Automating the Generation of Fake Documents to Detect Network Intruders

Ben Whitham
University of New South Wales
Ben.whitham@student.adfa.edu.au

ABSTRACT

This paper introduces two concepts: Canary Files and a Canary File management system. A Canary File is a fake computer document that is placed amongst real documents in order to aid in the early detection of unauthorised data access, copying or modification. The Canary File acts as a hidden watermark for a file directory containing critical documents; the Canary File and its contents can be used as signatures to detect suspicious copying, access and deleting of files in the directory in preference to, or in conjunction with monitoring all of the file activity within the network. The name originates from canaries, which were used within coalmines as an early warning to miners.

This paper also introduces the Serinus System, a Canary File management system designed to address some of the key challenges associated with creating realistic mimicry across a large and complex computer network. The Serinus System automates Canary File generation using content and file statistics drawn from three sources: (1) Internet harvested documents, (2) documents collected from across the entire enterprise environment, and (3) documents within the specific target directory. Each data source is allocated a weighting based on the strength of their relationship to the target directory. The weighting is seeded with a random value to avoid discovery by simple statistical based fake file detection systems. Research is continuing to assess the performance of both Canary Files and the Serinus System.

This paper is an extension of a conference paper presented at The Second International Conference on Cyber Security, Cyber Warfare and Digital Forensic (Cyber Sec 2013).

KEYWORDS

Data Loss Prevention, data leak, mimicry, fake documents, honeypots, honeyfiles, Canary Files, deception, cyber security.

1 DETECTING DATA LOSS

Data loss is the release of private or sensitive data to an unauthorised entity. Data loss could be the result of accidental or intentional action(s) [1].

The recent publication by WikiLeaks of a large quantity of US government documents highlights the challenge of protecting sensitive data, even by seemingly well-resourced organisations from deliberate theft. Detecting the theft of data is made difficult by the requirements to: (1) continually share information between partners, customers, suppliers, and government; and (2) manage data external to the organisation due to the adoption of mobile and cloud computing [2].

Most commercial data loss detection methods are either signature or behaviour based [3].

1.1 Signature-based Detection

Signature-based data loss detection relies on matching patterns (such as keywords or watermarks) in external-bound transmissions [4, 5]. Watermarking and keyword matching solutions generate alerts if the sensitive data or hidden marks are identified in the outgoing computer network traffic.

Keyword matching software relies on locating text that is identical or similar to a known sensitive phrase or word. For instance, running a process to detect and quarantine emails leaving the organisation's network that contain the word "secret".

Digital watermarking introduces one or more small errors into a critical document. These intentional errors (for instance spacing, re-ordering, insertion of data and substitution) are called marks and all the marks together constitute the watermark. Monitoring systems can use watermarks to identify sensitive data within other computer network traffic. Identified files can be prevented from transfer to removable media or emailed out of the organisation's computer network.

Watermarking and pattern matching techniques can be defeated by data manipulation and substitution [6]. These techniques also require the system owner to continually identify critical data [7]. Text-based pattern matching is poor at interpreting misspellings, colloquialisms, abbreviations, nuance and inference in communications [8]. Finally, signature based network Intrusion Detection Systems (IDS) frequently suffer from generating huge volumes of false positives [9].

1.2 Behaviour-based Detection

Behaviour-based detection identifies anomalous events by comparing user and application activity against their normal patterns of behaviour [10]. An alert is generated if activity falls outside an expected range of individual user or population's activity [11], [12], [13].

Behaviour-based detection is generally implemented by statistical analysis of activity logs collected from across the computer network environment. Centrally collecting, storing and processing the required data to support behaviour-based detection is difficult to scale to support geographically dispersed and/or large computer network environments [14], [15].

Behaviour-based detection can also be defeated by gradual modifications of behaviour [10]. For instance, copying one sensitive file to removable media on the first day, then adding one or two additional files to the number copied each day.

Large and complex computer networks can contain hundreds of thousands of users, in many geographic locations accessing, copying, modifying and deleting documents each day. Practical implementation of behaviour-based detection within complex environments can result in the spawning of a significant number of false positives because users do not perform exactly the same tasks each day. The generated volume of alerts from these changes in behaviour produced by traditional detection system can quickly exceed the capacity of the incident response team [16]. The alternative approach is to de-sensitise the monitoring system by adjusting the alert threshold, which may result in a decrease in the detection of data theft (an increase in false negatives).

2 CANARY FILES

2.1 Overview

A Canary File is a fake file that is generated to aid the detection of unauthorised access to critical documents at rest. Canary Files are generated to emulate 'real' documents. They are stored within file directory structures that contain sensitive or critical files. These fake documents have no production value. Legitimate network users should not have a requirement to access the Canary File; any user attempting to open, copy or delete the fake file signals the presence of suspicious activity.

The function of a Canary File could be considered similar to a watermark for a file directory, providing a unique and recognisable indicator to the system administrator if the contents of the directory are accessed, copied or deleted. For instance, in the case of the WikiLeaks example, if Canary Files existed and were employed on the US Department of Defense computer network environment, one or more Canary Files are likely to have triggered an alarm when whole directories were copied to removable media.

Canary Files draw their name from canaries used in coalmines. Canaries were used in coalmines to alert the coal miners to the presence of dangerous gases. Canaries are especially sensitive to methane and carbon monoxide that are, in large quantities, toxic to humans. While the canary continued to sing, the miners knew their air supply was safe; a dead canary signalled an immediate evacuation [17].

Canary Files employ mimicry in order to avoid detection from malicious software or users. This mimicry is different from the deceptions used to lure or bait an intruder, such as a honeypot or honeyfile (discussed later in the paper).

Canary Files could be used to provide a complementary strategy to detect large-scale data loss, such as that experienced in the WikiLeaks incident in conjunction with other prevention and detection strategies.

2.2 Potential Advantages of Canary Files

Cohen, et al [18] looked at a mathematical structure of simple defensive network deceptions and concluded that mimicry could be of significant value, particularly if the author can generate convincing simulations. Deceptions that effectively mimic the target can be particularly effective against automated attacks, especially those which make decisions based on pattern matching with content or file attributes [19]. Brevity is a key requirement of malicious software. Concise malicious software is easier to hide within the contents of other benign software or within normal computer network traffic patterns. Succinct malicious software may not have the luxury of complex decision code necessary to differentiate between real and fake documents, which makes them more susceptible to interacting with fake objects, assuming realistic mimicry.

Canary Files are likely to be highly effective against attacks that involve a slow theft of data over a long period of time. Threat actors use low periodicity attacks in order to avoid behaviour-based detection. For instance, an attack that copies one file a week from a large organisation with a busy document repository is, in normal circumstances, unlikely to be detected by traditional auditing programs, because a threshold configuration setting to alert every single transaction within this environment is likely to overwhelm the audit team with false positives.

Behaviour-based detection may not discover malicious software or intruders that access critical data using the compromised credentials of legitimate employees (during expected working hours). The real information owner is unlikely to randomly access files within a directory that they are familiar with; they are likely to know which files contain relevant and useful information to their task at hand. Conversely, people and processes unfamiliar with the directory contents, but accessing records with compromised accounts, may access, copy or delete the Canary File, when searching for information within the directory.

Canary Files are likely to share other deception techniques' advantages in their abilities to discover an attacker's intentions and capabilities. Tracking a threat actor's activities can assist the organisation in identifying critical targets, attribution and securing the network against the actor's next action [20]. For instance, if several Canary Files containing information on a common theme are accessed by an intruder and emailed out of the network, then the system owner could apply additional control measures to quarantine external-bound emails containing information related to this set of critical information.

Canary Files do not do any harm or otherwise modify real documents. When inserted into a database, or file system, they are only likely to change the global statistics of the data set, but the original data elements will still remain intact [21].

Deception capabilities, like Canary Files, show potential for avoiding some of the problems frequently encountered by network intrusion-detection systems, for example, high false-positive rates and high false-negative rates for unknown attacks [19]. False positives are very common in security event logs [10]. In comparison, Canary Files have no production value and should never be accessed by anyone other than the creator or owner. Any access, copying or manipulation of the file is likely to be suspicious. The comparatively low volume of event data produced minimises the impact on Security Information and Event Management (SIEM) and log collection, searching and archive resources [10].

Canary Files offer the ability to improve the defence of critical data with a negligible burden on the hardware and software infrastructure for the system owner. Unlike security appliances, or even honeypots, Canary Files do not require any additional physical or virtual hosts to deliver a security capability. Canary Files only require the storage space equivalent to an additional document in the target directory.

3 MIMICRY

Canary Files are not designed to distract or bait intruders. The success of a Canary File lies in its ability to remain undetected within an enterprise document repository. A successful deception occurs when an unauthorised person or process accesses, copies or moves the Canary File when attempting to perform a nefarious intrusion activity on the critical documents associated with the Canary File (for instance the directory containing the Canary File and a collection of real critical files).

Fake files are not the first or only form of mimicry. The mimicry associated with Canary Files has its origin in natural science and military strategy.

Other forms of mimicry are also employed to protect computer network systems, however often this mimicry is designed to lure and attract rather than avoid detection.

3.1 Mimicry in Natural Science

Some of the earliest analytical work on mimicry has been through the field of natural science. Mimicry, in natural science, occurs when a group of organisms evolve to share common perceived characteristics with another group [22].

Plants and animals can employ mimicry to lure and entrap prey. The Anglerfish has a fleshy outgrowth protruding from the fish's head. The outgrowth acts as a bait to attract prey like a fisherman’s rod and lure. The Venus Fly Trap is a carnivorous plant that emits a deceptive scent to attract its insect prey.

Naturalists differentiate the employment of mimicry to lure (predator seeking a prey) with crypsis (prey avoiding a predator). The Canary File’s use of mimicry is more akin to the later than the former. In ecology, crypsis is the ability of an organism to avoid observation or detection by other organisms (often performed by deceiving adversaries into treating the plant or animal as something else). For example, several chameleon species are able to change their skin colours to mimic the environment and avoid detection.

3.2 Mimicry in Military Operations

Mimicry and deception are essential components of modern warfare. Modern military forces employ crypsis (camouflage) to conceal unit locations and movements from an adversary using combinations of materials, coloration and/or illumination.

The employment of mimicry can cause the adversary to misallocate personnel, fiscal, and material resources by over-committing to unimportant activities or non-existent targets [23]. For example, during World War II the British tricked the German Air Force into attacking non-existent airfields and factories by setting up phoney targets and interfering with the German electronic navigational aids. During the period of June 1940 to October 1940, the dummy aerodromes absorbed twice as many attacks as the parent stations they were protecting [24].

When paired with an ambush, mimicry can be used as a lure to catch the adversary off guard. A well-known example is the tale of the Trojan Horse recorded by Homer in the 12th Century BC, where a small number of Greek warriors hid within a giant horse presented as a gift to the besieged city of Troy.

Current US military doctrine states that deception techniques, lime mimicry, can cause “ambiguity, confusion, or misunderstanding in adversary perceptions” [25]. If an opponent is unable to discern the real from the fake, they can be uncertain and unconfident in their actions.

3.3 Mimicry Techniques in Cyber Security

Bellovin [26] was the first to publicly document a case study featuring the employment of mimicry in the defence of a computer network. Bellovin employed a variety of phony daemons with the production environment that instead of providing services logged the request and initiated counter-intelligence strategies to learn about the source of the attack.

Bellovin’s research led to the creation of honeypots. A honeypot is a fake computer system designed to appear as a fully functioning element of the infrastructure placed strategically to entice malicious intruders [27]. The primary functions of a honeypot are to discover new threat actors, their methods, and track their actions within the network [10]. Honeypots have no legitimate requirement to interact with other devices within the computer network. Any connection attempts are likely to be suspicious [28].

Honeyfiles are fake documents, rather than computer systems, generated for the sole purpose of baiting intruders [1]. The aim of the honeyfile is to entice a malicious actor to open and interact with the fake file. For example, a file named *credit-card-numbers.txt* could be created on a legitimate production workstation.

Honeyfiles and honeypots do not employ mimicry to hide from an adversary. The role of honeypots and honeyfiles is to present an attractive target to the intruder in order to: (1) divert attention away from critical assets, (2) introduce uncertainty about the real and the fake, and (3) profiling identity, capabilities, and intent by the creation of opportunity and the observation of action [29]. While Canary Files present some similarities between these other techniques, the role of the Canary File is more akin to crypsis and camouflage to hide, rather than aggressive mimicry to bait, lure and ambush.

4 CHALLENGES ASSOCIATED WITH GENERATING CANARY FILES

4.1 Realistic Mimicry

The success of Canary Files is dependent on their ability to avoid detection by internal and external threat actors seeking to access and export data without authorisation. Canary Files, like honeypots, can be fingerprinted. Fingerprinting is when an attacker can use the characteristics of the deception to detect its presence.

Rowe [30] developed a system to detect fake files by analysing the characteristics of the target document population. His program was successful in detecting documents with attributes that matched the population mean (for numeric parameters such as file sizes), median (for the more representative document names), or files that shared content with a collection of other documents within the population (for instance a network log file created by randomly choosing lines from a number of real log files).

Figure 1 is an illustration of a document generation system that employs data solely from the target environment. This type of generation process is likely to experience challenges when the population size is small, or the content is highly repetitive such as directories with one or two documents or directories with a single document, with multiple similar versions.

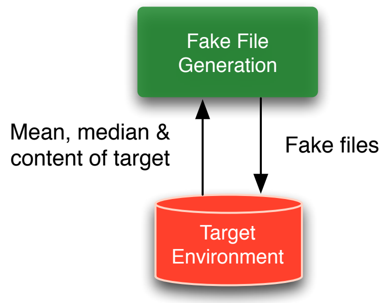

Figure 1. Fake file generation using data from the target environment

Rowe proposed a file generation process to counter his detection algorithm. Rather than generating fake files using statistical averages from the population group, Rowe employed data from outside of the target set (see Figure 2).

The limitation with Rowe's technique is identifying an external dataset that can successfully mimic the target population. Data drawn from an external weather web site is unlikely to successfully match a target directory containing medical records.

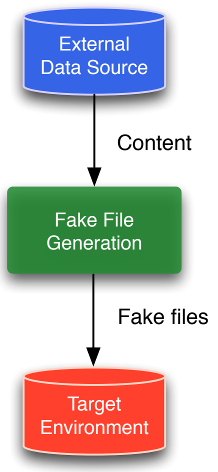

Figure 2. Fake file generation using data from an external data source

White and Thompson [21] employed external data to generate fake personnel records. Each personnel record contained sixteen individual attributes. White and Thompson used 16 individual data sources to generate the fake personnel records, for reasons of realism, rather than avoiding statistical analysis (see Figure 3). Their program randomly selected attributes (such as a first and last name) from separate pools of relevant harvested data, removing the selection from the pool after it was used.

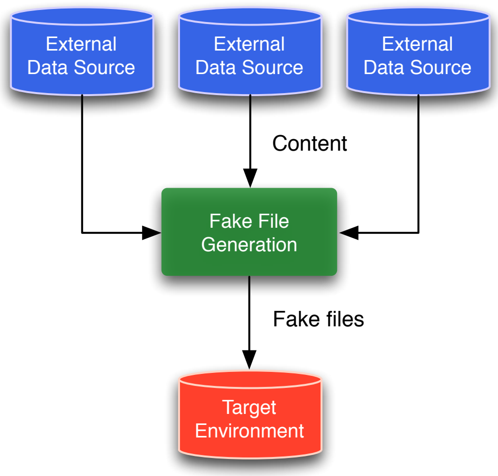

Figure 3. Fake file generation using data from multiple external data sources

Discarding data is a technique used in the operation of one-time cryptographic pads. While this technique can ensure the uniqueness of the data, or maintain the statistical distribution, it requires an inexhaustible supply of source data that accurately matches the target data set.

White and Thompson [21] observed three other challenges with employing an external data source to generate fake data: (1) When multiple components are generated independently, realism may be compromised – their program created uncommon and unusual ethnic pairings of first and last names; (2) the creation and management of a realistic data set is a very labour intensive task; and (3) employing a pool of source data requires resources to store and process. The latter would depend on the dataset, and while the storage and processing of this data may not be a significant burden, it needs to be considered.

4.2 Optimal Target Coverage

There is no precedent for optimal Canary File deployment. Canary Files, like honeypots, are worthless if the malicious user does not access them [10]. The challenge for the Canary File manager is to generate a sufficient quantity and to place these files in suitable locations in order to provide the greatest probability of data loss detection.

Not all directories are likely to warrant the deployment of a Canary File. Canary Files are more suited to file directories containing sensitive or critical files, where there is a risk of theft by an advanced adversary.

The identification of target directories and the deployment of a small number of Canary Files within a low complexity environment should be relatively simple; the security professionals responsible for delivery of the deception capability should have the capacity to individually create and monitor activity associated with a small set of fake files.

This task is much more difficult in large complex environments, where the system administration and security personnel are unlikely to have an understanding of every user’s role or the criticality of content on the network to the future success of the organisation. Generating large volumes of Canary Files with suitable realism, and their respective signatures for deployment on IDS, Anti-Virus (AV) and file monitoring software tools is unlikely to be something that can be undertaken manually. Finally, maintaining track of the location of Canary Files within the environment is likely to be a process more suited to automation.

4.3 Minimising Impact

Where possible a fake data generation system should minimise the number of files that are generated. Deploying too many Canary Files in a system may also have consequences. Canary Files have the potential to confuse legitimate users, distort the overall statistics of the records management database, and corrupt search indexes. Inserting fake records into a production environment can generate mistrust from legitimate users unsure which records are real or false [10]. Fake data can skew the properties of the actual data, and this bias can be very significant in certain applications [31].

The implementation of a fake data system should include sufficient user training and education. Manipulation of data within sensitive directories can also raise suspicion that the process is responsible for accidentally deleting or manipulating sensitive data, or that security staff are snooping on the actions of employees.

An important requirement of a fake data generation system is the ability to remove generated fake data when required. Manual management of these processes is not likely to be practical within large or complex environments.

4.4 Dynamic Defence

A Canary File generation system should be capable of rapidly generating and retiring Canary Files in order to adjust to the changes in the organisation's file repositories, create uncertainty in the minds of the adversaries, and avoid detection. In order to improve the chance of success, Dewar [32] argues, "*it is important that the deceiver keep a sufficiently open mind to be able to abandon or modify the deception plan without revealing the original aim*". Mel'nikov [33] also highlights the value of using initiative and creativity in executing deception measures that may reduce the impact of any detection. Tirenin, and Faatz [34] argue that in order to be most effective, a cyber deception capability must be dynamic in its implementation: (1) presenting a continually changing picture to the enemy, and (2) changing more rapidly than the enemy is able to interpret the information.

While an agile Canary File capability is likely to increase the effectiveness of the deception and reduce the likelihood of detection, rapid changes should be transparent and accessible to the system owner in order to differentiate the real from the fake, if required. This implies maintaining situational awareness of the composition and location of all of the current and retired Canary Files.

Dynamic defence also implies an ability to act quickly in response to the detection of a possible data loss incident. Automated generation of incident tickets is common practice with other cyber security systems, often paired with sending alarms to the mobile devices of personnel responsible for the security of the computer network [19]. Active response for Canary Files could also include the option to quarantine the target directory, or disable removable media, using access control mechanisms similar to the automated physical lock down defences in modern banks. If the access control system cannot be trusted in the event of a possible data loss incident, a Canary File alarm could trigger an automated transfer of the critical data into a designated safe environment, commensurate with a panic room.

4.5 Legality

A key concern raised regarding the use of honeypots and other related deception techniques is the characterisation of such activities as a form of entrapment. Entrapment refers to the inducement by public law enforcement authorities and their agents to commission a crime. The applicability to Canary Files depends on the jurisdictional interpretation. Owners will need to assess this risk before deploying a Canary File system.

4.6 Secrecy

Canary Files, like other deceptions, are likely to be more effective if the knowledge of their existence on the network is managed carefully. Machiavelli stressed the importance of maintaining the secrecy of a deception. "*When you are aware that the enemy is acquainted with your designs, you must change them. After you have consulted with many about what you ought to do, confer with very few about what you are actually resolved to do*" [35]. Grieffenberg was more direct stating "*If the strictest secrecy is not observed all deception projects, [they] are condemned to failure from the start*". As far as the consequences of withholding information are concerned, he also stated, "*Deceiving one's own troops for the sake of security is a normal by-product of deception*" [36].

Cohen, et al [18] have an alternative view. They believe excessive security hinders coordination and argues there should be a balance between protection and effectiveness. "An enemy is always alert for indications and warnings; hence perfect security does not exist". Cohen, et al recommends that managers should expect their deception plans to be compromised, and should be ready to use any security breaches to their advantage.

5 THE SERINUS SYSTEM

5.1 Overview of the Serinus System

This paper also introduces the Serinus System. The aim of the Serinus System is to addresses the challenges associated with employing Canary Files. Serinus performs several key functions: (1) automate the generation of Canary Files, (2) manages the deployment of the Canary Files, (3) automates the signature generation to leverage third party AV, IDS and file integrity systems, and (4) automate a continuous and dynamic retirement and generation of Canary Files across the system in order to create uncertainty for the intruder and improve the chance of avoiding detection.

The word *Serinus* is associated with a genus of birds, which includes the finch and canary family.

5.2 Canary File Generation

The Serinus System generates Canary Files using data from both external and target sources in order to create realistic mimicry and avoid statistical based detection (see Figure 4).

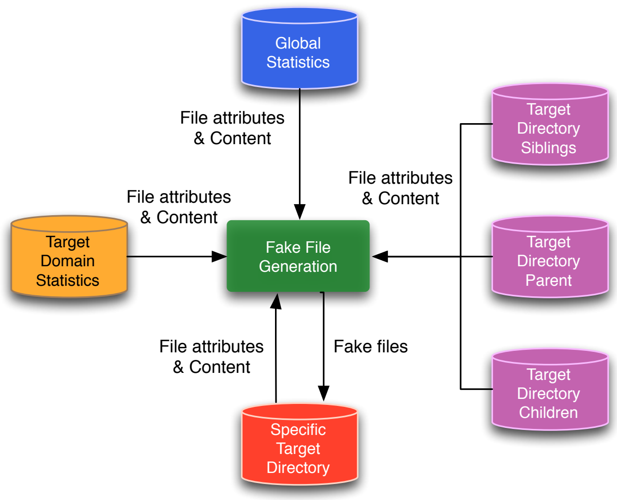
Global Statistics
File attributes & Content
File attributes & Content
Target Domain Statistics
Fake File Generation
File attributes & Content
Target Directory Siblings
Target Directory Parent
Target Directory Children
File attributes & Content
Fake files
Specific Target Directory

Figure 4. The Serinus System fake file generation

This technique seeks to leverage the lessons learn by Rowe [30] – described in section 4 of this paper.

Global statistics are a set of popular file attributes and contents drawn from outside of the target environment. The Serinus System uses file data from the Internet collected through mining search engine results returned from Google. Global statistics are valuable when the target domain and directory have a low population of data files.

Domain statistics comprises data drawn from across the entire selected target environment. The other data sets are local statistics from the specific target directory (See Figure 5).

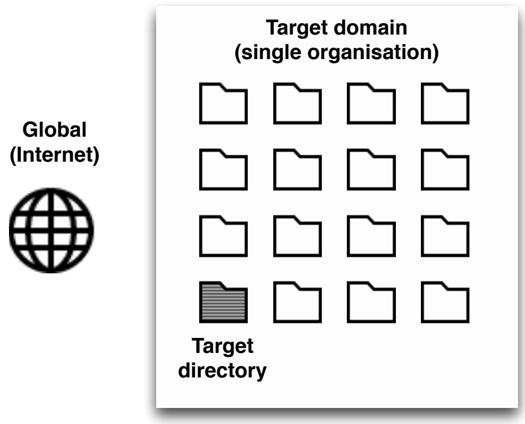

Figure 5. The three sources of data (global, domain and target directory)

Statistics are gathered on file attributes such as: (1) file names, (2) file sizes, (3) file creation dates, (4) file modification dates, (5) access control attributes, and (6) number of images. Statistics are grouped by file type (See Figure 6). A small sample of document contents is also collected.

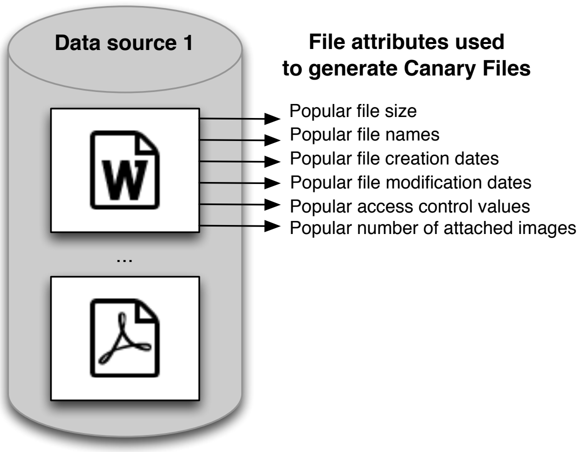
Data source 1
File attributes used to generate Canary Files
Popular file size
Popular file names
Popular file creation dates
Popular file modification dates
Popular access control values
Popular number of attached images
...

Figure 6. File statistics are grouped by source (global, domain and target) and file type

The current version of the Serinus System only supports text (txt) files. Text files were chosen in the initial implementation for simplicity; text files do not contain complex formatting and embedded images. Limiting the Serinus System to text files also allows for a larger sample collection of global statistics to be collected from the Internet due to the average size of the files. Other popular file types, such as: (1) rtf, (2) doc, (3) docx, (4) pdf, and (5) odt will be considered for inclusion at a later date.

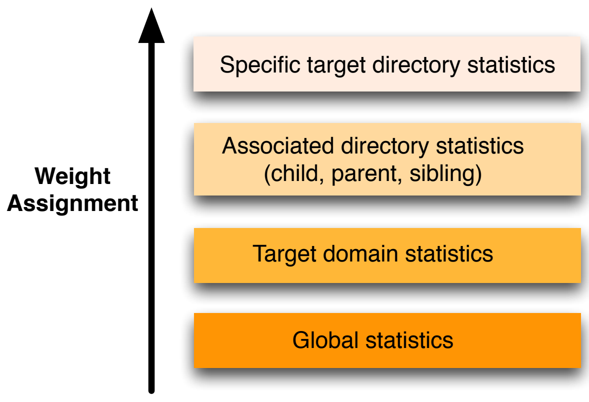
Weight Assignment
Specific target directory statistics
Associated directory statistics (child, parent, sibling)
Target domain statistics
Global statistics

Figure 7. Weighting is based on data origin

Each set of statistical data is allocated a weight based on their relationship with the target directory. Figure 7 illustrates that statistics obtained from the sibling directory are afforded a higher weighting than statistical data originating from either the entire target domain or the external data set (global). Weighting is important in order to ensure that the generated Canary File will match the target directory. Identification of suitable weightings is part of on-going research.

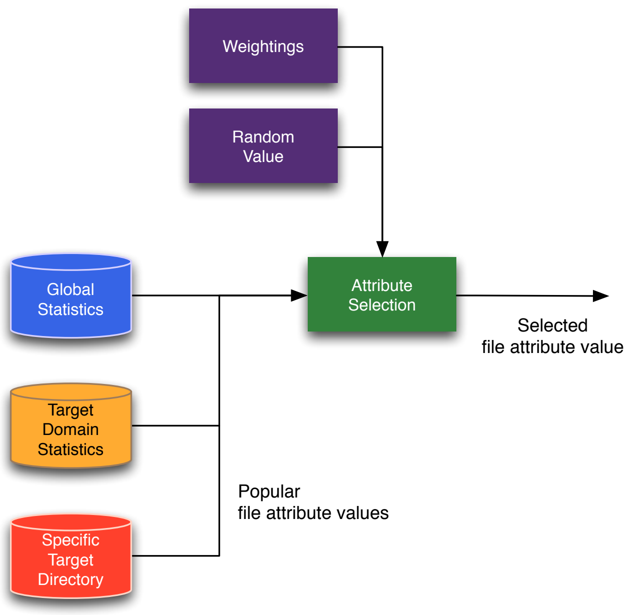
Weightings
Random Value
Global Statistics
Target Domain Statistics
Specific Target Directory
Attribute Selection
Selected file attribute value
Popular file attribute values

Figure 8. Attribute selection is based on popular selections from several weighted data sources, seeded with a random value

Collected statistics are also modified (and selected) by a random value, in addition to a proximity-based weighting (see Figure 8). The random value and weighting help to create variance that reduces automated deception detection programs that are capable of identifying files that conform to target domain and directory averages.

5.3 Canary File Deployment

The Serinus System simplifies and automates the deployment of Canary Files. An illustration of the deployment process is provided in Figure 9. Each step is explained below.

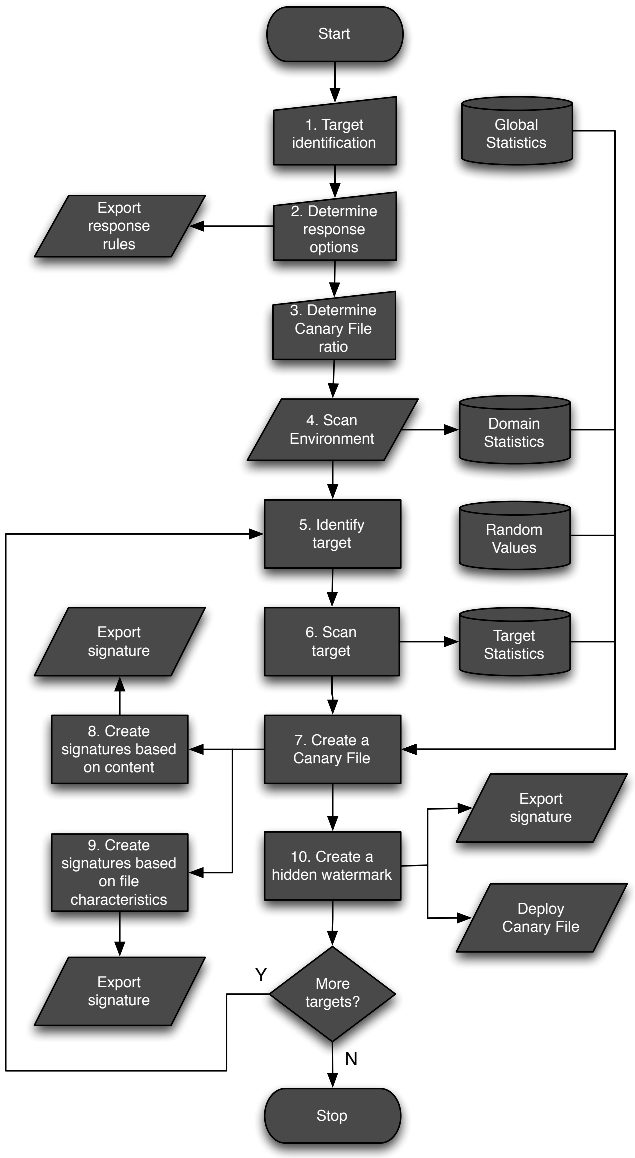

Figure 9. Canary File deployment using the Serinus System

Step 1: Target Identification. The first step in the process requires the user(s) to nominate the target environment. A target environment is a directory or set of directories within a file system or records management system that contains sensitive information.

It is also possible to automate this step using a process, such as the critical data selection algorithms proposed by [37], [38], [39]. Automations could be employed to select critical data without user intervention, or to propose options to a human user.

Step 2: Determine Response Options. The aim of this step is to field a dynamic defence. This step requires the user(s) to nominate one or more automated response policies in the event of a Canary File alert. Options can include: (1) sending alerts to the information technology ticketing system, SIEM, telephony or paging service, or an email or messaging service; (2) modifications to the directory or physical (to prevent users leaving the building) access control permissions for an individual or group; (3) modifications to removable media or printing permissions; or (4) transportation of the document to a nominated secure containment area.

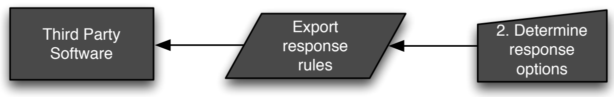

Figure 10. Exporting response rules to third party software

Policies could be created based on the detected event (copy, delete, access), the sensitivity of the data, or set globally. Response policies (rules) can be exported to one or more external decision systems, such as SIEMs for enforcement.

The current version of the Serinus System exports rules for OSSEC an open source host-based IDS with an active response capability (see Figure 10).

Step 3: Determine Canary File Ratio. The aim of this step is to address the challenge of fielding sufficient Canary Files to provide optimal coverage of the target environment. Too many Canary Files can create confusion and impact on the productivity of users. Too few Canary Files can result in undetected data loss.

In this step, the user(s) nominate an approximate ratio between the number of production files and the number of desired Canary Files within the target environment. Research is on-going to determine optimal Canary File ratios.

Step 4: Scan Environment. The aim of this stage is to collect statistical data from across the enterprise environment to support the generation of fake files (see Figure 11). Generation of fake files in undertaken in Step 7.

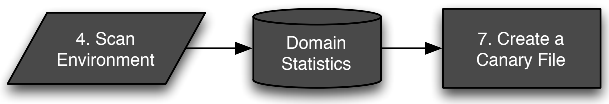

Figure 11. Scanning the environment collects statistical information that is used to generate realistic fake files

Statistics are gathered on file attributes such as: (1) file names, (2) file sizes, (3) file creation dates, (4) file modification dates, (5) access control attributes, and (6) number of images. Statistics are grouped by file type. Information is also gathered on each file's location within the file system.

This step needs to be completed rapidly to reduce the chance of modifications to the file system during the scanning process. The current process uses the inherent file listing process (ls) distributed as part of the base Linux operating system paired with the 'file' command to inspect and classify the directory contents.

Step 5: Identify Target. This is the first step in the generation of each individual Canary File. The Canary File population ratio set by the user(s) may not be sufficient to deploy a Canary File in every directory within the target environment. For instance, the target environment may have 1000 individual directories and the selected ratio may only allow for the generation of 50 Canary Files across the environment. This step calculates the number of Canary Files that will be initially deployed and compares this value to the number of individual child directories within the target environment. The Serinus System conceives a deployment plan and the process to generate the Canary Files is initiated.

The current deployment plan is produced using random values and considers each target directory having an equal priority. Algorithms proposed in [37], [38], [39] could be used to address the more critical directories first.

Step 6: Scan Target. This stage involves a scan of the immediate target directory to collect content that can be used to generate the fake files. Statistical data is already collected in Step 4. This step is undertaken for each target directory identified during Step 5.

Step 7: Create Canary File. Canary Files are generated using data collected from the Internet, environment and target directory in accordance with the previously described process (see Figure 12).

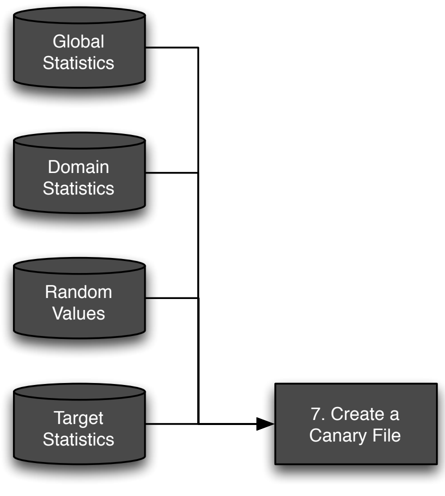

Figure 12. Global, domain and target file statistics are used in conjunction with random values to create realistic fake files

Step 8: Create Signatures Based on Content. The aim of this step is to use third party IDS and AV software to detect if part of a Canary File is copied to another part of the system (including removable media) or transferred out of the computer network through the organisation's Internet gateway. The contents of the Canary Files have no production value and either activity is likely to be suspicious, and should be investigated by an incident response team. This step is complementary to Step 10, which is designed to detect if the entire Canary File is copied to another part of the system (including removable media) or transferred out of the computer network.

The Serinus System is able to detect malicious processes that aim to avoid detection by traditional data loss detection systems that steal file contents rather than whole files. The contents of the Canary Files are divided into blocks of text. Each block of text is used to create a unique signature for external AV and IDS (see Figure 13).

It is important to create a balance when selecting the unit of division within a file. Too many signatures can impact the performance of the IDS and AV systems. Too few signatures can result in undetected data loss if the adversary chooses to extract smaller components of each file.

The current Serinus System creates a signature for each paragraph inside the Canary File. Paragraphs were chosen because they provide a distinct unit within the text file. Future research will consider if smaller or larger units should be used for signature generation.

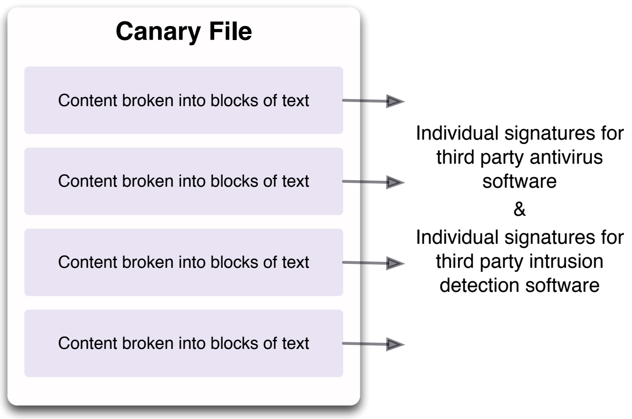
Canary File
Content broken into blocks of text
Content broken into blocks of text
Content broken into blocks of text
Content broken into blocks of text
Individual signatures for third party antivirus software
&
Individual signatures for third party intrusion detection software

Figure 13. Canary File contents are used to create signatures in order to detect attempts to exfiltrate partial files

The IDS signatures can be used on the network perimeter to detect if some or all of the contents of the Canary File have been exported from the network. The Serinus System uses Snort, an open source IDS software [40]. The Serinus System creates signatures in a format that is exportable to the Snort software.

Snort version 2.9.4 uses the Boyer-Moore pattern match function, developed by Robert S. Boyer and J Strother Moore in 1977 [41]. The algorithm's performance improves as the pattern length increases, making it ideal for matching Canary File paragraphs.

Snort signatures can be created to match strings against the contents of files rather than inspecting every packet, to assist in performance improvement and decode MIME and other content transpositions / substitutions. Data contained within documents is often repacked in order to optimise delivery and management. Signature based detection system should reconstruct the file from the data stream and decode its contents before the inspection process, in order to successfully match content.

AV signatures can be used to scan the file systems to detect duplicates. Duplicate files can indicate that one or more of the content blocks within the fake file have been copied. This event could indicate that one or more critical files within the target directory have also been copied.

The Serinus System exports signatures created from content into a format compatible with ClamAV. Clam AV was chosen because it is a widely used free open source, multi-platform AV tool. System administrators can configure desktop and server settings to scan removable media with ClamAV before mounting or removal.

At the conclusion of the file generation process, the Serinus System stores the IDS and AV signatures in a directory that can be accessed by ClamAV and Snort software. Snort and ClamAV rules contain a unique reference to the Canary File associated with the alert.

Signature based detection requires an exact match to the text blocks (in this case paragraphs) in order to trigger an alert. Future research could consider fuzzy signature generation to improve the chance of detection from malicious activities that involve partial content extraction. Any changes to the detection algorithms should consider real-time detection performance in a high-speed network.

Step 9: Create Signatures Based on File Characteristics. The aim of this step is to construct signatures to allow third party file-monitoring software to detect if the Canary File is accessed, modified, deleted or copied.

```
2013 Jan 07 10:22:58,0 - /files/canary1
File changed. - 1st time modified.
Integrity checking values:
        Size: >28050
        Perm: rw-r--r--
        Uid:  0
        Gid:  0
        Md5:  >50da55def41bcede…
        Sha1: >97f4b2b48a97321a3…
```

Figure 14. An example of an OSSEC alert generated as a result of a modification to a Canary File

The Serinus System uses OSSEC, a free, open source host-based IDS that performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response. OSSEC can be used on Linux, MacOS, Solaris, HP-UX, AIX and Windows.

Figure 14 illustrates an example alert produced by OSSSEC when a file is modified. Details of the changes can also be reported by OSSEC.

At the conclusion of the Canary File generation process the Serinus System generates a set of signatures that can be imported into OSSEC.

Step 10: Create Hidden Watermarks. The aim of this step is to utilise third party IDS and AV software to detect if the entire Canary File is copied to another part of the system (including removable media) or transferred out of the computer network through the organisation's Internet gateway. Canary Files have no production value and either activity is likely to be suspicious, and should be investigated by an incident response team. This step is complementary to Step 8, which is designed to detect if part of the contents of a Canary File is copied to another part of the system (including removable media) or transferred out of the computer network.

The Serinus System employs hidden watermarks within the Canary Files as the basis for generating AV and IDS signatures, in preference to generating signatures on the entire contents of the file. Hidden watermarks are resistant to file name changes that can be used to avoid detection by data loss prevention technologies.

A watermark is generated for each file to support IDS and AV detection systems. Watermarks are individually paired for each Canary File to allow the detection systems to simplify the alert generation process. Signatures are created in a format that is exportable.

The Serinus System uses Snowdrop, an open source tool that inserts an MD5 hash into a text file. Snowdrop uses whitespace reformatting, inserting mistakes, word substitutions and punctuation changes to hide the MD5 hash. Snowdrop was chosen because of the speed of extraction and matching. Alternatives could be considered as part of future research that may be less susceptible to data manipulation attack.

This process is continued until the required ratio of Canary Files is achieved.

5.4 Continuous Management of Canary Files

One of the key challenges identified when employing deception capabilities within a complex environment is the requirement to maintain Canary Files spread across multiple file and records management systems. The Serinus System simplifies the on-going management of the Canary Files within the target environment by: (1) randomly retiring Canary Files, (2) updating file attributes (the timestamp when the file was last accessed) of existing Canary Files to ensure that the deception is maintained, and (3) creating new Canary Files to maintain the required ratio of Canary Files when one is retired.

New Canary Files may not necessarily be deployed to the location where the previous file was retired. A new location will be chosen at random from the list of directories within the target environment that does not already contain a Canary File. The aim of this continual renewal process is to address the challenges of providing coverage across a large target environment and the risk of detection.

Finally, the Serinus System supports the identification of Canary files that have been compressed by an adversary before exfiltration by examining the event records generated by the compression software. File compression generates a new file that is (hopefully) a more space efficient representation of the original document. This new document is unlikely to present data in a way that matches the original signatures that were generated from the Canary File's contents. Compression can be used as a mechanism to avoid signature based detection by content inspection systems, such as those created during Steps 8 and 10. Compression tools usually record details of the files that are compressed by the software. These details can be mined by the Serinus System to identify the name of the newly created compressed file, and where possible, generate new IDS, AV and file integrity signatures.

6 RELATED WORK

There is a small body of research conducted into honeyfiles. In 2004, Yuill, et al [19] proposed a honeyfile system that could generate: (1) files with information pertaining to the security of the environment, such as password files, user manuals, and network diagrams; (2) system or application programs that an intruder may operate, but an authorised user does not have a requirement to employ, such as a software compiler; (3) files that contain evidence of the attack, such as log files; and (4) files that contain sensitive information, such as credit card numbers, intellectual property, and military intelligence.

A year later, Cenys, et al [42] created a deception capability within an Oracle Database Management System in order to trap and lure insider threats.

In 2009, White, and D. Thompson [21] created a program that could insert synthetic decoy personal records into a database in order to detect data loss and attribute the data spill incident associated with an identify theft crime.

That same year, Bowen et al published research involving a Decoy Document Distributor that automated the creation of bait files to trap intruders [43].

The common challenges with each of these processes are: (1) their inability to scale in order to provide protection across a large and complex network, and (2) the generation of realistic documents that are capable of avoiding detection.

Jadhav [44] proposed the creation of fake objects that acts as a type of watermark for a real set of data. Unfortunately, while he made the observation, he concluded that the creation of fake but real-looking objects is a non-trivial problem that was beyond the scope of his research.

7 FURTHER RESEARCH AND CONCLUSIONS

7.1 Conclusions

This paper introduces Canary Files and the Serinus System. A Canary File is a fake file placed amongst real documents in order to aid early detection of data theft. Canary Files draw their name from canaries that were used within coalmines to provide miners early warning of danger.

Canary Files emulate real documents within the production environment. They are stored within existing file directory structures, which contain sensitive or critical files. Canary Files documents have no production value. Legitimate network users should not have a requirement to access the fake files and any user attempting to open, copy or delete them signals the presence of suspicious activity.

Canary Files employ mimicry in order to avoid detection from malicious software or users that may seek to avoid triggering an alarm when stealing sensitive data from an organisation. This mimicry is different from the deceptions used to lure or bait an intruder, such as those employed by honeypots and honeyfiles.

Canary Files are likely to generate a relatively small number of high priority alerts compared to traditional data loss detection systems, and be effective against more advanced threat actors. Canary Files should complement more traditional data loss detection / prevention approaches, providing defence in depth and alternative options to guard against information theft from advanced adversaries.

This paper also introduces the Serinus System, a Canary File management system designed to address the limitations of generating and managing fake files within a complex enterprise environment. The Serinus System generates Canary Files using data obtained from the target environment and external pool of statistics based on properties of similar files found outside the target environment.

The Serinus System leverages third party software tools, by automating the generation of AV, IDS and file integrity signatures. The Serinus System also automates a continuous retirement and creation of Canary Files in order to avoid detection and create uncertainty for the intruder associated with a dynamic defence.

7.2 Further Research

The next stage of the research is to test the effectiveness of Canary Files and the Serinus System using cyber security specialists and against statistical based detection methods. Current research is seeking to: (1) evaluate the ability of Canary Files to detect data loss, (2) determine optimal ratios of Canary File deployments within an organisation's file repository, and (3) test the weightings assigned to the global, enterprise and target file contents and attributes.

Further research could involve testing the ability of the Serinus System to generate and manage misinformation in a large and complex network. Canary Files could be generated to deceive an adversary with false content.

There is also scope to expand the capabilities of the Serinus System to generate more varieties of documents, or enhance the watermarking, AV and IDS integration and performance capabilities.

8 REFERENCES

1. Shabtai, A., Elovici, Y., Rokach, L.: A Survey of Data Leak Detection and Prevention Solutions. Springer, New York (2012).
2. Ponemon, L.: Fourth Annual US Cost of Data Breach Study. Technical report, Ponemon Institute, (2009).
3. Ouellet, E., Proctor, P.E.: Magic Quadrant for Context Aware Data Loss Prevention, Technical Report. RA4 06242010. Gartner RAS Core Research, (2009).
4. Aho A. V., Corasick, M. J.: Efficient string matching: An aid to bibliographic search, Commun. ACM, vol. 18, no. 6, (1975).
5. Heberlein, T., Dias, G., Levitt, K., Mukherjee, B., Wood, J., Wolber, D.: A network security monitor. In: Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, pp. 296–304. IEEE, IEEE Comput. Soc. Press, Los Alamitos, CA, USA, (1990).
6. Kale, S. A., Kulkarni, S. V.: Data Leakage Detection. In: International Journal of Advanced Research in Computer and Communication Engineering, vol. 1, issue 9, pp. 32-35, Nov (2012).
7. Borders, K., and Prakash, A.: Quantifying information leaks in outbound web traffic. In Proceedings of the IEEE Symposium on Security and Privacy, May (2009).
8. Ierusalimschy, R.: A Text Pattern-Matching Tool Based on Parsing Expression Grammars. In: Software-Practice and Experience, vol. 39, pp. 221-258, (2009).

9. Sommer,: Enhancing Byte-Level Network Intrusion Detection Signatures with Context. CCS'03, October 27–30, Washington, DC, USA (2003).
10. Denning, D.E.: An Intrusion Detection Model. IEEE Transactions on Software Engineering, 13(2): 222–232, February (1987).
11. Smaha, S. E.: Haystack: an intrusion detection system. In: 4th ACSAC, pp. 37-44. IEEE Press, Washington (1988).
12. Anderson, J.: Computer Security Threat Monitoring and Surveillance. James P. Anderson Co., Fort Washington (1980).
13. Denning, D. E.: An Intrusion-Detection Model. In: IEEE Transactions on Software Engineering, vol. 13, Issue 2, pp. 222-232. IEEE Press, Piscataway (1987).
14. Joshi, R. C., Sardana, A.: Honeypots: A New Paradigm to Information Security. Science Publishers, Enfield, (2011).
15. Kayacik, H. G.; Zincir-Heywood, A. N., Heywood, M. I.: On dataset biases in a learning system with minimum a priori information for intrusion detection. Communication Networks and Services Research Conference, pp. 181–189, (2004).
16. Proctor, P.E.: Practical Intrusion Detection Handbook. Prentice Hall PTR, Upper Saddle River, NJ, USA, (2000).
17. Burnley, A.: Gas Detection: Mining, Health and Safety Practices, Regulations. In: Health and Safety International. January, pp. 15-25, (2007).
18. Cohen, F., Lambert, D., Preston, C., Berry, N., Stewart, C., Thomas, E.: A Framework for Deception. Computers and Security, (2001).
19. Yuill, J., Zappe, M., Denning, D., Feer, F.: Honeyfiles: Deceptive Files for Intrusion Detection. In: Proceedings of the 2004 IEEE Workshop on Information Assurance. United States Military Academy, West Point, NY, (2004).
20. Gerwehr, S., Weissler, R., Medby, J., Anderson, R.H., Rothenberg, J.: Employing Deception in Information Systems to Thwart Adversary Reconnaissance-Phase Activities (OUO). PM-1124-NSA0, RAND National Defense Research Institute, November (2000).
21. White, J., Thompson, D.: Using synthetic decoys to digitally watermark personally identifying data and to promote data security. In: Proceedings of the International Conference on Security and Management, June, (2006).
22. King, R. C.; Stansfield, W. D.; Mulligan, P. K.: *A dictionary of genetics* (7th ed.). Oxford: Oxford University Press. p. 278, (2006).
23. C. Von Clausewitz, On War, edited and translated by Michael Howard and Peter Paret, Princeton Princeton NJ: University Press, 1976.
24. M. Howard, Strategic Deception in the Second World War, Pimlico: London. 1992.
25. US Joint Publication 3-58, *Joint Doctrine for Military Deception*, (1996).
26. Bellovin, S.: There be Dragons. In: Proceedings of the Third Usenix UNIX Security Symposium. Baltimore, September (1992).
27. Gupta, N.: Improving the Effectiveness of Deceptive Honeynets Through an Empirical Learning Approach, (2003).
28. Spitzner, L.: Honeypots—Tracking Hackers. In: Addison-Wesley, Indianapolis, pp. 242–261. (2003).
29. Gerwehr, S., Rothenberg, J., Anderson, R.H.: An Arsenal of Deceptions. In: INFOSEC (OUO), PM-1167-NSA, RAND National Defense Research Institute Project Memorandum, October (1999).
30. Rowe, N. C.: Automatic Detection of Fake File Systems. Cebrowski Institute, U.S. Naval Postgraduate School, (2006).
31. Adams, N., Worthmann, J.: Security-control methods for statistical databases: a comparative study. ACM Computing Survey, vol. 21, issue 4, December, pp. 515 – 556, (1989).
32. Dewar, M.: The Art of Deception. David and Charles Military Books, (1989).
33. Mel'nikov, P.: Operatihnmaya Maskirovka [Operational Deception]. Voyenno Istoricheskil Ztumal [Military History Journal]. April, pp. 18–26, (1982).
34. Tirenin, W., Faatz, D.: A Concept for Strategic Cyber Defense. In: Military Communications Conference Proceedings, MILCOM 1999, pp. 458–463, (1999).
35. Machiavelli, Frederick II (King of Prussia), The Works of Nicholas Machiavel, vol. 2, translated by T. Davies, (1762).
36. Daniel, D.C., Herbig, K.L.: Propositions on Military Deception. In: Gooch, J., Perlmutter A. (eds), Military Deception and Strategic Surprise, Frank Cass, (1982).
37. White J., Panda, B.: Insider Threat Discovery Using Automated Detection of Mission Critical Data Based on Content. In: Proceedings of the 5th International Conference on Information Systems Security, pp. 227-232, (2009).
38. B. Yang, J. Sun, T. Wang, C. Zheng, "Effective multi-label active learning for text classification", In Proceedings of KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 917-926, New York, NY, USA, (2009).
39. D. Sculley, G. Wachman, "Relaxed online SVMs for spam filtering", In Proceedings of the Thirtieth Annual ACM SIGIR Conference, (2007).
40. Roesch, M. Snort: lightweight intrusion detection for networks In Proceedings of the 13th Conference on Systems Administration (LISA-99) (1999).
41. Boyer, Robert S.; Moore, J Strother (October 1977). "A Fast String Searching Algorithm.". Comm. ACM (New York, NY, USA: Association for Computing Machinery) 20 (10): 762–772. doi:10.1145/359842.359859. ISSN 0001-0782.
42. Cenys, A., Rainys, D., Radvilavius, L., Gotanin, N., Implementation of Honeytoken Module in DSMS Oracle 9iR2 Enterprise Edition for Internal Malicious Activity Detection. In: Proceedings of Conference on Detection of Intrusions and Malware & Vulnerability Assessment, (2005).

43. Bowen, B. M., Hershkop, S., Keromytis, A. D., Stolfo, S. J.: Baiting Inside Attackers Using Decoy Documents. In: Proceedings 5th International ICST Conference (SecureComm ’09), (2009).
44. Rekha Jadhav, Data Leakage Detection, International Journal of Computer Science & Communication Networks,Vol 3(1), 37-45