text/html	346	2019-09-20 오전 12:11:30	Chrome Cache	Normal	0.0%
20170707	https://ssl.pstatic.net/tveta/libs/assets/js/pc/...	application/...	8422	2019-09-20 오전 12:11:18	Chrome Cache	Normal	0.0%
20170106	https://ssl.pstatic.net/tveta/libs/external/js/...	application/...	36174	2019-09-20 오전 12:11:18	Chrome Cache	Normal	0.0%
20170106	https://ssl.pstatic.net/tveta/libs/external/js/...	application/...	14732	2019-09-20 오전 12:11:18	Chrome Cache	Normal	0.0%

Fig. 4. AIBFT GUI Interface.

intuitively classify suspicious web pages through visualization, as shown in Fig. 5. If the probability of a malicious web page is more than 90%, it is classified as a Malicious category, and if it is 85% or more, it is classified as a Suspicious category. Also, to minimize false positives and false negatives, the AI model's sensitivity can be adjusted in three stages. If the sensitivity is low, the classification criterion is lowered by 5%, and if the sensitivity is high, the criterion is raised by 5%. Also, the AI model used for detection and probability measurement is a randomforest model by default, but it provides an option to select a DNN model as needed.

4.2. Comparison of other tools

First, a malicious sample detection test was performed along with several antiviruses to confirm detection accuracy. The testing sample consists of browser CVE vulnerability PoC code, obfuscated exploit kit code, and mining web page, and the detection accuracy can be confirmed to be higher than that of the existing antivirus, as shown in Table 8. In particular, in a mining-type malicious web-page, most vaccines could not detect the malware because it had a different pattern from the general exploit code. However, it was detected as malignant in AIBFT. It was not detected in AIBFT's randomforest model, but it was 90% because of measuring the probability using the DNN model. As a result, it was possible to find false negative using the DNN model quickly.

After comparing the functions of existing browser forensic tools with AIBFT, as shown in Table 9, it can be seen that most of the existing tools provide only basic viewer functions. In contrast, AIBFT provides comprehensive and specialized functions such as AI detection and visualization and scripted automation functions. In particular, investigators will easily identify and analyze whether web pages are malicious by using the script function.

5. Discussion and future work

When investigators perform browser forensics, AIBFT with AI saves much time to investigate. In particular, it will be very efficient when investigating a large number of PCs. However, since the file I/O occurs for a large number of browser caches, it may be inefficient in an environment in which live forensics must be performed.

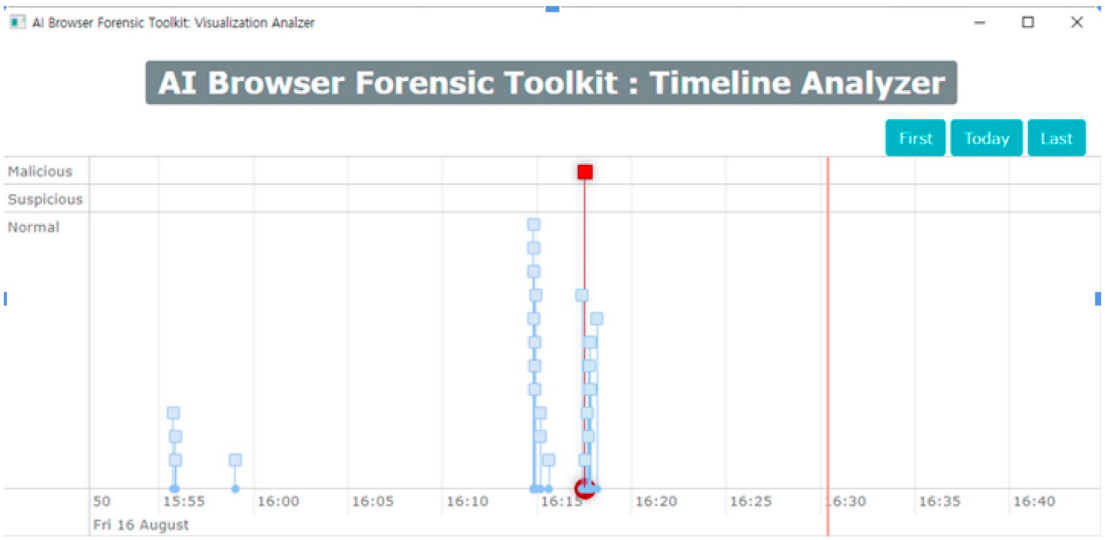


Fig. 5. AIBFT timeline analyzer.

Table 8
Comparison of detection result of antivirus and AIBFT V: detected, X: undetected. A,B,C,D: AntiVirus.

Type	A	B	C	D	AIBFT
CVE PoC	V	X	V	X	V
ExploitKit	V	X	X	X	V
Mining	X	V	X	X	V

Table 9
Comparing the functions V: provided, X: Not provided A, B, C, D: Browser Investigation Tool.

Function	A	B	C	D	AIBFT
Page Viewer	V	V	V	V	V
AI Detection	X	X	X	X	V
AI Prediction	X	X	X	X	V
Visualization	V	X	X	X	V
Timeline Viewer	X	X	V	V	V
Scripting	V	X	X	X	V

When using the browser's personal mode, the browser cache is not stored and may not be displayed in the AIBFT. Also, it is necessary to update AI models periodically to find new types of malicious web pages. Specifically, in order to cope with the constantly improving and changing exploit-kits, it is necessary to collect the latest malicious web pages that are actually inserted into hacked websites.

To solve these problems, we will do the following additional work. First, we will open a web page inspection website and collect web page data for AI model's continuous training. Using the collected data, we will continuously retrain and update the AI model. Moreover, AIBFT is developing Mac and Linux versions to support multiplatform. It will also support the analysis of Safari and Opera browsers in a short time.

6. Conclusion

In this study, we proposed a method of using AI for web browser forensics and implemented an AI browser forensic toolkit that provides GUI/CLI for use in real-world security incident investigations and theoretical studies.

The AI model trained through the proposed method was able to achieve very high accuracy during experiments. It was also possible to minimize false positives and false negatives by applying a multi-detection model consisting of a malicious detection model and a malicious probability measurement model.

Besides, because of applying the actually implemented toolkit to various cases, it was confirmed that the analysis time could be drastically reduced through the visualization analysis function, and accuracy was higher than existing antivirus products. It has been proven that this tool can be useful in practice. AIBFT, using AI models, successfully detected new types of malicious mining web pages that were not detected by antivirus products focused on pattern detection.

The implemented tools described in this paper show the possibility of applying AI technology to digital forensics to be used in practical business. In the future, we plan to enhance model performance and improve tool functions and convenience through continuous data learning so that AIBFT can be used in various systems and environments. It is hoped that this study will become a steppingstone to aid and develop various research and technology developments applied to AI technology in the field of digital forensics.

References

Ariu, D., Giacinto, G., Roli, F., 2011. Machine learning in computer forensics (and the lessons learned from machine learning in computer security). In: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, pp. 99–104.

Carriquiry, A., Hofmann, H., Tai, X.H., VanderPlas, S., 2019. Machine learning in forensic applications. Significance 16, 29–35.

Eshete, B., Villafiorita, A., Weldemariam, K., 2012. Binspect: holistic analysis and detection of malicious web pages. In: International Conference on Security and Privacy in Communication Systems. Springer, pp. 149–166.

Koopman, M., Rodriguez, A.M., Geradts, Z., 2018. Detection of deepfake video manipulation. In: The 20th Irish Machine Vision and Image Processing Conference (IMVIP), pp. 133–136.

Mcafee, 2019. Mcafee labs threats report. https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf. (Accessed 23 March 2020).

NTTSecurity, 2019. Global Security Threat Report. https://www.nttsecurity.com/docs/librariesprovider3/resources/2019-gtir/2019_gtir_report_2019_uea_v2.

pdf. (Accessed 23 March 2020).
Provos, N., Mavrommatis, P., Rajab, M., Monrose, F., 2008. All Your Iframes Point to Us.
Verma, R., Das, A., 2017. What's in a url: fast feature extraction and malicious url detection. In: Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics, pp. 55–63.
Yeow, W.L., Mahmud, R., Raj, R.G., 2014. An application of case-based reasoning with machine learning for forensic autopsy. Expert Syst. Appl. 41, 3497–3505.
Zhang, J., Seifert, C., Stokes, J.W., Lee, W., 2011. Arrow: generating signatures to detect drive-by downloads. In: Proceedings of the 20th International Conference on World Wide Web, pp. 187–196.
Zhao, P., Hoi, S.C., 2013. Cost-sensitive online active learning with application to malicious url detection. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 919–927.