

Automating the Generation of Enticing Text Content for High-Interaction Honeyfiles

Ben Whitham
University of New South Wales
ben@whitham.net.au

Abstract

While advanced defenders have successfully used honeyfiles to detect unauthorized intruders and insider threats for more than 30 years, the complexity associated with adaptively devising enticing content has limited their diffusion. This paper presents four new designs for automating the construction of honeyfile content. The new designs select a document from the target directory as a template and employ word transposition and substitution based on parts of speech tagging and n-grams collected from both the target directory and the surrounding file system. These designs were compared to previous methods using a new theory to quantitatively evaluate honeyfile enticement. The new designs were able to successfully mimic the content from the target directory, whilst minimizing the introduction of material from other sources. The designs may also hold potential to match many of the characteristics of nearby documents, whilst minimizing the replication of copyrighted or classified material from documents they are protecting.

1. Introduction

1.1. Honeyfiles

Honeyfiles [35], also referred to as honeytokens [26], digital decoys [15], decoy files [9], and canary files [31], is a cyber deception approach that has the potential to assist in the detection of data exfiltration and unauthorised access. Honeyfiles perform this role by emulating ‘real’ documents in order to lure and bait data thieves.

Honeyfiles have several advantages over honeypots and traditional security approaches. When correctly positioned and configured, honeyfiles have been proven to generate a negligible quantity of important alerts [6]. Ordinarily, honeyfiles should not be accessed, as their fake content provides zero benefit for legitimate users. Any user or process attempting to open, copy or delete a honeyfile provides a warning,

analogous to a canary in a coalmine. By concentrating on the alerts generated by honeyfiles, intrusion detection teams can reduce the volume of documents requiring constant observation.

Honeyfiles, like other forms of deception, can create uncertainty regarding the disposition of critical information [27] and increase the effort required to distinguish between real and fake data [30]. The data thief does not know where the traps are placed and risks detection each time they open a document [35].

Unlike honeypots, honeyfiles do not need dedicated hardware, nor expose additional software vulnerabilities to exploitation [31]. Honeyfiles can also be placed directly within document repositories, amongst the files that require protection, rather than isolated on different network hosts, collision domains, and/or network segments [28].





Type	Characteristics	Advantages	Example
 Low Interaction	Blocked or random data No words	Detection of simple automation of document harvesting	
 High Interaction	Words, sentences, paragraphs, images, tables and other realistic content	Bait for automated document extraction based on content searching Help to determine the why and who	

Figure 1. Low and High Interaction Honeyfiles

1.2. High-interaction Honeyfiles

Honeypot technologies have been previously classified as either low-interaction or high-interaction [19]. Low-interaction honeypots trade the depth of mimicry for lower risk of sensor compromise, implementation cost and management overhead [36]. High-interaction honeypots employ greater realism, which provides more potential to sustain a deception, allowing the system owner to gather additional intelligence on the source and intentions of the intruder/insider [19]. This comes at a greater cost of design and management overhead.

This paper proposes that honeyfiles can be similarly categorized (see Figure 1). Low-interaction honeyfiles simulate only the existence of a file. They may mimic basic file features, such as size and time stamps, but lack the complexity to sustain the illusion. Low-interaction honeyfiles are unlikely to contain words, sentences, video or any meaningful information. These low-interaction options may be useful to create initial confusion and detection of simple document harvesting programs.

High-interaction honeyfiles replicate the local environment with greater accuracy. One key advantage is that their content can entice external threat actors (who typically scan newly accessed file systems to identify valuable documents) [30][29]. High-interaction honeyfiles are also effective against malicious insiders masquerading as legitimate users; the attacker's lack of knowledge of the victim's access profile (files, locations of important directories, available applications, etc.) means that they are likely to engage in information gathering and search activities before initiating specific actions [6]. Both these threat actors could employ direct searches of the content or employ extant knowledge management systems that centralize or index the critical data [14] (see Figure 2).

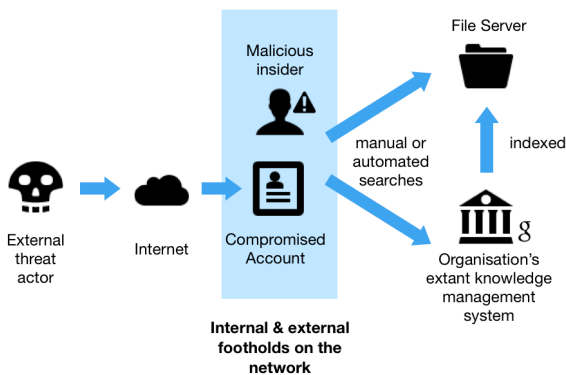


Figure 2. Threat actors target file system content

Like their honeypot namesakes, high-interaction honeyfiles are more suited to assist in determining the patterns and behaviors, objectives, tactics, techniques and procedures of the data thieves. Honeyfiles can be built with a range of properties, including the choice of content, format, complexity, quantity and location [30]. For instance, consider if honeyfiles that contain text associated with nuclear fusion are accessed, but those containing information relating to location of nuclear facilities are not. This intelligence may be valuable in understanding the data thief's intended target(s) and/or priorities, especially when fused with other traditional information sources. Employment of more complex content can also allow a system owner to detect subtle

and partial data exfiltration. For instance, signatures can be generated using individual honeyfile paragraphs to detect partial or slow theft of document data hidden amongst other seemingly benign email or web traffic.

Similar to honeypots, high and low interaction honeyfiles could also work in unison as a compromise among sensor coverage, detection and management cost. For instance, low-interaction honeyfiles could be deployed across the file system or operating system environments to detect automated intrusions. These could be paired with high-interaction honeyfiles in locations containing high value sets of documents, where the 'why' and 'how' are more important (and more cost effective).

2. Generating High-Interaction Honeyfiles

2.1. Honeyfile Diffusion

While high-interaction honeyfiles have been employed in computer security for more than 30 years their success in the hands of advanced defenders has not translated into mainstream use. Comparatively, the concepts of usage control, encryption and content tagging (watermarking), have evolved to commercial products. The lack of diffusion appears related to the difficulty of constructing deceptive content across the spectrum of potential data formats [33]. Rowe [24], a prolific author on cyber deception, stated that honeyfile content could never be automated. His position appears justified. When building fake data for 11 fixed database fields White and Thompson [30] noted, "It takes a lot of effort to produce realistic decoys... It took several months of research to locate all of the data that was needed to produce each field. This information also had to be processed into a usable form in order to incorporate it into the program. This is a very labor intensive task". Even a recent solution proposed by Wang et al [29] requires a new software module to be developed for each topic. This approach is not scalable [25] and is at odds with the 'ease of use' security product design principle [2].

2.2. Attempts to Generate Honeyfile Content

Most of the previous work on honeyfiles has focused on the placement and monitoring of baits, and has rarely paid attention to content or automatic creation of files [13]. Discussions on fake content have been centered on specific data types and formats, such as database fields [30][17], passwords and account details [8][21][16] and file attributes [24][35] [9][6][28][29].

There have been several attempts to automate the generation of fake content. This paper groups these current designs into four categories: Type I - IV.

Type I constructs produce low-interaction honeyfiles and simple deceptions that populate fake files with random data. Rowe [24] built a prototype fake file generator (NFDir) that is capable of populating documents with random characters. A similar approach has been adopted in free online resources. Fake File Generator and File Destructor 2.0 are tools to generate corrupted files so that students can gain more time to complete their assignments. Both software solutions create non-working files using random binary data and adopting common extensions.

Type II high-interaction honeyfile processes populate documents with content from an external source. NFDir is also able to generate files comprising of images, captions, and page names extracted from publicly accessible military web pages. Bowen et al [9] also considered using words that were frequently searched on the Internet as the constituent words in fake documents. The Impressions software program [1], while not used to make honeyfiles, creates content using word popularity models of the English language.

Type III approaches attempt to reduce the content generation problem for high-interaction honeyfiles to a manageable size by employing templates and fixed data fields. In 2005, Stribling and his colleagues developed an automatic computer-science paper generator, called SCIGen [3]. SCIGen constructed entire academic publications, including text, graphs, figures, and citations. The program relies on hand-written rules and a set template. Infamously, the authors used the software to generate a confusing and partially illegible paper, which was successfully accepted by the World Multi-conference on Systemics, Cybernetics and Informatics, albeit without peer review. White and Thompson [30] built fake content for an 11-field database containing personal records. Three years later, Bowen et al [9] produced the Decoy Document Distributor (D3) System, a tool for automatically generating and monitoring fake documents. The initial D3 prototype was limited to Microsoft Word and PDF documents from a set of fixed templates, including tax returns, medical records, credit card statements, and e-Bay receipts.

Type IV honeyfile methods comprise those designs that (1) use existing production files as bait, or (2) reproduce part or all of an existing document using another format, language or synonyms. Yuill et al [35] designed a system to allow a user to nominate a production file to become a honeyfile. Similarly, 'Honeydocs' is a free deception solution. Users provide Honeydocs with a file, and the program embeds a beacon to track its location on the Internet.

Park and Stolfo [22] built a system to generate fake Java programs. Their design obfuscates the original source code, creating compilable but syntactically dissimilar software. Shortly after, Voris et al [28] proposed a method of translating existing documents into a language foreign to the file system. They sequentially processed selected paragraphs through five different language translations.

2.3. Limitations of Current Solutions

Random data produced by Type I honeyfiles are likely to be unrealistic, foreign to the file system and directory, and appear suspicious. These types of processes do not typically generate sentences, words or other standard content that would be expected to reside within a document or match keywords searched by data thieves. These files merely present the illusion that the file contains, or once contained useful data. The only key advantages of these types of files are that they are simple to produce, and they will not contain classified, sensitive or copyright material.

Type II honeyfile content is likely to be more realistic and enticing than Type I, because it contains indexable words. Type II files, however, are unlikely to match the topics, word frequency and author stylometry. They may not necessarily contain sentences or paragraphs, or similar embedded artifacts.

There is no doubt that templates can be developed that will produce realistic and enticing text. The main challenge with Type III approaches is that templates and hand-written rules are neither scalable nor adaptable to the range of content that is likely to be encountered in a real world application.

While current approaches still largely rely on hand written rules, Type IV methods of construction hold potential to produce realistic and enticing decoys that match the local content that they are attempting to protect. The primary challenge with employing a production document as a honeyfile is that this process is likely to risk classified, sensitive or copyrighted material. Type IV processes typically retain the core meaning of the original text or functionality of the software. This limitation is consistent throughout the entire set of Type IV honeyfiles. For example, although content containment was tested to limit the retention of original code artifacts, Park and Stolfo's [22] process appeared to generate a functioning replica of the original software. It is unclear, therefore, if this modified code is just as valuable as the original intellectual property and should not be risked with deception activities. Similarly, the solution proposed by Voris et al [28] simply transforms sensitive material into another language. It is still likely to retain sensitive artifacts from the source documents thereby

creating a protection paradox. Finally, any approach that uses production documents that are no longer, relevant, critical or sensitive, by definition, will be less enticing to a data thief, and therefore unsatisfactory honeyfile candidates.

2.4. Type V - NLP-based Designs

Section 4 of this paper presents four new designs based on Natural Language Processing (NLP). NLP is a field of computer science, artificial intelligence, and linguistics concerned with automating the understanding and creation of human readable text. NLP may help to widen the approachability and convenience of honeyfiles. Like other fields, automation could (1) reduce the labor cost associated with the process, (2) improve their reliability and consistency, (3) speed up the generation process, and (4) allow the process to happen more frequently [23]. The main advantage of automation, however, would appear to lie in its ability to address complexity barriers, by allowing the honeyfile generation process to be undertaken by staff with less skill [23]. Automated solutions may also adjust and adapt to new content and scale to large and complex environments.

3. Assessment of NLP-based Designs

3.1. Scenario

The scenario developed for this research is that the system owner seeks to place a honeyfile within a directory in their digital file system. The role of the honeyfile is to act as an early warning for the theft of any or all of the files residing within the file repository, but in particular, those files within that file directory.

There are a number of assumptions and limitations associated with the scenario. The honeyfiles are protecting directories within a file system, rather than databases and other information management systems. While other information storage systems are becoming increasingly popular for big data, file systems remain a fundamental mechanism for storing digital information [20]. This research only considered English text content. Diagrams, figures and images were not included. Despite the number of multimedia documents rising, most of the interesting information in digital format is still numeric or textual [17]. The study assumes that all files contained within the directory are equally sensitive. It could be possible to encounter folders that contain a mixture of document sensitivities. In practice, however, most human users organize folders by common topics and authors [4].

This grouping provides a theme for each honeyfile to defend and mimic.

3.2. Test Data

Two test data sets were used. The first data set contained 1000 test directories populated with academic papers (in PDF format) harvested from the Internet, using a similar manner to [11]. The majority of papers were downloaded from SIGCOMM Conference web sites. The remaining papers were harvested using Google Scholar. All papers were manually inspected to confirm the contents were not scanned images or fake files. All of the harvested PDF documents were reduced to text files using the open source Apache Tika software [17]. Each directory was populated with a uniform distribution of between 2 and 13 documents, chosen at random, based on the observations of previous file system surveys [32]

This synthetic data set was deliberately chosen to align with an existing honeyfile template. This eliminated the requirement to create a new Type III honeyfile template to match the dataset.

The second dataset used a copy of a small business's production file system. The real-world data set evaluated the honeyfile design's ability to navigate duplications, data errors and expose the designs to realistic human organizational patterns. 1000 directories were selected at random. The files were also converted to text using Apache Tika.

3.3. Current Honeyfile Design Representatives

An exemplar design was chosen to represent each of the current honeyfile content generation approaches (Type I-IV). The chosen Type I exemplar process creates honeyfile content using a subset of ASCII characters, selected at random. The sub-set included all members of the upper and lower case English alphabet, numbers and the blank space character. Each of the characters in the sub-set was given an equal chance of selection.

The Type II exemplar process selected words at random from the Brown Corpus (a popular dataset used by the NLP community) [10], based on the underlying word frequency distribution.

Type III constructions were represented by the SCIGen [3] fake academic paper generation tool (previously discussed). The SCIGen PDF documents were converted to text using Apache Tika, matching the process used to create the test data sets.

A method similar to Voris, et al [28] was chosen to represent Type IV honeyfiles. In this example process, a file is chosen at random from the target directory and

then translated from English to Spanish, to French and then back to English. The selection of languages was made to support an offline conversion process.

3.4 Comparing Honeyfiles and Defining 'Enticing' Assessment Criteria

This paper compared the enticement produced by the individual honeyfile designs. Whitham [32] identified seven criteria for honeyfile content generation. Of these, the most essential requirement of a honeyfile content generation process is to spawn enticing material [6]. Bowen et al [9] associated a fake file's ability to entice with the desirability of the data thief to access and/or exfiltrate its contents. For instance, a fake file that suggested or contained a list of passwords might be more desirable than a document containing the monthly office social calendar.

There is currently no existing theory on defining or evaluating the enticement of a honeyfile. This paper presents an initial theory and evaluation method that matches the chosen scenario.

One of the principal aims of high-interaction honeyfiles is to present themselves as genuine targets to indexing technologies, baiting external threat actors and insiders who are searching for valuable information within the file system (see Figure 2). The ability of the honeyfile to entice these threat actors can be improved with selective attribute choices [6].

Creating honeyfiles that mimic this content achieves two primary goals. First and foremost, like angling, it is important for the lure to match the typical diet of the fish. Honeyfiles can only detect data theft if the fake file is accessed [12]. The more enticing the deceptive content, the more likely that an intruder will access the fake file and trigger the alarm. The scenario assumes that the content in the test directories is sufficiently valuable to warrant protection. Matching the material found within the local directory populates the honeyfile with (equally) desirable content.

Moreover, one of the advantages of cyber deception is the ability for the system owner to obtain information about the data thief, their motivations and targets. For instance, the system owner might gain a valuable insight into the goals of the perpetrator if a set of honeyfiles containing engineering and research content was accessed, but honeyfiles mimicking financial records were not. This knowledge is only possible if the honeyfile content is deliberately tailored to carry a subset of the critical data. The file directories, by their nature, are likely to be associated with a single topic, and provide a logical source of content for an individual honeyfile.

The second reason for matching the content of the honeyfile with the legitimate files in the directory is

associated with minimizing false positives. One of the limitations of honeyfiles is their potential to disrupt normal work activity and pollute genuine information. By limiting the content of the honeyfile to strictly those topics discussed in the target directory, the system owner can minimize the impact on legitimate workers. Consider a honeyfile that contained random content. To a corporate information management system, random content is likely to match a range of topics across the file system, and not just those topics discussed in the sensitive directory folders. These honeyfiles containing random content are more likely to be recommended when legitimate users employ content searching / indexing technologies to locate unrelated, less sensitive information in the file system. This will increase the number of unintended interactions with the honeyfile, creating additional work for the incident responders, and lowering their confidence in the deception system.

These two content enticement goals can be summarized as:

1. Maximize the number of matches with topics discussed in the files in the local directory.
2. Minimize the number of matches with topics that are not discussed in the local directory.

In its simplest form, the above two enticement goals can be described using standard set theory. Consider the given scenario where a high-interaction honeyfile is created to assist in the detection of suspicious activity relating to a directory of critical documents housed within a file system. Assuming that the critical file is text based, the unique topics (T) in the target directory can be represented as $D = \{T_1, \dots, T_m\}$. Given a honeyfile content generation function, $f(X)$, the unique topics (θ) contained in a honeyfile can be represented as $H = \{\theta_1, \dots, \theta_j\}$. Assuming that all of the topics discussed within the document are of equal sensitivity, the ultimate aim of developing enticing content, therefore, is for $f(X)$ to build a honeyfile that contains every topic in D , or $H \subseteq D$.

Partial enticement success can therefore be achieved if the honeyfile contains at least one of the topics discussed in the critical file (where $H \cap D$ is inhabited), that is: $\exists H \in D : f(X)$ - the lower the set difference between D and H ($D \setminus H$), the better. These goals are illustrated in Figure 3.

Assuming there is no ability to compress topics, the process of including all of the topics contained in the directory could, at worst, make the file size of the honeyfile equal to the sum of the file lengths of the legitimate documents in the directory:

It would appear that the most effective method of meeting this goal would be to build large honeyfiles that contain a replication of all of the topics in D . This action, however, is at odds with the requirement for

building realistic honeyfiles (the honeyfile is expected to have a similar file size to the legitimate files in the directory). That is, the honeyfile may only contain 1/n of the topics found in the directory.

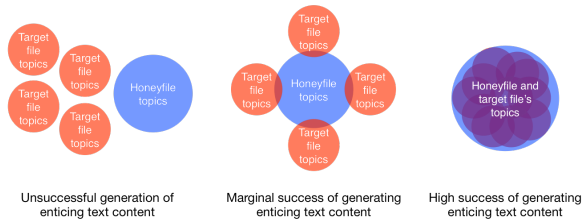


Figure 3. Maximize directory topic matches

Assuming that the legitimate files contain a similar number of topics, then it follows that the greater the number of files in the directory, the more difficult it would be to achieve this enticement goal (without sacrificing realism or compressing the topics).

The worst-case scenario is unlikely. Firstly, the above assumes that there is topic repetition within a document or within files in the target directory, which is unlikely. Secondly, one of the scenario assumptions is that files have been organized into a directory based on topic and/or author. It is possible, therefore, that there is a natural duplication and overlap of topics between files in the same directory. Thirdly, this also assumes that topic conversations are of equal length, which is unlikely to be the case in natural language [34]. Finally, the above assumes that all content in the files is relevant and required to communicate the topic and there is no redundant content, either through structure, partly completed work or through tautologies, verbosity and other discursive behavior.

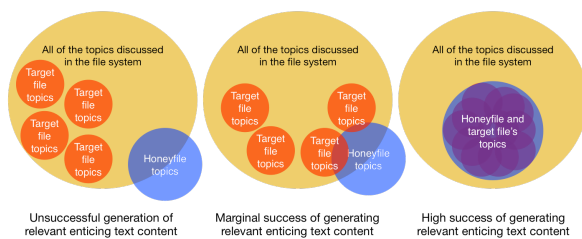


Figure 4. Minimize file system topic matches

While optimization may address these challenges, there remains the strong possibility that not all topics contained within the directory can be represented by a single honeyfile, particularly in directories with a large number of documents. Prioritizing the selection of topics is therefore critical to the content generation of high-interaction honeyfiles. The ability of the honeyfile generation process to successfully recognize and

prioritize these topics forms the basis of the first enticement assessment criteria.

For the second enticement goal, the set of topics that are included in the honeyfile, but are not discussed in the critical file can be identified by the set difference between H and D. The closer H\D tends to the empty set (\emptyset), the better. This includes the topics discussed in H that are also present in elsewhere in the file system, and those topics in H that are introduced from external sources. This concept is illustrated in Figure 4, where the smaller the percentage of blue area, the better, regardless of the intersections with the yellow set (file system).

4. New NLP-based Designs

This paper presents four new NLP-based designs. These designs are based on the substitution and transposition of words collected from the target directory and file system. Each of the designs employed the same three stages: (1) template selection, (2) content extraction, and (3) document population (see Figure 5).

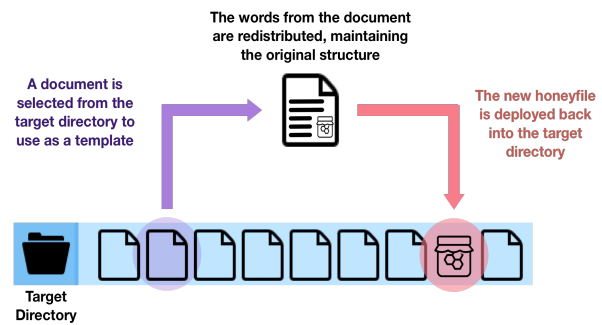


Figure 5. The new designs employ a 3 stage process

In the first step, the median sized file is selected from the target directory to use as a template for the honeyfile. In the second step, the words are stripped from the template and replaced with parts of speech XML-like tags (see Figure 6). The original punctuation, numbers and symbols are retained to ensure that the sentence lengths and structure of the document are mimicked.

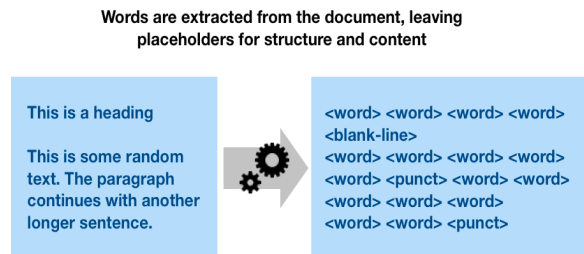


Figure 6. Text is replaced by POS mark-ups

Finally, the reduced template is used as a guide to build the honeyfile. Once the honeyfile is constructed, the signatures relating to the file and its content are forwarded to the extant intrusion detection systems for monitoring, and the honeyfile is deployed into the target directory.

One of the more complex steps involves converting the selected document into Parts Of Speech (POS) XML-like tags. The designs employ the popular Natural Language Tool Kit (NLTK) to classify the words into 35 POS tags. Figure 7 provides an example of a section of words prior to and after the conversion. Note that the case of the original word is retained in the POS marking.

Keywords: Deception, honeypot, honeynet, optimization.

1 Introduction

Recent trends indicate that enterprises today face a growing threat of sophisticated attackers who seek to steal or compromise proprietary information and assets [19, 11, 2, 1]. These attacks are executed in multiple stages and each stage is highly customized for each targeted enterprise. Attackers typically leave few network and system level footprints in attempts to evade detection. They exploit zero-day vulnerabilities to deliver malware using a single event, such as a carefully crafted spear-phishing email, as the entry point into an enterprise network. Detecting such events by leveraging existing techniques is difficult, especially when social engineering techniques are used to infiltrate the target organization. Once an attacker has infiltrated an organization, detecting other phases of an attack, such as data exfiltration, is a very difficult process which requires correlation between multiple events, such as firewall, IDS and DNS logs. As attackers customize their attacks for individual targets, prior knowledge such as blacklists of malicious domain names, drop servers IPs and malware signatures may not be useful for the attack. Therefore, rapid detection is technically difficult and stopping an attack in the early stage is challenging.

<Nnp>: <Nnp>, <nn>, <nn>, <nn>.

1 <Nnp>

<Nnp> <vbz> <vbpp> <in> <nns> <nn> <vbpp> a <vbg> <nn> <in> <nn> <ij> <nns> <wpp> <vbpp> <to> <vb> <cc> <vb> <ij> <nn> <cc> <nns> [19, 11, 2, 1]. <Dt> <nns> <vbpp> <vbns> <in> <ij> <nns> <cc> <dt> <nn> <vbz> <rb> <vbns> <in> <dt> <ij> <nn>. <Nnp> <rb> <vbpp> <ij> <nn> <cc> <nn> <nn> <nns> <in> <nns> <to> <vb> <nn>. <Prp> <vbpp> <cd> <nn> <nns> <to> <vb> <nn> <vbg> a <ij> <nn>. <ij> <in> a <rb> <vbns> <ij> <vbz> <nn>. <in> <dt> <nn> <nn> <in> <dt> <nn> <nn>. <Nnp> <ij> <nns> <in> <vbz> <vbg> <nns> <vbz> <ij> <nn>. <rb> <wrbs> <ij> <nn> <nns> <vbpp> <vbns> <to> <in> <ij> <nn> <dt> <nn> <nn>. <in> <dt> <nn> <vbz> <in> <vbns> <dt> <nn>. <vbg> <ij> <nns> <in> <dt> <nn>. <ij> <in> <nns> <fw> <nn>. <vbz> a <rb> <ij> <nn> <nn> <wdb> <vbz> <nn> <in> <ij> <nns>. <ij> <in> <nn>. <NNP> <cc> <NNP> <nns>. <ln> <nns> <vbpp> <prp\$> <nns> <in> <ij> <nns>. <rb> <vbpp> <ij> <in> <nns> <in> <ij> <nn> <rb>. <ij> <nns> <Nnp> <cc> <ij> <nns> <md> <rb> <vb> <ij> <in> <dt> <nn>. <Nnp>. <ij> <nn> <vbz> <rb> <ij> <nn> <cc> <vbg> <dt> <nn> <in> <dt> <ij> <nn> <vbz> <ij>.

Figure 7. Text from the template is replaced with POS mark up language, retaining the case

4.1. Type Va

In Design Va all of the words from the target directory are sorted into POS ‘word buckets’. The frequency of the words is retained. Design Va replaces each of the POS tags in the honeyfile template with a random word drawn from one of the ‘word buckets’ collected from the target directory. Once a word is selected from a ‘word bucket’ it is no longer available for future selection.

4.2. Type Vb

Design Vb is an extension of Design 1 except that the honeyfile population process employs conditional frequency to select a word from a ‘word bucket’. Conditional frequency is a form of Markov process

that takes advantage of the fact that sentence construction is not random. The process assumes that the future behaviour of the system only depends on its recent history [5]. Conditional frequency has previously been applied to predictive text and spelling corrections [18], where models guess the next word based on the relative frequency and combination of previous words. In these cases, the conditional frequency model employs n-grams of prior context to make the selection on the next word. In order to build the cumulative frequency models, the target directory text was organised into 2, 3 4 and 5 word n-grams. A ‘word bucket’ was created for each n-gram collection.

During the honeyfile population step, Design Vb attempts to select a word that matches an n-gram from one of its ‘word buckets’ and the honeyfile template’s POS tag. Design Vb starts this process by attempting to identify word options that could match known quingrams. If no matching quingrams can be identified, the process tries quadgrams. This continues until either a bigram match is found or a word is chosen at random from that particular POS ‘word bucket’.

4.3. Type Vc

Design Vc is an extension of Design Vb. In addition to the POS tagging and cumulative frequency model selection process, design Vc also collects bigrams from across the entire file system as a final option when a Markov Chain cannot recommend an option from the target directory text alone.

4.4. Type Vd

Design Vd is an extension of Vc, but it captures all of the 2, 3, 4 and 5 word n-grams across the target file system and uses a logarithmic scaled Term Frequency - Inverse Document Frequency (TF-IDF) to prioritize the n-grams captured from the directory over those in the corpus. TF-IDF is a popular product of two statistical algorithms that have been commonly applied in tandem to weight the value of documents in information search processes.

A random weighting is applied to each option at the time of word selection in order to ensure that: (1) the most common term, for instance “in to the”, is not continually selected; and (2) that a new honeyfile is created each time the process runs. The weighting value was randomly selected from values between the lowest score and the highest score.

5. Results

5.1. Maximizing Matches - Directory Content

Figure 8 presents the raw distribution of results from processing the data set of 1000 directories containing academic-paper extracts. Each value represents the percentage of words (topics) that appeared in both the honeyfile and the target directory. The higher the score the better.

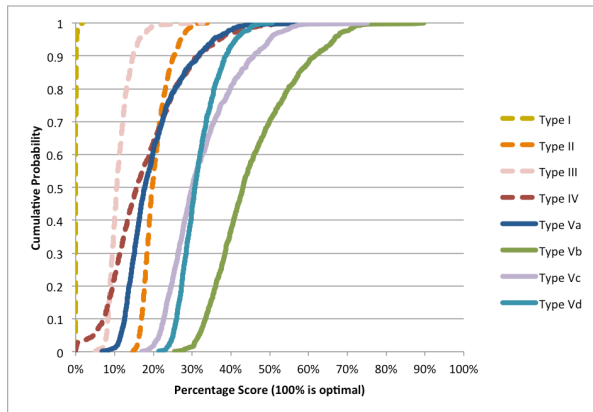


Figure 8. Probability Distribution of Maximizing Content from Target Directory

By observation, designs Vb, Vc and Vd outperformed all of the previous approaches, while Va scored similar results to the highest performing current approach. As expected, the random data from Type I constructions bear virtually no similarity to the file content in the directories that they are attempting to mimic. Type II and III have a small level of similarity, as expected through sharing the natural properties of English language. Type IV does not perform as well as expected. This is most likely due to the fact that the process mimics one file well, but not an entire file directory.

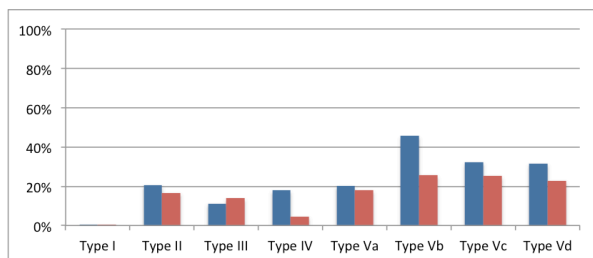


Figure 9. Maximizing Content - Target Directory

Figure 9 presents the mean scores from each of the academic and small business data sets. Contrast is limited, except for Type IV. Additional investigation is required, but it is likely that these constructions were affected by a combination of the diversity of files contained in the real-world file directories (e.g. partly finished documents and different content types), and the potential for random choice to make relatively poor

selections of files within these directories to use as a representative honeyfile template.

A one-sided p-value test was conducted on the four proposed designs against the current designs with a significance of 5% to test the hypothesis that the new Type V designs were a measurable improvement on the current approaches. The Va comparison against Type II constructions (using the academic data set) was the only assessment that scored greater than 0.000%.

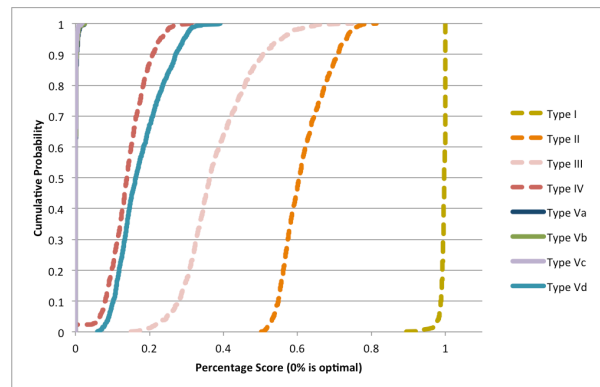


Figure 10. Probability Distribution of Minimizing Content from Target File System

5.2. Minimizing Matches – File System Content

Figure 10 presents the raw distribution of results from processing the data set of 1000 directories containing academic-paper extracts. Each value represents the percentage of words (topics) that appeared in both the honeyfile and the entire target file system but not the target directory. The lower the score the better.

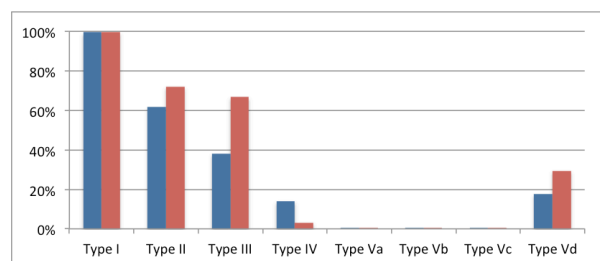


Figure 11. Minimizing Content - Target File System

Va, Vb, and Vd all produced near perfect scores, out-performing all of the previous approaches. Vd scored similar results to the highest performing current approach (Type IV). Type I could not be properly assessed as the text only produced ‘words’ by chance. Type II and III contain large traces of content that does not appear in the target directory, but appears across the file system. The translation processes involved in Type IV constructs also seem to introduce a small

number of words that do not appear in the target directory, but appear elsewhere in the file system.

Figure 11 presents the mean scores from each of the academic and small business data sets. Once again, the different data sets do not provide much contrast.

A one-sided p-value test was conducted on the four designs against the current designs with a significance of 5% to test the hypothesis that the new Type V designs were an improvement on the current approaches. The Vd comparison against Type IV (both data sets) was the only assessment that failed the hypothesis evaluation.

6. Conclusions

Advanced defenders have successfully used honeyfiles to detect unauthorized intruders and insider threats for more than 30 years. Unfortunately, the complexity of the content generation process has prevented their wider diffusion.

This paper presents an NLP-based content generation design for honeyfiles that select a document from a target directory and employ transposition and substitution of text using parts of speech tagging and n-grams harvested from the target directory and surrounding file system.

The level of enticement of these new designs was compared to previous methods of high and low interaction honeyfile generation. Two sets of 1000 test directories containing text content were used to evaluate the honeyfiles. A honeyfile from each design was built to simulate the process of creating bait files for each of the directories.

There is no previous published work on the quantitative evaluation of honeyfile enticement. This paper introduced a new approach, based on set theory, using topic (word) comparisons. This study also found that NLP-based designs could produce English text content for honeyfiles that match file directories. This process appears independent to the target content.

While the four new designs may not produce completely legible text, there is potential for the constructions to contain sufficient traces of enticing material to deceive an automated search process or malicious insider into accessing the document and triggering an alarm. These designs may also match many of the characteristics of the nearby documents, including the sentence and paragraph lengths and complexities, as well as the underlying format. The use of randomisation in text selection may also minimise the replication of classified, sensitive or copyrighted material and reduce the chance that the honeyfile creates a protection paradox by duplicating sections of

sensitive sections of text from the file system (that also need protection).

7. Future Research

Further research is underway to improve the realism of these new honeyfile designs, while minimising the retention of sensitive content. Future research could also consider alternative options for building honeyfiles and testing their enticement.

10. References

- [1] Agrawal N, Arpaci-Dusseau A C and Arpaci-Dusseau R H (2009). Generating realistic impressions for file-system benchmarking, *ACM Transactions on Storage (TOS)* 5(4), 16.
- [2] Axelsson S (2000). The base-rate fallacy and the difficulty of intrusion detection, *ACM Transactions on Information and System Security (TISSEC)* 3(3), 186–205.
- [3] Ball P (2005). Computer conference welcomes gobbledegook paper, *Nature* 434(7036), 946–946.
- [4] Barreau D and Nardi B A (1995). Finding and reminding: file organization from the desktop, *ACM SigChi Bulletin* 27(3), 39–43.
- [5] Baum L E, Petrie T, Soules G and Weiss N (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, *The annals of mathematical statistics* pp. 164–171.
- [6] Ben Salem M and Stolfo S (2011). Combining a baiting and a user search profiling techniques for masquerade detection. *JoWUA*, 3(1/2), 13-29.
- [7] Bercovitch M, Renford M, Hasson L, Shabtai A, Rokach L and Elovici Y (2011). HoneyGen: An automated honeytokens generator, in *Intelligence and Security Informatics (ISI)*, 2011 IEEE International Conference on, IEEE, pp. 131–136.
- [8] Bojinov, H., Bursztein, E., Boyen, X., & Boneh, D. (2010, September). Kamouflage: Loss-resistant password management. In *European Symposium on Research in Computer Security* (pp. 286-302). Springer Berlin Heidelberg.
- [9] Bowen B, Hershkop S, Keromytis A and Stolfo S (2009). Baiting Inside Attackers Using Decoy Documents. In *International Conference on Security and Privacy in Communication Systems* (pp. 51-70). Springer Berlin Heidelberg.
- [10] Francis W N and Kucera H (1979). *Brown corpus manual*, Brown University.

- [11] Garfinkel S, Farrell P, Roussev V and Dinolt G (2009). Bringing science to digital forensics with standardized forensic corpora, *digital investigation* 6, S2–S11.
- [12] Joshi R and Sardana A (2011). *Honeypots: A New Paradigm to Information Security*, Science Publishers.
- [13] Kaghazgaran P and Takabi H (2015). Toward an Insider Threat Detection Framework Using Honey Permissions, *Journal of Internet Services and Information Security (JISIS)* 5(3), 19–36.
- [14] Kramer L A and Heuer Jr R J (2007). America's Increased Vulnerability to Insider Espionage, *International Journal of Intelligence and CounterIntelligence* 20(1), 50–64.
- [15] Kushner D (2003). Digital decoys [fake MP3 song files to deter music pirating], *Spectrum*, IEEE 40(5), 27.
- [16] Liu B, Liu Z, Zhang J, Wei T and Zou W (2012). How many eyes are spying on your shared folders?, in *Proceedings of the 2012 ACM workshop on Privacy in the electronic society*, ACM, pp. 109–116.
- [17] Mattmann C and Zitting J (2011). *Tika in Action*, Manning Publications Co.
- [18] Mays E, Damerau F J and Mercer R L (1991). Context based spelling correction, *Information Processing & Management* 27(5), 517–522.
- [19] Mokube I and Adams M (2007). Honeypots: concepts, approaches, and challenges, in *Proceedings of the 45th annual southeast regional conference*, ACM, pp. 321–326.
- [20] Nguyen N T, Reiher P L and Kuenning G H (2003). Detecting Insider Threats by Monitoring System Call Activity., in *IAW*, Citeseer, pp. 45–52.
- [21] Nikiforakis N, Balduzzi M, Van Acker S, Joosen W and Balzarotti D (2011). Exposing the lack of privacy in file hosting services, in *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*, LEET, Vol. 11.
- [22] Park Y and Stolfo S J (2012). Software decoys for insider threat, in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ACM, pp. 93–94.
- [23] Pettichord B (1999). Seven steps to test automation success, STAR West, San Jose, NV, USA.
- [24] Rowe N C (2004). A model of deception during cyber-attacks on information systems, in *Multi-Agent Security and Survivability*, 2004 IEEE First Symposium on, IEEE, pp. 21–30.
- [25] Schilder F, Howald B and Kondadadi R (2013). Gennext: A consolidated domain adaptable NLG system, in *Proceedings of the 14th European Workshop on Natural Language Generation*, pp. 178–182.
- [26] Spitzner L (2003). Honeypots: Catching the insider threat, in *Computer Security Applications Conference*, 2003. Proceedings. 19th Annual, IEEE, pp. 170–179.
- [27] Tirenin W and Faatz D (1999). A Concept for Strategic Cyber Defense, in *Military Communications Conference Proceedings*, MILCOM, pp. 458–463.
- [28] Voris J A, Jermyn J, Keromytis A D and Stolfo S J (2013). Bait and Snitch: Defending Computer Systems with Decoys, *Cyber Infrastructure Protection Conference* .
- [29] Wang W, Bickford J, Murynets I, Subbaraman R, Forte A G and Singaraju G (2012). Catching the wily hacker: A multilayer deception system, in *Sarnoff Symposium (SARNOFF)*, 2012 35th IEEE, IEEE, pp. 1–6.
- [30] White J and Thompson D (2006). Using Synthetic Decoys to Digitally Watermark Personally-Identifying Data and to Promote Data Security., in *Security and Management*, pp. 91–99.
- [31] Whitham B (2013). Canary Files: Generating Fake Files to Detect Critical Data Loss From Complex Computer Networks, in *The Second International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec2013)*, The Society of Digital Information and Wireless Communication, pp. 170–179.
- [32] Whitham B (2014). Towards a Set of Metrics to Guide the Generation of Fake Computer File Systems, in *Proceedings of the 12th Australian Digital Forensics Conference*, Security Research Institute, Edith Cowan University, Perth, Western Australia.
- [33] Whitham B, Turner T and Brown L (2015). Automated Processes for Evaluating the Realism of High-Interaction Honeyfiles, in *Proceedings of the 14th European Conference on Cyber Warfare and Security 2015: ECCWS 2015*, Academic Conferences Limited, p. 307.
- [34] Wu Z and Palmer M (1994). Verbs semantics and lexical selection, in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, Association for Computational Linguistics, pp. 133–138.
- [35] Yuill J, Zappe M, Denning D and Feer F (2004). Honeyfiles: deceptive files for intrusion detection, in *Information Assurance Workshop*, 2004. Proceedings from the Fifth Annual IEEE SMC, IEEE, pp. 116–122.
- [36] Zhuge, J., Holz, T., Han, X., Song, C., & Zou, W. (2007, December). Collecting autonomous spreading malware using high-interaction honeypots. In *International Conference on Information and Communications Security* (pp. 438-451). Springer Berlin Heidelberg.