APPROVAL SHEET

Title of Thesis: EBIDS-SENLP: A System to Detect Social Engineering Email Using Natural Language Processing

Name of Candidate: Allen Brian Stone
M.S. Computer Science,
May 2007

Thesis and Abstract Approved: _____

Dr. Alan Sherman
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County

Date Approved:

Curriculum Vitae

Name: Allen Brian Stone

Permanent Address: 3008 Baybriar Rd, Dundalk, MD, 21222

Degree and date to be conferred: M.S. Computer Science, May 2007

Date of Birth: March 2, 1980

Place of Birth: Baltimore, MD

Secondary Education: Archbishop Curley High School, Baltimore, MD

Collegiate institutions attended:

University of Maryland Baltimore County, B.S. Computer Science, 2003

Major: Computer Science

Professional positions held:

Tools Integrator (Software Developer and Researcher) (Aug. 2003 to current)
(formerly a Network Security Analyst)
Jacob & Sundstrom, Inc.
401 E. Pratt St, Ste. 2214, Baltimore, MD, 21204

ABSTRACT

Title of Thesis: EBIDS-SENLP: A System to Detect Social Engineering Email Using Natural Language Processing
Allen Brian Stone, M.S. Computer Science, 2007
Thesis directed by: Dr. Alan Sherman, Dr. Sergei Nirenburg, Dr. Charles Nicholas
Keywords: ***Social Engineering, Natural Language Processing, OntoSem, Intrusion Detection***

EBIDS-SENLP is an Ontology-based Intrusion Detection System that uses natural language themes, specifically manipulative themes for the purpose of Social Engineering (online fraud), to detect such manipulation in email text. The project includes a performance test against two industry standard intrusion detection systems, Snort and SpamAssassin, to see if the new approach is feasible and how it stacks up initially. The project features a novel algorithmic approach to detection of malicious content by utilizing the Natural Language Processing capabilities of the UMBC ILIT Laboratory's OntoSem project to parse and understand the email text to ferret out the concepts of manipulation in the emails. This project was shown to present an immediate value to the realm of Network Defense, because, although it was outperformed by SpamAssassin in testing, it still showed an impressive 75% detection rate with only four detection rules in its signature set and a very low 1.4% false positive rate. The detection rate is low for a production system, but it is a promising start by any means, and the false positive rate is much lower than anyone involved in the project expected. Thus, if the signature set is updated significantly, this product can approach the performance of SpamAssassin and do so with a much smaller and more easily adaptable signature set (it is based on English Language concepts instead of digital signatures).

EBIDS-SENLP: A System to Detect Social Engineering Email Using Natural Language Processing

by
Allen Brian Stone
University of Maryland, Baltimore County

Thesis submitted to the Faculty of the Graduate School
Of the University of Maryland in partial fulfillment
Of the requirements for the degree of
Master of Sciences, Computer Science
2007

Dedicated to my long-suffering friends and family

ACKNOWLEDGEMENTS

The author would like to thank all who have made this work possible. I want to thank Jacob & Sundstrom, Inc., and their client staff for being patient and flexible and supporting my academic career. I would also like to thank Dr. Jose Nazario, who provided the phishing corpus and provided support to me. I would like to personally thank Jesse English for all of his help understanding and utilizing OntoSem through the Dekade II interface, along with thanks to Dr. Nirenburg. I would like to thank Dr. Sherman for guiding me and encouraging me in this endeavor, despite the relative dearth of research on the topic when I began. I would like to thank Dr. Nicholas for all of his help throughout my academic career at UMBC. Most of all, I want to thank the friends and family who believed in me for all of these years and sacrificed time spent with me to allow me to advance to this point.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction to the Problem: Social Engineering

Social Engineering is the act of gaining unauthorized access or information from a computer system or business location by deceiving the humans that have knowledge about the target, usually through charm and cunning. It represents a significant threat to even the most technologically secure networks, because it exploits the human element. According to Kevin Mitnick, one of the world's foremost "social engineers", "A company may have purchased the best security technologies that money can buy, trained their people so well that they lock up all their secrets before going home at night, and hired building guards from the best security firm in the business. That company is still totally vulnerable" (Mitnick & Simon, The Art of Deception, pg 3). Its most common means of reaching the user now is email. It usually involves impersonation of some official-sounding person or organization, some exploitation(s) of human psychology, and then a request for action, which usually involves either opening the email, clicking a link, logging into some server, or passing personal or network information back via web or email. It is extremely common to see these emails as they relate to stealing information with the intent of financial fraud, or "phishing", where they will refer to a user's financial accounts that may already be on the internet in some form.

The mechanisms that exist to foil Social engineering are varied, some more effective than others. Personnel resistance training is the most effective, since the "wetware" (human user) is what the attackers are targeting specifically. Training of this sort involves introducing the user base to the sort of attacks that can happen and providing them with the correct action and outlet for when such things occur. This is very effective, especially in the time directly after training occurs, since it is fresh in the user's mind. Training of this sort requires that the organization issuing it knows which of its employees need what extent of training. That organization also has to set mechanisms in place for handling the event that such things occur, so that attacked users have an outlet for response, since they will most likely not be the only users/employees targeted. This is effective, because an informed and prepared human user can be vigilant and confident, and confidence and knowledge of the problem are the best ways to thwart any social engineering attempts.

However, this approach also has major drawbacks. People can forget or ignore the training over time or are not always going to detect that it is occurring, since people have a tendency to build a false sense of confidence as time passes without seeing such attacks. Plus, the average human end user does not think in terms of exploiting other people similarly or why some information is sensitive, so the detection rate is low, even when training is fresh in their minds. And naturally, some people are less motivated or capable at their positions than others and will not go the extra step of being prepared and vigilant against these attacks.

Subject: Notice of blocked account
From: "PayPal" <Service@PayPal.com>

Security Center Advisory

We recently noticed one or more attempts to log in to your PayPal account from a
foreign IP address and we have reasons to belive that your account was hijacked
by a third party without your authorization. If you recently accessed your account while

traveling, the unusual log in attempts
may have been initiated by you.
If you are the rightful holder of the account you must click the link below

and then complete all steps from the following page as we try to verify your

identity.
Click here to verify your account

Subject: PayPal Flagged Account
From: "service" <service@paypal.com>

At PayPal, we want to increase your security and comfort level with every transaction. From our Buyer and Seller Protection Policies to our Verification and Reputation systems, we'll help to keep you safe

We recently noticed an attempt to log in to your PayPal account from a foreign IP address and we have reason to belive that your account was used by a third party without your authorization. If you recently accessed your account while traveling, the unusual log in attempts may have been initiated by you. Therefore, if you are the rightful account holder, click on the link below to log into your account and follow the instructions.
https://www.paypal.com/cgi-bin/webscr?cmd=3D_login-run

If you choose to ignore our request, you leave us no choice but to temporarily suspend your account.

If you received this notice and you are not the authorized account holder, please be aware that it is in violation of PayPal policy ...

Figure 1. Subtly different exploit emails. Notice the similarity in the theme of the emails and the slightly different terms applied to achieve the same goal.

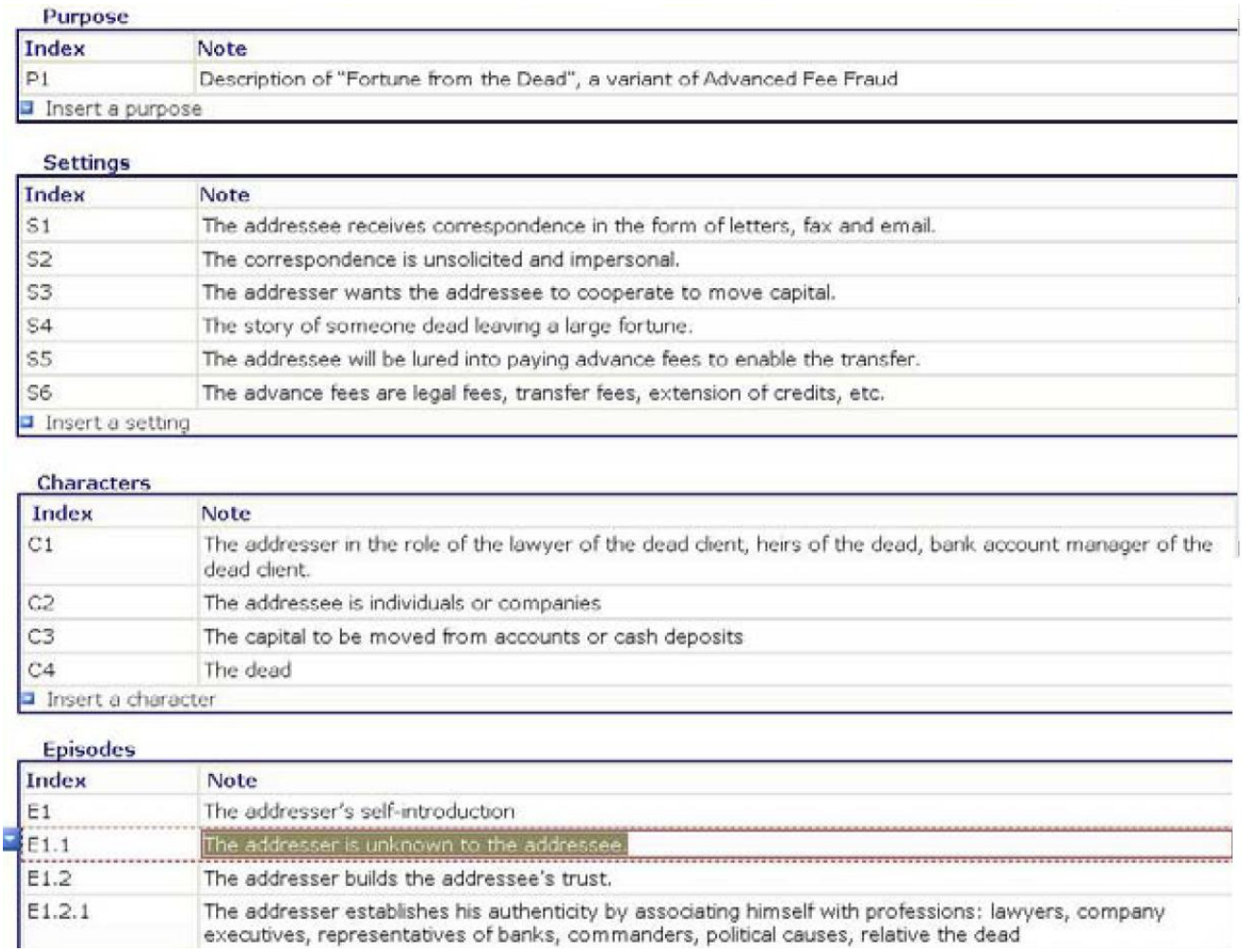
Intrusion Detection Systems exist that can detect malicious traffic by a number of digital signature mechanisms, such as string-based signatures (Signature-based IDS) or network traffic anomalies (Anomaly-based IDS). Intrusion Detection is a field that has been around for some time, and Intrusion Detections systems still generally adhere to the models and concepts discussed in Dorothy Denning's "An Intrusion-Detection Model" (1986) and Mukherjee, Heberlein, and Levitt's "Network Intrusion Detection" (1994). These two papers are seminal to the practice of computer forensics using detection methodology. They outline the basic mechanisms of Network Intrusion Detection, where the Intrusion Detection System is basically a part of one's network, and it sits between systems, or between networks, and reads some part of the traffic coming through,

studying some facets to make a determination on whether or not that traffic is malicious, and then performing some action in response/mitigation.

Signature-based intrusion detection/prevention, including spam filtering, uses digital strings to detect attacks as they occur. This can be used to some success in detecting strings and characteristic bits of code relating very specifically to protocol-specific packet-level text, such as the bytecode for a specific exploit or the specific flags of a packet. It can be used to judge just about everything about exploitive mails, but if it were used on its own to detect literal strings in email text, the approach would either fall short or the signature list would grow so much as to negatively impact performance. Because social engineering can be flexible about all of the strings in the email, this approach can only delay the attackers and can be prone to false positives. Figure 1 demonstrates this issue by showing two nearly identical emails that attempt to exploit the user using nearly identical stories and approaches, but with slight variants that may require a standard signature-based IDS to change its rule set radically.

Anomaly-based intrusion detection/prevention uses traffic analysis to try and recognize abnormal patterns in traffic. An example of such a system is MINDS, the Minnesota Intrusion Detection System (Ertoz, Eilertson, Lazarevic, Tan, Srivastava, Kumar, and Dokas). However, while it may be possible to detect large swings in email traffic due to a high-profile distribution, not all social engineers are going to utilize a wide and fast distribution. Thus, as long as the traffic remains under the radar of anomaly, it is not detected. There are other issues with anomaly-based IDS, but that is the major flaw that applies to this problem set, the fact that a subset of social engineering emails rely on the appearance of network normalcy.

Natural Language Processing is the science of teaching computing systems to interpret the context of written human language. UMBC's **OntoSem** project is the large-scale NLP system (Nirenburg, McShane, and Beale, "Operative Strategies in Ontological Semantics") that EBIDS will interact with through "DEKADE II" (English), a Java API. The purpose of such a system is to be able to translate context from natural human language, as it is written, to something that a computer system can recognize. It achieves this purpose through a multi-layered approach. There is a semantic analyzer that recognizes the definitional details of words, including what groups the word belongs to inside of the ontology. The ontology itself is the knowledge base of categorization that all words fit into, defined through data acquisition phases. The referential analyzer attempts to recognize links between words, for example when a pronoun refers to a previously mentioned proper noun. There are other components of the system, but within the scope of this project, only these first three elements are necessary. NLP is traditionally used for things like large-scale email analysis or document analysis. A knowledge base has to be built up front in order for it to be effective, but that is a one-time cost and maintenance updates are trivial.



Purpose

Index	Note
P1	Description of "Fortune from the Dead", a variant of Advanced Fee Fraud
Insert a purpose	

Settings

Index	Note
S1	The addressee receives correspondence in the form of letters, fax and email.
S2	The correspondence is unsolicited and impersonal.
S3	The addresser wants the addressee to cooperate to move capital.
S4	The story of someone dead leaving a large fortune.
S5	The addressee will be lured into paying advance fees to enable the transfer.
S6	The advance fees are legal fees, transfer fees, extension of credits, etc.
Insert a setting	

Characters

Index	Note
C1	The addresser in the role of the lawyer of the dead client, heirs of the dead, bank account manager of the dead client.
C2	The addressee is individuals or companies
C3	The capital to be moved from accounts or cash deposits
C4	The dead
Insert a character	

Episodes

Index	Note
E1	The addresser's self-introduction
E1.1	The addresser is unknown to the addressee.
E1.2	The addresser builds the addressee's trust.
E1.2.1	The addresser establishes his authenticity by associating himself with professions: lawyers, company executives, representatives of banks, commanders, political causes, relative the dead

Figure 2. An F. F. Poirot Ontological Signature

Ontology-based phishing detection systems exist, such as the F. F. Poirot project, which relies on a pre-built database of terms and definitions to detect phishing attacks against financial institutions in several different languages over email. However, this project is of a very narrow focus and may be difficult to expand. It works by having users build a database of ontological knowledge about phishing attacks. It then recognizes the knowledge and looks in depth at the email text, looking specifically to recognize certain tendencies, in some cases the very elaborate tall tales woven by Social Engineering attacks by focusing on high level concepts such as "the addressee will be lured into paying advance fees to enable the transfer", rather than the underlying text. There are a few problems with the ways in which the rules here are built. Firstly, although the signature developed becomes more accurate when built to the very rigid specifications of F. F. Poirot Rules (see Figure 2), it also makes their rules less flexible and hard to update. Simply stated, it does not obscure the problem text enough for social engineering attacks, still requiring a huge signature set. Another issue is that the data bank here is built solely around phishing, whereas the OntoSem Ontology can be updated comfortably to handle phishing knowledge, as well as other types of Social Engineering, such as targeted impersonation attacks and worm spread emails. That capability of F. F.

Poirot remains to be seen, since it is not publicly available, so it may also have that capability, but the project has no stated purpose to transcend this niche of social engineering limited to phishing. There are publicly available research papers which outline the ways in which the system was developed and its signatures developed. These include: "Knowledge-based Information Extraction: A Case Study of Recognizing Emails of Nigerian Frauds" by Gao and Zhao, "Knowledge-based Information Extraction: A Case Study of Recognizing Emails of Nigerian Frauds" by Kerremans, Tang, Temmerman, and Zhao (the best reference on the whole system), and "Engineering an Ontology of Financial Securities Fraud" by Zhao, Kingston, Kerremans, Coppens, Verlinden, Temmerman, and Meersman. In these papers, it is laid out how F. F. Poirot's signatures begin with stories and generalize to pick up specific cues from within the stories, such as "Foreign Royalty presenting an opportunity." As was found out in testing EBIDS and its competitors, many fraud emails do not involve elaborate stories anymore. In fact, only a very small number of emails had a deception any more elaborate than claiming to be from a financial institution.

In the realm of signature development for an Intrusion Detection System, the approach used by F. F. Poirot is very counter-intuitive. For example, when addressing a problem with a very specific digital signature for a complex exploit, perhaps something that takes advantage of a buffer overflow vulnerability on a certain line in a very specific piece of code, then it is a good idea to make the electronic signature used to detect it as specific as possible, since it is unlikely that this exploit will change forms, so it is a good idea in that case to rigidly define a lengthy signature. However, if a class of highly variable attacks are all coming out against a similar target, which is the case with social engineering, then the signature should be written to be as general as possible within the bounds of acceptable false positives, so that signatures do not have to be constantly created and modified as often as the attacks, since that is largely infeasible and is, at best, catching one's tail. The proper approach is generalization, using one stone to kill two hundred birds and *possibly* an innocent chipmunk or two in doing so. To elaborate, using a single signature to catch the majority of attacks against a network is worth a small amount of innocuous traffic being flagged as harmful and relying on responsible analysis to determine if the traffic was indeed malicious or not.

EBIDS-SENLP is an Email-Based Intrusion Detection System to detect Social Engineering using Natural Language Processing. It uses OntoSem to analyze plain text in email to determine whether or not the language used is indicative of Social Engineering. It mixes the ideas of ontology-based social engineering detection with signature-based IDS in a new and unique way. It is intended to be more generally defined to detect all forms of social engineering on a broad scale, and its basis in OntoSem's library of terms makes it potentially much more powerful than F. F. Poirot, since OntoSem can hold a nearly unlimited number of terms. Although relegated to English Language and email for now, there may be efforts to expand OntoSem that can expand the capability of EBIDS in the long run. The project transcends normal signature-based IDS, because instead of looking only for literal strings, the EBIDS rules can be written to look for generalized contexts and themes, which will then act as equivalence set names that are produced by OntoSem. Thus, instead of matching on

“send us your account information”, EBIDS can match on the concept of “request for personal information”, which may refer to any number of similarly phrased terms, which would be defined in OntoSem for a one-time-cost up front with slight tweaking time in the middle. The entire concept of EBIDS is to use OntoSem’s processing constructs to create equivalence sets of strings that, when used together, constitute a theme of social engineering’s flavor of user manipulation. In comparison to the very strict and rigid definitions of F. F. Poirot’s rules, which include cues such as “Foreign Royalty presenting an opportunity,” EBIDS generalizes even that concept more to “Anyone presents a financial opportunity,” because not every phishing scam that promises money uses foreign royalty, so when an email that offers free money as part of a bogus internet offer from a reputable company comes through, Poirot will either miss it or require a second rule be written, but EBIDS will identify it. Part of the focus of EBIDS is to keep the signature set small and sophisticated, because standard IDS rule sets can often become large and unmanageable, and natural language is a very large rule set. The logical conclusion of this line of thought is to use a system that already categorizes the English language, OntoSem, to create a very small set of focused signatures that are easily recognizable and easily modifiable, in an attempt to “keep up with” and adapt to the readily changing medium of electronic social engineering, using the tenants of standard signature-based IDS to write effective signatures.

Chapter 2

Email-Based Intrusion Detection System to find Social Engineering using Natural Language Processing

2.1 Concept

As social engineering is a relatively difficult thing to defend against, the question arises of whether or not it can be defended with technology at all, and then if so, does there exist a technology which can be applied to perform the task. Social engineering is a unique problem in the scope of computer security, because the onus of sophistication does not lie necessarily within a code exploit or machine solution. Rather, it lies with the language and variability thereof in order to make a social engineering attack effective and adaptable. So, in order to target a social engineering attack, one can either doggedly create literal signatures out of the exploits and web links used in certain attacks, updating frequently, such as a Symantec solution in their Norton AV software, or one can focus on the other element, the one that varies far more except in its tone and theme, the natural language itself. By focusing on the recurring themes of deception within the email, EBIDS should be able to discover new and emerging attacks, because even though English Language can vary incredibly, the types of deceptions used to exploit the human psyche are pretty much defined and well-known. Thus, the scientific leap that EBIDS takes is to use a natural language processing system to perform the task of providing the thematic insight to the system to detect when deceptions are being performed over email. OntoSem was not purpose-built for this, but it could be expanded and tuned to incorporate it. Thus, the problem exists and, arguably, the technology to provide a solution.

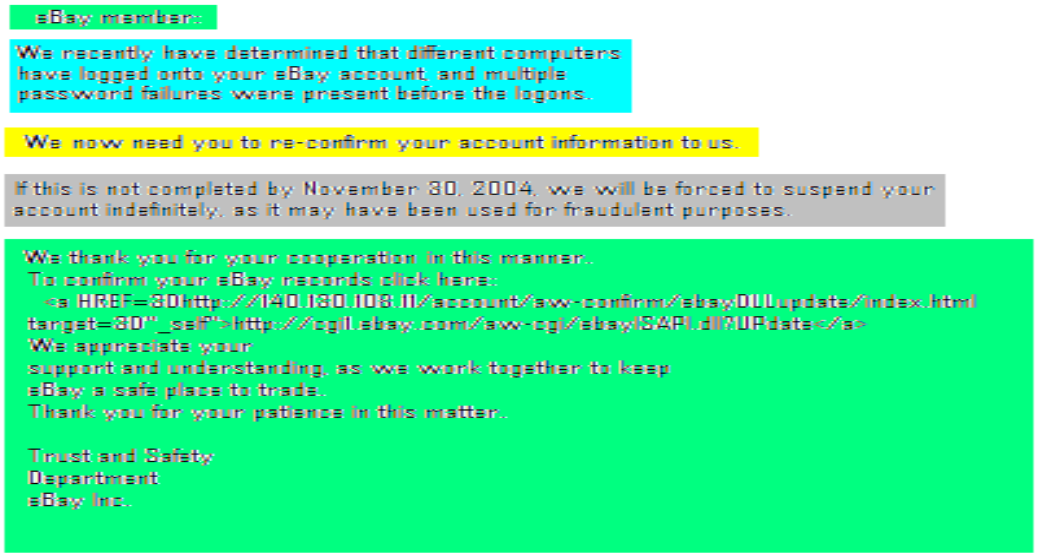


Figure 3. A social engineering email, highlighted by recognizable concepts.

Specifically, the problem that this system attempts to solve is one of recognizing concepts in email text. Figure 3 on the previous page shows an example of a fraudulent email indicative of a social engineering attack. Separate sections of the deception are highlighted differently. The text highlighted in green signifies the more "official" looking corporate parts of this email. The language included is meant to suggest that the email is an official corporate email from eBay regarding the target's eBay account. Highlighted in blue is text that suggests that the user's account has been compromised by an outside, probably unauthorized, user, to give the user a reason to acquiesce to their request, specifically to help eBay help the user supposedly. The text in yellow represents a request for account information, which is being asked for shamelessly here. More sophisticated attacks have more subtle and indirect ways to ask for information, for example by suggesting a login as a necessary go-between for some greater end. The text highlighted in grey above is a significant psychological trigger. It tells the user that there is a deadline, presumably not long after the email was sent, by which they must comply, lest some denial or limitation of their service be experienced. This is a tactic often employed in social engineering to get the victim to respond quickly before they've had time to think it through. The concept behind EBIDS is for it to focus on the concepts represented in the colored boxes, not the text or exploits themselves. For example, an EBIDS detection rule can be seen below in Figure 4. The format may be confusing in the figure, but focusing on the left-most field on each line, one can see the concepts mentioned above: a corporation name, some consequences suggested as an alternative to acquiescence to the information request, some text about account compromise, and the information request itself.

```
START Corporate Deception Rule 1 - Account Compromise
company name::SEM::CORPORATION::1
false consequences::SEM::THREAT-DOS::1
compromise event::SEM::DECEPTION-INTRUSION::1
request info::SEM::ACTION-INFO_REQUEST::1
END
```

Figure 4. An EBIDS detection rule.

OntoSem has the capability to read raw text, evaluate the meanings and relations within sentence structure, and perform many wondrous functions based upon that knowledge. EBIDS will only use a subset of this functionality, and it will do so completely within the bounds of the DEKADE II Java API. DEKADE II is a project intended directly to provide a safe and easy interface to OntoSem's computational knowledge, and in one of the most striking advances of DEKADE, you can access it through a Java API with very few method invocations. This therefore led to the decision to code EBIDS in Java, although there are plenty of other great reasons to do so, considering Java's extensive standard use in academia and industry, along with its considerable portability across operating systems and ease of use within object-oriented design. There was initially a plan to use Perl to provide some scripting language finesse in making a better presentation to the end user, but that can be trivially added at any time, so the decision was made to leave Perl out completely and just use Java for

the time being. Perl was used to write execution and testing scripts, but it is not a part of the system itself.

Much of the initial coding of the system was to parse email and present the natural language that is the message text. EBIDS reads each email in as a text file. It would not take a huge leap to expand its functionality to read the email from pcap, but since it can be proven that such things are readily accessible and done, the decision to just use plain text email files was made. It was also much easier to find corpuses online which presented the emails in text file format. The back end of the program is a simple text parser that feeds OntoSem with text through the DEKADE API, and then reads the XML results, running a simple string match algorithm to determine if the requested social engineering context was detected. When testing, it was determined that OntoSem cannot, in its current state, provide the equivalence matrix needed to identify terms as being associated with certain deceptions or exploits, so as a temporary fix, the equivalence sets for detection will be written directly into the underlying Java code of the system (hard-coded), but only because Dr. Sergei Nirenburg has shown repeatedly in his research (Nirenburg and Mahesh, "A Situated Ontology for Practical NLP", and Nirenburg, Raskin, and Sheremetyeva, "Lexical Acquisition") that data acquisition, although a lengthy process, can be used to tailor OntoSem for several varied purposes, and the use that EBIDS needs is certainly within its domain. The hard-coding is certainly not a desirable long-term solution, as the data set is likely to need periodic updates, and not only is it much easier to do this with OntoSem, but as the ontology grows, it will quickly outgrow EBIDS, but OntoSem's knowledge base could easily support it. It was also discovered very early on that using OntoSem to analyze email text took a very long time, so the initial goal of using an email corpus from the Enron scandal, which was gigantic, was abandoned in favor of a much smaller corpus. The end result was to be a set of less than 100 emails, because it would simply be extremely impractical to put that strain on OntoSem. Future efforts, however, would certainly encourage scaled-up testing of the system with a large corpus, after data acquisition. In this case, because the equivalence sets were built into the project itself, the connection to OntoSem is bypassed and the testing corpus can be scaled up to over 500 emails, making for a more robust test bed. It should be noted that EBIDS has been written to perform as it was initially intended, through interaction with OntoSem, but that the hard-coded equivalence sets with avoidance of OntoSem is a functional workaround that can be taken out after OntoSem has been put through a lengthy data acquisition process. That process is only lengthy once, since it currently has no concept whatsoever of this data type, but expanding/adapting it is considerably quicker.

2.2 Hardware

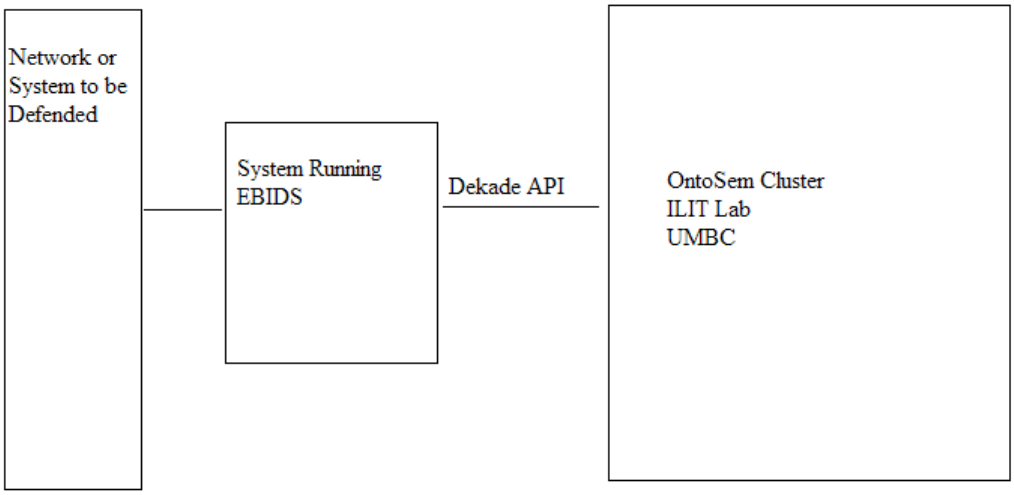


Figure 5. Simple Hardware Diagram

The hardware for this project is largely in the ILIT lab, which is OntoSem and the cluster which hosts it. That system communicates with the system that runs EBIDS, and presumably, that system will be set up to talk to the network that is to be defended. For now, the system simply runs on one laptop, but it could easily be installed and run on any machine. If this system were to run in a production environment, the OntoSem component would either have to be productized and run on a production cluster or would have to be somehow replicated to run without dependency on that specific laboratory, because otherwise it is not scalable. However, the diagram in Figure 5 is how it is being run currently to prove the concept.

2.3 Software Specs of EBIDS System (not including OntoSem)

EBIDS, for the moment, is set up to run on an MS Windows System Environment. Because it was written in Java, with very minor code tweaks, it could be made to run on a Linux or Solaris environment. It is currently running on Java 1.5, and that is all that it has been built and tested on, so that is the recommended version for future distributions. For the evaluation of EBIDS, two current free detection systems have been downloaded and installed, Snort 2.6 and Spamassassin 3.1.6, along with Winpcap 3.1 for network capture capabilities. Snort is being run with an open source rule set from Bleeding Snort, an industry standard rule developer, current to early 2006, as well as the Snort standard rule set, most recent to the date of this research. To run the testing on Snort, Apache 2.2 had to be employed on a separate testing system to pass the test corpus over a pcap process, since Snort cannot run with plain text files as input unless they are being passed over the network. EBIDS is reliant on the DEKADE II Java API being installed and available.

2.4 EBIDS Architecture

EBIDS consists of five components, a front end reader, a text parser, an OntoSem interface module, a rule builder and detection mechanism, and an output mechanism. It is a very simplistic model with the onus of complexity on the rules themselves and the hard-coded equivalence matrix, which will eventually be removed when OntoSem acquires the knowledge to make it obsolete, which will place all of the complexity squarely on the rule set. In the long term, that is likely to be an undesirable situation because it should not be necessary to know OntoSem intimately to use this program, so another area of future work this problem brings up is a mechanism for adding signatures that abstracts the complexity from the end user.

The Front End Reader of the system is very simplistic. All it does is read the file specified by the user and forward it on to the Text Parser. The driver passes the name of the file(s) to be processed as emails to the text parser by invoking an instance of the Email Parser class, which both reads the email and performs preliminary parsing steps, which is technically part of the next component, the text parser.

The Text Parser is an extremely complicated and distributed piece of code that is scientifically insignificant for this project. It simply parses out the message text and removes characters that would make OntoSem choke. It is not perfect in this effect but was tuned to at least accept a majority of the training corpus. Email Text Parsers exist out there, so this piece of the program, just like the front-end reader, is expendable and insignificant to the scope and aim of the research. It is written into EBIDS as the Email Parser and Email Parts classes, which simply strip out all html tags, the various expected email fields, and things such as newline characters in order to pass on only raw natural language from the email body. The essentials of how and why code was stripped from the emails can be attributed to familiarity with the email protocol, as noted in Volumes 1-3 of TCP/IP Illustrated (Stevens and Wright).

The OntoSem Interface Module (OSIM) is one of the significant pieces of this program. After the text parser forwards on the email message body that was readable, the OSIM uses calls to the DEKADE API to connect to OntoSem and pass it the email text (DEKADE provides simplified method invocations to utilize OntoSem, including the capabilities to connect and request specific analysis of given text), which is then processed and fed back to EBIDS in this module. OSIM them writes the XML results from OntoSem to file for later processing and exits gracefully. OntoSem actually becomes the key component in this chain at this point. It is the part of the OSIM that is the key scientific advance for the project. It is almost overkill at this point, since OntoSem is processing the data for every single possible use for OntoSem, rather than focusing on one specific thing. Future research could write an EBIDS-specific interface with the system to improve performance in this aspect. But, in most general terms, this component takes the email text and generates an XML file. Currently, this step take about 2 seconds per email to parse out the text body, but it can take up to 20 or 30 minutes, depending on the email, to actually process in OntoSem, especially if

Referential analysis is also required. Thus, this performance bottleneck represents a vital part of the system that requires improvement before it can enter a production phase.

The Rule Builder and Detection Mechanism (RBDM) is the other significant portion of EBIDS. Its job is to build a collection of detection rules from a specific text file. After doing so, it then runs the rules from that file against the literal email message text and the OntoSem Analyzer XML files, both Semantic and Referential, depending on which file the signature specifies. Again, the onus of complexity was passed on to these very specialized rules. They include multiple lines to build a multi-signature rule, spaces for comments, places to specify what string should be located, and in a step toward targeted abstraction, a field to specify whether the signature line for a rule is to be matched against the email text or either the semantic or referential results, the latter two of which is how EBIDS uses OntoSem to build a virtual equivalence set, by specifying high-level definitional concepts to match against the meanings behind the words instead of literal strings themselves. Refer to Figure 4 for the exact rule format. In the future, it would be optimal to house the rules in a database, but again, simplicity and accuracy were the chief concerns, rather than storage or speed. Once a rule has fired (based on a %-threshold of signature lines within a rule), it is forwarded to the output mechanism. The algorithm that guides detection is less focused on speed performance than it is on simplicity and accuracy. It relies on Java's plain-text matching procedures from the java.lang.String class. For future optimization, it is not out of the question to use one of the more advanced string matching algorithms for this purpose, especially if EBIDS is to be run on larger networks. Basically, the Detection algorithm breaks down as follows:

1. Given the literal text string that contains the message body and the names of the XML files containing the output of OntoSem's Semantic and Referential Analyzers, copy the entire contents of those three strings into large String objects.
2. Iterate through the rules array. For each rule, iterate through each signature in the rule by doing the following.
    a. Determine which signature type (Semantic, Referential, or Literal) it is and match the string associated with it (either a literal string or an OntoSem ontological concept string, i.e. "INFORMATION") by testing the appropriate string for whether or not that signature string is a substring. In special cases, the subroutine chain match is called, which does the following.
        i. Since the Semantic or Referential concept requires several strings within an equivalence set to be matched, for example if the string "ACCOUNT_COMPROMISE_SET_1_OF_2" was to trigger, it may need a few strings from other sets to appear. So, iterating over the range specified in that string, in this case the requirement is two sets having matches, so attempt to match both "ACCOUNT_COMPROMISE_SET_1_OF_2" and

"ACCOUNT_COMPROMISE_SET_2_OF_2", and if both match, then alert, and if one does not match, then do not alert.

b. If a match has been established, add the weight of this signature to the total rule weight and move on.

3. For that rule, if the total rule weight of the match outweighs the threshold for that rule, then alert on that particular rule, adding it to a string of the rules that fired, and repeat step 2 until all rules have been checked.
4. If no rule fired, return the string "clean". Else, return a string built up from the rules that fired successfully.

The Output Mechanism is also very simple. It writes to a text file the email details, a summary line of which rules fired, and a snippet of text from the email itself to better show the user at first glance what the email is saying.

```
<tmr>

<sentence>
". Attached is the information you have requested."
</sentence>

<concept name="INFORMATION-243" type="OBJECT">
      <attribute type="textpointer" value="INFORMATION"/>
      <attribute type="word-num" value="4"/>
      <attribute type="FROM-SENSE" value="INFORMATION-N1"/>
      <attribute type="TEXT" value="information"/>
</concept>
<concept name="HUMAN-244" type="OBJECT">
      <attribute type="textpointer" value="YOU"/>
      <attribute type="word-num" value="5"/>
      <attribute type="FROM-SENSE" value="YOU-N1"/>
      <relation type="RELATION" target="INFORMATION-243"/>
      <attribute type="SAME-SCORE" value="(HUMAN 0 YOU-N1)"/>
      <attribute type="TEXT" value="the information you"/>
</concept>
</tmr>

<tmr>
```

Figure 6. Sample OntoSem XML Output for the sentence, "Attached is the information you have requested."

The two key components of the system, the OSIM and RBDM, work hand-in-hand as follows, beginning with the OSIM. It fully processes the text, writing its semantic and referential knowledge of the sentences passed to it in ontological terms, represented as XML tags. Each sentence is processed within "tmr" tags, which encapsulate the literal sentence itself, followed by the ontological breakdown of each

phrase or term within the sentence that is recognized by OntoSem. It is here that the abstract step of creating equivalence sets for literal natural language strings can be made. Since OntoSem recognizes the literal and creates XML data that represents that term's role(s) in the Ontology of the overall system, the EBIDS detection rules can be set to look for those ontological concepts by string matching the XML results in the Detection Mechanism (described below), thus matching on a theme rather than a literal string, as seen in Figure 6 if you were to match on either of the equivalent terms "data" or "information" by string matching the XML Attribute "INFORMATION". And to build context, terms can be broken down to ensure that greater complexity exists than just one of a set of terms.

There may need to be a match of one of a set of terms for some number of terms, making the rules more strict and accurate. For example, for the rule example from Figure 4, the Account Compromise rule, examining just the line about "compromise event", the equivalence set would look theoretically similar to an example set of {{account, access}, {compromise, hijack, unauthorized, password failure, fraud, discrepancy, discrepancies}}, where the subsets represent the fact that for the set to be considered matched, a term from each of its subsets must be matched. Thus, a sentence like "Your account shows signs of unauthorized access" would match, while a sentence like "Your account may have been hacked" would not, since only the term "account" is matched, and nothing from the other subset matches. In order to achieve a similar effect in OntoSem, such sets could be defined with numerical definitions. So, if "COMPROMISE EVENT" was an OntoSem tag, it could be broken into "COMPROMISE EVENT-1" and "COMPROMISE EVENT-2", where both would have to be matched to match the original concept, and if any single term was a dead giveaway for a compromise event on its own, it could be added to both sets, thus causing both to match. Again, all of this can be added up front in a lengthy data acquisition process, but after that price is paid in full once, it is later paid less drastically as small modifications become necessity to advance with trends.

2.5 EBIDS Signature Set

A Signature-Based IDS is only as good as the rules that define how it alerts. This is one of the subtler problems with Intrusion Detection. There is a real art to defining specific and accurate signatures. If the signature is too specific, then it may be too limited to be useful. If it is not specific enough, then the false positive rate associated with that rule may be too high for analysts to bear as end users. Thus, the EBIDS signature set was defined using a number of themes taken directly from The Art of Deception. Some best practices for signature writing were followed from the entire text of Intrusion Signatures and Analysis, which is less a guideline for signature writing as a whole as it is a snapshot of effective signatures for that timeframe, but good practices in signature writing can still be gleaned from following the flow of the text.

The signature development, similar to the system as a whole, was partially defined using the training half of the Nazario phishing corpus. It was decided to go this route so that the rules could target some of the specific language that might be used in

the testing half of the corpus, since the emails from both corpuses were from around the same time period and were likely to overlap. Since the English language is so vast and these attacks vary in terminology to suit trends, to avoid the risk of a higher-than-necessary miss rate, the signatures were designed thusly, also with the hope of preserving time, which was limited to begin with.

```
###
OBIDS RULES FILE
Rules in the following format:
START <RULE DESC>
comments::<MATCHTYPE>::<CONCEPT>::<WEIGHT>
...
END

All comments not involved in a rule should have a line of "###",
followed by the comments, followed by another line of only "###"

Blank lines will have no net effect.

<RULE DESC> is the name/rule description that will show up in the alerts.
<MATCHTYPE> is either SEM (semantic concept), REF (referential
concept), or LIT (literal string).
<WEIGHT> is the numerical importance of the line in a rule on a whole to the rule.
###

START Corporate Deception Rule 1 - Account Compromise
company name::SEM::CORPORATION::1
false consequences::SEM::THREAT-DOS::1
compromise event::SEM::DECEPTION-INTRUSION::1
request info::SEM::ACTION-INFO_REQUEST::1
END

START Corporate Deception Rule 2 - Financial Opportunity
company name::SEM::CORPORATION::1
false congratulation::SEM::CONGRATULATION::1
financial aspect::SEM::MONEY::1
request info::SEM::ACTION-INFO_REQUEST::1
END

START Corporate Deception Rule 3 - Change/Update to Account
company name::SEM::CORPORATION::1
account change::SEM::ACCOUNT_CHANGE::1
false consequences::SEM::THREAT-DOS::1
request info::SEM::ACTION-INFO_REQUEST::1
END

START Corporate Deception Rule 4 - Opportunity
company name::SEM::CORPORATION::1
request info::SEM::ACTION-INFO_REQUEST::1
false congratulation::SEM::CONGRATULATION::1
financial aspect::SEM::CREDIT_CARD::1
END
```

Figure 7. The EBIDS Rule Set

The rules were defined to allow for multiple lines of definition, so that the signatures involved could really be drilled down to get specific, since the individual matches imply a level of abstraction. One would not want to match a rule solely on an "information request" signature, which asks for language indicative of asking someone to click a link or email their account information. Even though that is the one common signature found in all of the rules in Figure 7, that alone is not enough to flag an email as malicious, since some malicious emails will not explicitly request information. Also, there is reason to believe that some traffic may match part of a rule instead of the whole

thing, so a threshold scheme was utilized, with the weights for each signature within a rule signified as the last field in the line. There are also times where one might want to utilize the referential analysis from OntoSem, or to match a literal string, rather than just a semantic one. In the below example, taken from the original test set for system development, one can see in the second field in each line exactly how to specify what aspect of the system gets used in matching.

```
START Cheater Rule
Congratulations::sem::CONGRATULATION::9
A genuine thank you::sem::THANKS-INTERJECTION::4
Asking the user to look::ref::INVOLUNTARY-VISUAL-EVENT::7
Threshold Test – this literal will never appear::lit::smoochydies::1
END
```

Comments in this rule set are achieved in two ways. Any general comment can be enclosed in "###" delimiters, and any comment for a single-line signature within a rule can be specified in the first field. If there is no comment for a line, just start that line with a "::" delimiter. The second field is the mode of matching for that line: "sem" for Semantic, "ref" from Referential, and "lit" for literal string matching. Semantic matching is most often used, as it is the definitional and contextual meaning for a term within a sentence. Referential may have use to determine if an unspoken relationship exists between two words. Literal string matching capability was more or less an added capability, and it can be useful if there is a definite signature associated with an attack, but it is more or less extra in the scope of this project. The above reasons are also why all of the signatures used to test the program, noted in Figure 7, are all completely dependent on Semantic signatures. Figure 8 shows the literal strings in the underlying equivalence set for each of the conceptual signatures. These sets of literals are literally coded in as a temporary workaround, but eventually, OntoSem would be able to hold this information conceptually if the data was acquired into it in specially defined Ontological branches.

The rules defined in Figure 7 look for phishing emails specifically. The goal of this project is to eventually expand beyond this, but time was a limiting factor, so the rule set had to be focused on the corpus that was available by using the training half. The rules, as they exist for testing, are as follows:

1. Account Compromise – This rule is set up to detect whenever an attacker falsely claims that a user's account or computer may be hacked, compromised, or otherwise in the hands of unauthorized users. It looks for four specific themes in the email.
    a. Language that intimates an account compromise, seen later in Figure 8 in the deceptions_intrusion string equivalence set.
    b. A corporation name to make the entire thing sound official. This is detailed in the corporations string set.
    c. Threat of denial of service (DoS) to the user's account, which may appear as "We will suspend your account until you verify your information." This is obviously the threats_dos set.

    d. Information request, which is any language asking a user to follow a link or reply to the email with their account information, specified below as action_info_request.
2. Financial Opportunity – This rule is set up to detect any time a windfall offer is being sent to the user that specifically deals with monetary value (specifically spelled out in plain language).
    a. Congratulatory language, which usually accompanies financial windfall emails, detailed below in the offer set. This type of language is used to hype the user up to believe that indeed they are being offered an exciting offer.
    b. Money language, dollar amounts, the word "free". This is very basic, but essential to this specific rule. This is found in the money_term set.
    c. Corporation name, as above.
    d. Information request, as above.
3. Change/Update to Account – This rule is set up to detect any time that account verification is being requested due to a change to a user's account. A common justification for requesting a user's info is that it has been lost or their account is somehow being refreshed or modified. This rule aims to catch such activity.
    a. Language that applies that a user's account is being updated or changed in some fashion. This is found in the acct_change set.
    b. Corporation name, as above.
    c. Information request, as above.
    d. Threat of denial of service, as in the compromise mail, because most times when someone is requesting an information update, they will suggest that there is a time deadline before the account is deactivated.
4. Opportunity – This rule is set to detect all windfall offers that do not strictly refer to money. This could include vacation offers, credit card offers, and stock tips. For the purposes of this experiment, it was mainly focused on credit cards, but it can be expanded/changed later.
    a. Language that refers to credit cards, as seen below in the credit_card_terms set.
    b. Congratulatory language, as above.
    c. Corporation name, as above.
    d. Information request, as above.

These rules were rigidly defined around the training corpus, based on psychological triggers mentioned in The Art of Deception, which is further detailed later in this section.

```
public String [] corporations = {"eBay", "paypal", "PayPal", "Paypal", "Suntrust", "SunTrust", "Washington Mutual", "wamu",
"U.S. Bank", "Huntington Bank", "First PREMIER Bank", "Citizens Bank", "Keybank"};

public String [][] threats_dos = {{"limit", "restrict", "suspend", "block", "allowed", "avoid", "cancel", "suspension"}, {"access",
"rights", "account", "fees"}};

public String [][] deceptions_intrusion = {{"account", "access"}, {"compromise", "hijack", "unauthorized", "passwords
failure", "fraud", "discrepancy", "discrepancies", "password failure"}};

public String [][] action_info_request = {{"ACTIVATE", "activate", "Activate", "confirm", "verify", "verification", "sign in",
"signing in", "log in", "Log in", "click", "go", "Go", "Click", "access", "Prove"}, {"information", "account", "link"}};

public String [] money_term = {" $", "dollars", "bucks", "free", "Free"};

public String [][] credit_card_terms = {{"Visa", "Mastercard", "Master Card", "Master card", "Discover", "American
Express", "American express"}, {"Gold", "Platinum", "Black", "credit", "Credit"}, {"Card", "card", "account", "Account"}};

public String [][] acct_change = {{"switched", "changed", "updated", "switch", "change", "update", "switching", "changing",
"updating"}, {"account", "security", "credit card", "debit card", "card information"}};

public String [][] offer = {{"pleased", "excited", "congratulate", "happy", "congratulations", "Congratulations", "Congrats",
"congrats"}, {"offer", "opportunity", "inform", "congratulate", "congratulations", "Congratulations", "Congrats", "congrats"}};
```

Figure 8. Equivalence Set that defines the signature sets

The concepts used from The Art of Deception include some of the following, citing specific psychological triggers. These triggers are cited on pages 247 through 249, in Chapter 15 of the book.

1. "People have a tendency to comply when a request is made by a person in authority. … a person can be convinced to comply with a request if he or she believes the requestor is a person in authority or a person who is authorized to make such a request." This drove the decision to look for corporation names is some of the rules, since the attacker will often pose as a legitimate company that an end user is likely to have a financial account with, such as Paypal. This is also why the idea of an account denial of service threat becomes more believable from a user standpoint, and is thus why that set of ideas is included in many attacks, and thusly the signature set. The psychological trigger referred to here is Authority.
2. "We may automatically comply with a request when we have been given or promised something of value. … When someone has done something for you, you feel an inclination to reciprocate. This strong tendency to reciprocate exists even in situations where the person receiving the gift hasn't asked for it." This is the psychological trigger known as Reciprocation. This takes many forms in attacks, and is involved with all of the EBIDS rules. The premise of phishing emails is nearly always going to be an information request of some kind, based on the false pretext that the requestor is reliable and providing some benefit to the end user. So, the "opportunity" rules in the signature set use false congratulations and the intimation that some great gift is being given to or won by the user, even though that user most likely didn't request it, and all the social engineer is asking in return is for the user's information, possibly to help along in the process of getting them their gift faster. That breaks down to: "I've given you a gift, so now

please give me something." It is much the same with the "account compromise" rule. Here, the social engineer is using the pretext that he/she has found the user has possibly been hacked. That phony information is the perceived value provided to the user, with the reciprocation step being whatever the attacker asks for in return. In the "change/update to account" rule, there are a number of triggers used. This rule is very generalized, but some of the pretenses cited as false benefits to the user include: adding new features to one's online account, theft/fraud prevention, an inability to find the user's proper records. Whatever the "reason" for the email, the perceived value is that the organization is now asking for verification in return for what they've done already.

3. "People have the tendency to comply after having made a public commitment or endorsement for a cause." This is what is called Consistency. The "change/update to account" rule was designed for this reason, and this also has to do with the threat of DoS being a potent and believable threat, because the attacker will falsely cite rules or terms of one's account as being the reason for discrepancy, and thus asking the user to verify their information in order to comply with rules or terms that they already have committed to, publicly or not. The idea that one's account can be suspended or limited because of this is very troubling, and most people will do what they can to stand by the guidelines that they may have signed up for, even if they did not read the terms of service when they created their account.
4. "People have the tendency to comply when doing so appears to be in line with what others are doing. The action of others is accepted as validatin that the behavior in question is the correct and appropriate action." Social Validation is a strong trigger, and it appears in phishing emails in a number of ways. In some of the windfall/stock tips emails, it is mentioned that smart investors are getting on board with these requests. Also, perhaps more poignantly, in several emails, a phony security audit is cited as the reason for the verification, saying that the user was randomly selected, which implies that other users are selected and that it happens all the time. This trigger did not explicitly guide rule creation for this testing, but it is significant, because it shows up in the testing corpus later.
5. "People have the tendency to comply when it is believed that the object sought is in short supply and others are competing for it, or that it is available only for a short period of time." This is the concept of Scarcity. This is an important trigger that Mitnick does not go into great enough detail on. The concept of limited time or supply is a trick that salesmen often exploit. The main reason to do so is not only to have the user believe that whatever it is may be more valuable than it actually is, as Mitnick suggests, but it is also to put the user in the state of mind that they have to make a decision immediately, rather than to think it out. This is a trigger that relies on one of the other triggers having been successful. For example, telling a person that their account will be shut off if they do not comply in general is scary enough, but telling them that it will be shut off if they do not comply within three days makes the emotional shock of losing account access more strong and present in one's mind, since they now have little time to make their decision. This is why one of the key points in Personnel Training is usually to tell people to take their time before responding. It gives them a chance to think

objectively and logically before moving forward. This did not specifically lead to any of the rules used in the EBIDS evaluation, but it could be employed later, since almost all of the emails with a threat of denial of service or extra fees was accompanied by a time deadline. This also appears in other forms. For example, in one particular phishing email, it is noted that a stock tip is only offered to a limited number of people, and that many of the slots to own the stock are filling up, putting the stock in short supply. This is a trigger well worth consideration for signature development, but it was too general, too large a data set, to include for the purposes of this project's evaluation.

The signature set is a complex and significant part of the success of this project, and it was thoroughly and rigidly defined for that purpose, based both on the common psychological triggers that social engineers hope to exploit and the language and terminology familiar to the Nazario phishing corpus. It is because of this familiarity with the material and a knowledge and background in signature analysis and writing, as well as a knowledge of the basic working mechanisms of OntoSem (here bypassed by hard-coded equivalence sets), that EBIDS stands a chance of reaching its performance goals in the testing phase. Signature writing, especially for the use of EBIDS, is not to be taken lightly. If falsely tuned with access to OntoSem's extensive knowledge base, EBIDS could have the potential to overwhelm OntoSem, the user's system, or the analyst who looks over the data. It is vital that a trained professional write these signatures, because this aspect is such a heavy component of a relatively simply designed system. Future iterations of this project should take steps away from needing to know the specifics of the underlying architecture, such as OntoSem's ontological names for terms. A layer of abstraction can benefit users greatly and make EBIDS usable to the general Network Security field.

Chapter 3

Testing and Results: Three Systems and Two Corpuses

3.1 Testing Parameters

The corpus of this project is complexly defined, but defined for specific reasons. The first corpus, the all-known-bad corpus, has been built from a publicly available phishing corpus, compiled by Jose Nazario, PhD. Although publicly available, Jose was contacted for his permission to use this corpus for this specific project, to which he consented and provided extra insight. This corpus, found at http://monkey.org/~jose/phishing/phishing0.mbox , is composed of a large number of phishing emails in plain text. This set was divided in half, one to be defined as the "training" set, and the other to be defined as the "testing" set. The training set is defined with the specific purpose of being a testbed for the definition and primary testing of EBIDS during primary development. The testing set is left completely untouched until the testing phase. It will then be the all-known-bad portion of the system evaluation, and it will be run against Snort (Roesch, "Snort – Lightweight Intrusion Detection System") and SpamAssassin (Schwartz, SpamAssassin) to evaluate their effectiveness. The emails in Figure 1 are examples of social engineering from the Nazario phishing corpus. The all-known-good corpus will be separately defined from a chunk of the Enron Corpus, also publicly available.

The systems will be evaluated on their effectiveness in detection rate, which is how many of the malicious emails were actually detected as malicious, and false positive rate, which is how many of the innocuous emails from the Enron corpus falsely flagged as malicious. The expertise to do this comes from years of Network Security Analysis, following methodologies derived from the following list of publications: Practical UNIX Security (Garfinkel and Spafford), Hacking Exposed: Network Security Secrets and Solutions, Third Edition, 3rd Edition (McClure, Scambray, and Kurtz), Intrusion Detection: An Analyst's Handbook (Northcutt), Security Warrior (Peikari and Chuvakin), Hacker's Challenge 2: Test Your Network Security & Forensic Skills, 2nd Edition (Schiffman, Pennington, Pollino, and O'Donnell), and the TCP/IP Illustrated series (Stevens and Wright). All three systems, EBIDS, Snort, and Spam Assassin, will be tested in this manner. Performance and scalability are not going to be included here, because OntoSem is not set up to perform to the needs of an IDS and although scalability will ultimately be a major concern, the purpose of the EBIDS project is to prove the concept that the approach is feasible. Future work may be done to answer the question of whether or not it can be done within a reasonable time.

3.2 Expected Results

Although the corpus will be flatly divided between known-bad and known-good traffic, there is bound to be a miss rate and false positive rate in all three systems. The expected run on Snort is going to likely be both high miss rate and false positive rate, since Snort is largely not focused on catching phishing attempts. Spam Assassin, the

email filter, is expected to fare much better, since its signature list should include information from at least some of the attempts. Its false positive rate should be rather low. EBIDS is expected to outperform them, since its rules are being defined with the training corpus, so it is custom tailored to this sort of data. Its false positive rate is going to be a chief concern, however, since the signatures are based in natural language, which has such a variety of use that it is difficult to predict in what other context words may be used that end up falsely firing the system. Also, since EBIDS is looking at only the natural language of the email body, exploitive language in the subject line or emails that have no text (but are only images or code) will go unnoticed. EBIDS has a rule threshold in this run of 70%, which means that at least 70% of a rule's matches have to occur for it to alert.

3.3 Testing Methodology

For EBIDS, the code was simply set to loop over email in text files in a directory, reading in each one individually and outputting to a corresponding output file. Because the text parser for html formatting was written from scratch, a small number of the testing corpus was excluded due to exceptions in execution.

For SpamAssassin, no extra steps were necessary to run the tests. The files were simply piped in as input at the command line, and the output piped to similarly named output files. A quick PERL script was written to perform this task en masse.

Snort has no option to analyze plain text files, so in order to get snort to analyze the data, Apache 2.2 was loaded onto a system with the testing corpus loaded on it in order to have it act as a web server. Meanwhile, the laptop which had snort installed on it was taken to another location to utilize another internet connection and connect to the web server, downloading each of the files one by one by clicking links in a simple html page created just for the purpose of hosting the files, while snort ran in detection mode to try and pick out exploits in the plain text. A PERL script was also employed here to very simply create that html file.

All PERL scripts for running the testing code have been included in the appendices, along with the signature sets of all three detection tools.

The total testing corpus was composed of around 230 emails from Jose Nazario's corpus, which were untouched during the training and tuning phase of EBIDS, along with around 300 emails from the Enron data corpus, publicly available at http://www.cs.cmu.edu/~enron/ (Carnegie Mellon University). The Enron data set is assumed to be mostly void of social engineering attempts, so it was cleanly merged with the known bad set to be a test of false positive rate.

3.4 Testing Results

When judging any Intrusion Detection System, one has to consider very carefully a number of performance metrics. As this is highly theoretic work at this point and

more of a proof-of-concept, the focus in testing is on how well the system achieves its goal of accuracy, rather than testing performance numbers. OntoSem has not been optimized to run in real time speeds for this sort of querying in the manner that current industry standard IDS systems have, so time efficiency was not a key factor in choosing the algorithms used for matching, and so even after the signature set's equivalence sets were coded into the program, the algorithms were simply left as they were, focused more on accuracy of results over performance.

The metrics of accuracy usually related to Intrusion Detection Systems involve two very key statistics, the hit rate, which refers to how much of the bad traffic is actually being flagged when it is seen, and the false positive rate, which refers to how many times completely innocuous traffic is being flagged wrongly as being possibly malicious. There is a tradeoff, and it usually varies from signature to signature, based on the threat the signature is set up to mitigate, and that tradeoff is basically what number of false hits is acceptable to the end user for the ability to keep the hit rate high. The rates are related, since they both involve how often alerts are triggered, and best practices with regards to signature writing can minimize false positive rate while compromising hit rate as little as possible. For the end user, these accuracy measures are crucial to how they perform their task. Too low a hit rate, and the user is assuredly missing malicious traffic. Too high a false positive rate, and the user is potentially inundated with false hits, making it harder to find the malicious traffic and wasting the time of the analyst using the tool by following false leads.

EBIDS is being held to another standard in addition to the rest. Because the experiment's signature set is based on string matching natural language, a very realistic fear exists that not only will the number of false positives skyrocket, but also that rules will more often fire when they are not supposed to during correctly firing rules. An example of such a false hit is an instance that occurred during the testing of EBIDS quite frequently, where an email would be formed requesting information verification fraudulently but without mention of account compromise as the cause, but the account compromise would fire in addition to the rule that picks up fraudulent information requests. In that case, it is clear that the account compromise rule should not have fired. However, the converse is a case that is not true. In almost every instance where the account compromise rule fires correctly, there is some request within the email text to update or verify account information, so in those cases when both rules fired, it was not deemed a "partial false positive," unlike the prior case. So, in the below stats, "partial false positive" refers to alerts where a rule fired when it shouldn't have, but the correct rule fired on malicious traffic in addition to the incorrect rule. Because of how extensive the Snort and SpamAssassin signature sets are, it would take too long to analyze when a partial false positive occurred with each correctly flagged malicious email, so that step was avoided altogether.

The fields for each summary are:

1. Number of total emails in the test corpus, which varies because the EBIDS email parser was written by hand, so it could not handle all of

the various html and smtp coding styles of the email corpuses, and SpamAssassin was unable to handle 2 emails as well.

2. Number of emails in the Nazario phishing corpus, which are all known-bad phishing emails.
3. Number of emails from the Enron corpus, which consist of no social engineering emails.
4. Number of total alerts that fired, including both correct alerts and false positives.
5. Number of total alerts that fired on malicious traffic accurately.
6. Number of total alerts that fired on innocuous traffic wrongly (false positives.
7. (EBIDS Only) Number of alerts where rules fired wrongly on malicious traffic, where the correct rule also fired.
8. (EBIDS Only) Number of false positives and incorrect rule fires.
9. Hit Rate, defined as the ratio of accurate alerts to total known-bad emails from the Nazario phishing corpus.
10. False Positive Rate, defined as the ratio of false positives to total innocuous emails.
11. (EBIDS Only) Adjusted False Positive Rate, defined as the ratio of false positives and partial false positives to total emails.

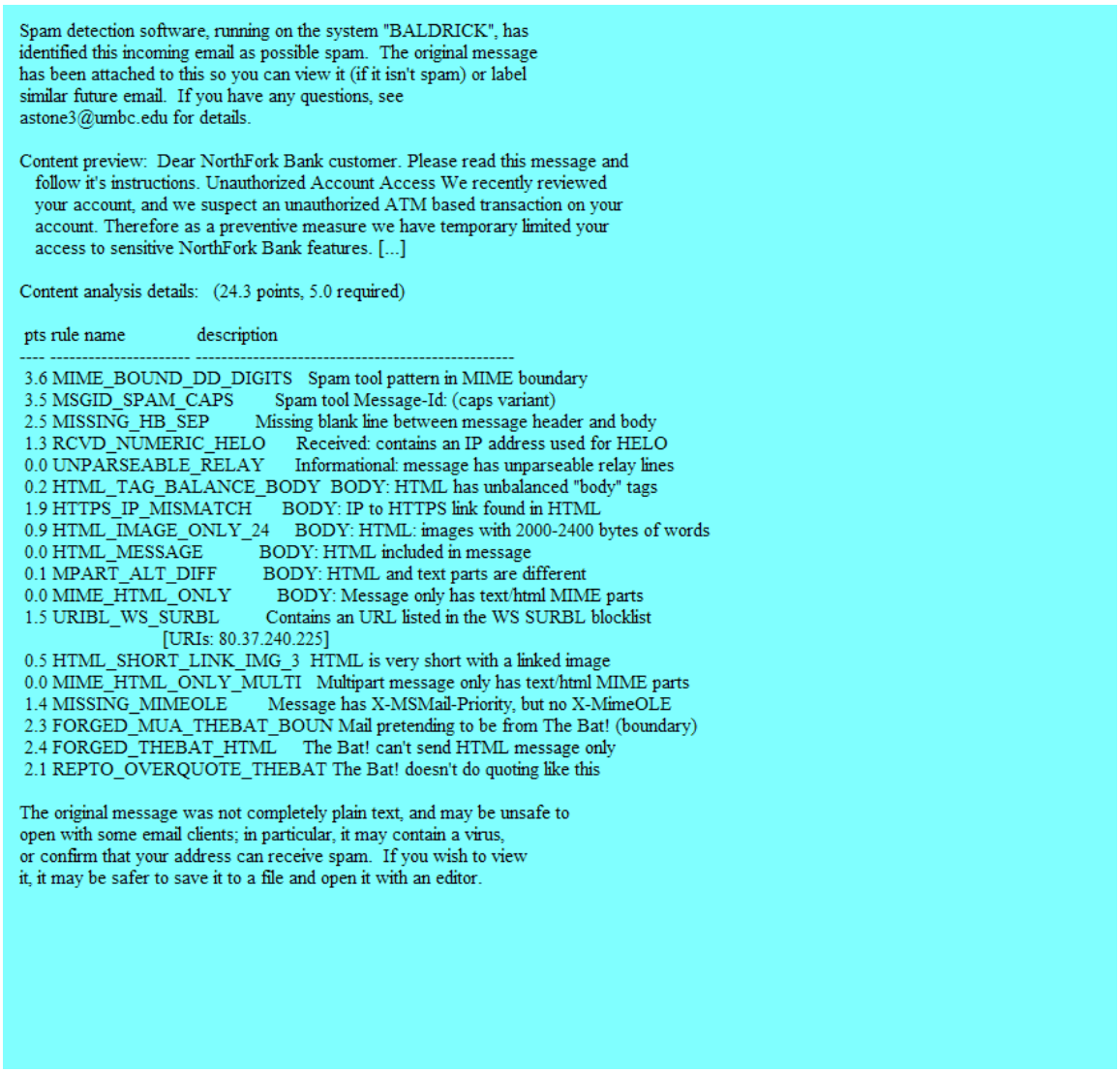
```
OBIDS Alert File
Mon Mar 26 11:42:04 EST 2007

-------------------------------------------
1. Sender - "PayPal" <service@paypal.com>   Timestamp - Mon, 07 Mar 05 19:48:08 GMT
-------------------------------------------
Receivers - <username@domain.com>
-------------------------------------------
Subject - PayPal Account Security Measures
-------------------------------------------
Description - Corporate Deception Rule 1 - Account Compromise, Corporate Deception Rule 3 - Change/Update to
Account
-------------------------------------------
Snippet - . . . . . .                                   Dear PayPal Member,   Your account has been randomly
flagged in our system as a part of our rout ine security measures.  This is a must to ensure that only you have access and use
of your PayPal  account and to ensure a safe PayPa
-------------------------------------------
```

Figure 8. EBIDS Output File with Partial False Positive: From the text "snippet", it can be discerned that no Account Compromise is being alleged, yet the rule fires. The other rule, the Change/Update to Account, fires correctly.

Summary of Results: EBIDS

Total Emails: 536
Total Known-Bad Emails: 224
Total Non-SE Emails: 312
Total Alerts: 175
Total Correct Alerts: 169
Total Completely False Alerts: 6

Total Partially False Alerts: 71
Total Adjusted False Alerts: 77
Hit Rate: 0.75446
False Positive Rate: 0.01923
Adjusted False Positive: 0.14365



```
Spam detection software, running on the system "BALDRICK", has
identified this incoming email as possible spam.  The original message
has been attached to this so you can view it (if it isn't spam) or label
similar future email.  If you have any questions, see
astone3@umbc.edu for details.

Content preview:  Dear NorthFork Bank customer. Please read this message and
   follow it's instructions. Unauthorized Account Access We recently reviewed
   your account, and we suspect an unauthorized ATM based transaction on your
   account. Therefore as a preventive measure we have temporary limited your
   access to sensitive NorthFork Bank features. [...]

Content analysis details:   (24.3 points, 5.0 required)

 pts rule name              description
---- ---------------------- --------------------------------------------------
 3.6 MIME_BOUND_DD_DIGITS   Spam tool pattern in MIME boundary
 3.5 MSGID_SPAM_CAPS        Spam tool Message-Id: (caps variant)
 2.5 MISSING_HB_SEP         Missing blank line between message header and body
 1.3 RCVD_NUMERIC_HELO      Received: contains an IP address used for HELO
 0.0 UNPARSEABLE_RELAY      Informational: message has unparseable relay lines
 0.2 HTML_TAG_BALANCE_BODY  BODY: HTML has unbalanced "body" tags
 1.9 HTTPS_IP_MISMATCH      BODY: IP to HTTPS link found in HTML
 0.9 HTML_IMAGE_ONLY_24     BODY: HTML: images with 2000-2400 bytes of words
 0.0 HTML_MESSAGE           BODY: HTML included in message
 0.1 MPART_ALT_DIFF         BODY: HTML and text parts are different
 0.0 MIME_HTML_ONLY         BODY: Message only has text/html MIME parts
 1.5 URIBL_WS_SURBL         Contains an URL listed in the WS SURBL blocklist
                            [URIs: 80.37.240.225]
 0.5 HTML_SHORT_LINK_IMG_3  HTML is very short with a linked image
 0.0 MIME_HTML_ONLY_MULTI   Multipart message only has text/html MIME parts
 1.4 MISSING_MIMEOLE        Message has X-MSMail-Priority, but no X-MimeOLE
 2.3 FORGED_MUA_THEBAT_BOUN Mail pretending to be from The Bat! (boundary)
 2.4 FORGED_THEBAT_HTML     The Bat! can't send HTML message only
 2.1 REPTO_OVERQUOTE_THEBAT The Bat! doesn't do quoting like this

The original message was not completely plain text, and may be unsafe to
open with some email clients; in particular, it may contain a virus,
or confirm that your address can receive spam.  If you wish to view
it, it may be safer to save it to a file and open it with an editor.
```

Figure 9. SpamAssassin Output, showing the usual message, followed by the summary of technical signatures that fired and their explanation.

Summary of Results: SpamAssassin
Total Emails: 549
Total Known-Bad Emails: 241
Total Non-SE Emails: 308
Total Alerts: 230
Total Correct Alerts: 219
Total Completely False Alerts: 11
Hit Rate: 0.90871

False Positive Rate: 0.03571

Summary of Results: Snort
Total Emails: 551
Total Known-Bad Emails: 241
Total Non-SE Emails: 310
Total Alerts: 0
Total Correct Alerts: 0
Total Completely False Alerts: 0
Hit Rate: 0
False Positive Rate: 0

3.5 Evaluation of Results

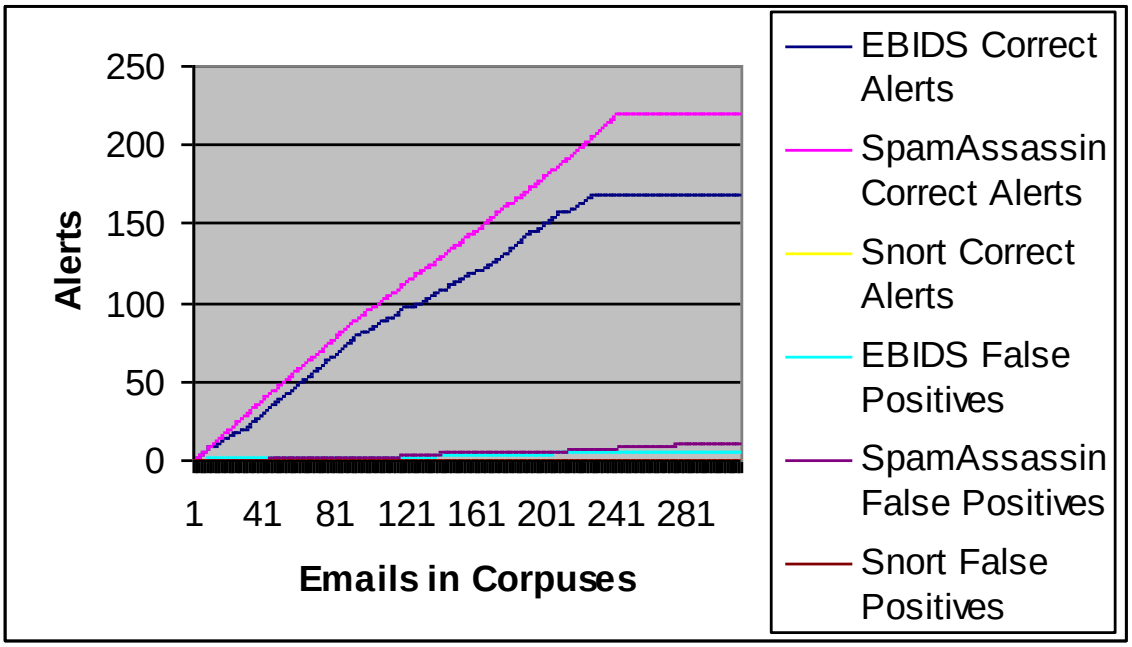


Table 1. Total Alerts by Tool

The performance of EBIDS was impressive. With only four detection rules written, it achieved a relatively high hit rate of 75%, which is being compared unfavorably only to SpamAssassin's 90% hit rate, whereas SpamAssassin has a much more extensive signature set. So, as far as detecting malicious social engineering emails as they occur, it did not outperform SpamAssassin, but the system outperformed SpamAssassin when it came to false positives, although both systems fared relatively well in that regard. False positives were expected to be high with EBIDS because of the nature of its algorithm as it relates to string matching plain English.

There were a few common foibles of EBIDS that led to its higher-than-expected miss rate (inverse of the hit rate). One such problem was a subset of the Nazario phishing corpus that included emails where the subject line of the email contained the

actual misleading natural language and the body of the email was simply an image or executable/zipped text. This was a foreseen problem with the system, but there were more of that type of email than expected. Also, this seems to be a problem which can be corrected with slight changes that include checking the subject of the email and testing for blank email bodies that include only executable/zipped text and/or images. EBIDS also partially false fired on a specific Paypal security audit fraud, and since that fraud was the most frequently occurring email in the Nazario corpus, the adjusted false positive rate was higher than expected. Of course, due to the lower-than-expected false positive rate, the adjusted rate just about leveled out to expectation with the high adjusted false positives.

SpamAssassin performed much better in its hit rate. It caught 90 percent of the phishing emails, looking for technical points in each email, such as the structure of the email code, the legitimacy of the sender and date fields, along with other such criteria. It is finely tuned to detect spam as it comes in, and the emails in the corpus are not extremely recent, so it fared as well as it was expected, if not a little better. However, its false positive rate was higher than expected, nearly doubling the rate of EBIDS. In perspective, the rate was still manageable at around 3%, which is not bad for most practical applications. Overall, SpamAssassin would be the better choice to run in a production environment, even if the current iteration of EBIDS was feasible in production, but with a more complete signature set, EBIDS may compare more favorably in the future.

Snort did not perform at all. One by one, the emails were downloaded through a web browser as plain text over port 80, which is the port and style that some web-based email clients run. Snort was run in verbose mode and was seen reading the packets, but not one single alert triggered. Thus, neither side of the corpus triggered a single detection rule from Snort's own native rule set or Bleeding Snort's rule set. To be fair to Snort, its focus is mainly on other types of exploits, largely because products like SpamAssassin exist. But it should have found something wrong with 241 malicious emails crossing its tripwire. Snort is still a viable IDS tool, but I think that this demonstration is at least an indicator of one of the tenants of computer systems security, that it should be layered, and there do exist people who rely on only Snort, only Norton's Anti-Virus, only a firewall, or only some anti-spyware software to secure their system/network. EBIDS or SpamAssassin could be a valuable addition to what Snort seems to be lacking here. Because of its reputation and versatility for finding the forms of intrusion that it specializes in, one can never recommend passing on Snort, but for this one specialized purpose, it would at least be in the interest of anyone running Snort by itself to either find or write social engineering signatures that are not in its or Bleeding Snort's standard signature set.

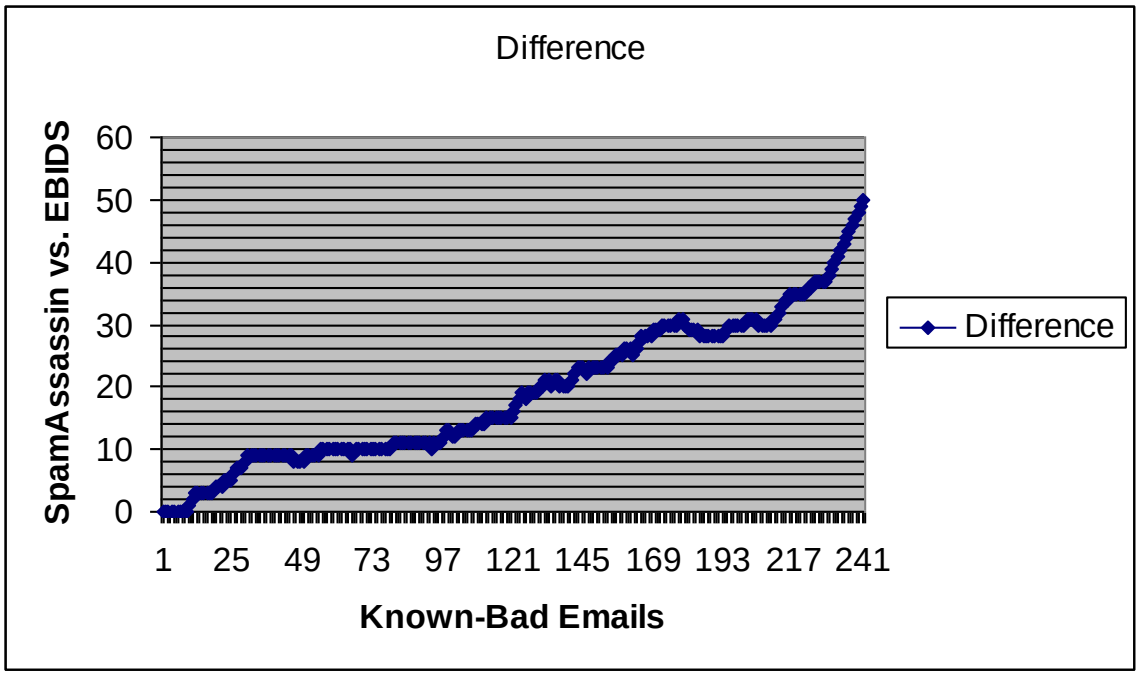


Table 2. SpamAssassin's numbers in comparison to EBIDS for Correct Alerts

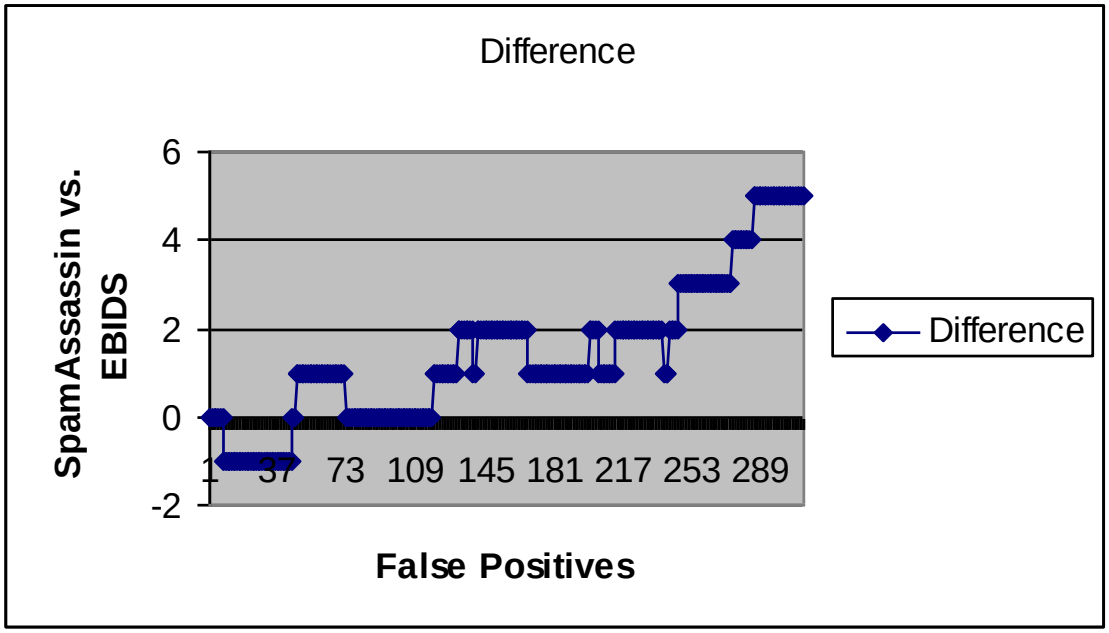


Table 3. SpamAssassin's numbers in comparison for EBIDS for False Positives

Taking in all three systems objectively from these results, it is clear that for the current time being, SpamAssassin is the best detection system for electronic social engineering in email, but EBIDS fared well in its conceptual stage and initial run and thus may be worth further investigation and tuning, and Snort's rule set is not at all appropriate for finding this sort of traffic. In both of the tables above, SpamAssassin ends up with the higher value. The hit rate difference is more favorable for

SpamAssassin, and while the false positive rate is less favorable for it, the numbers are not significantly large.

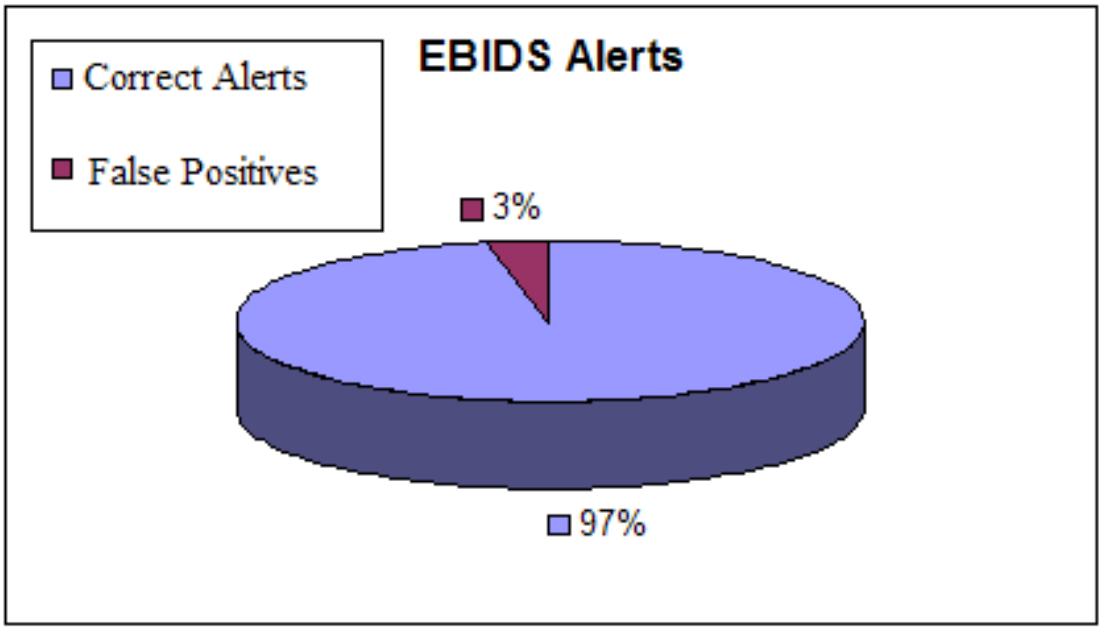


Table 4. EBIDS performance numbers (hits vs. false positives)

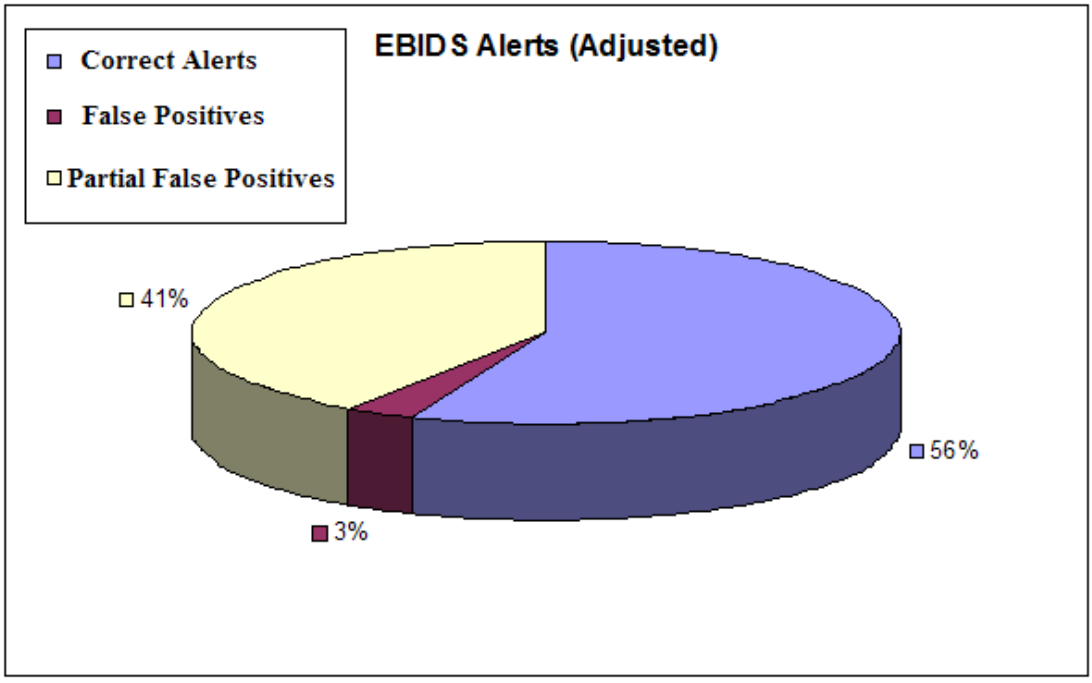


Table 5. EBIDS performance numbers (hits vs. false positives vs. partial false positives)

EBIDS alerts, shown in both Tables 4 and 5, can be looked at in two ways. The number of hits vs. complete false positives is amazing. When the system alerts,

chances are that it is dead on the money. Only three percent of its alerts were fired on completely innocent traffic. In table 5, however, one can see the growing concern of partial false positives, which involves rules that were never intended to fire firing alongside the correct rule. This is not as big a deal in the realm of security, since the data itself was malicious, so alerting on it draws attention where it is needed, but it is still worthy of concern, because it could lead to greater completely false positives down the road. This result is not to take away from the fact that EBIDS needs a higher hit rate in general, but it means that the rules have to be tuned to not only increase the hit rate, but also to lower that partial false positive rate. Now, for a comparison to SpamAssassin, Table 6 shows that when SpamAssassin triggers an alert, there is a higher probability that it is a false positive. Also, SpamAssassin's correct alerts were not investigated in depth to look for any partial false positives.

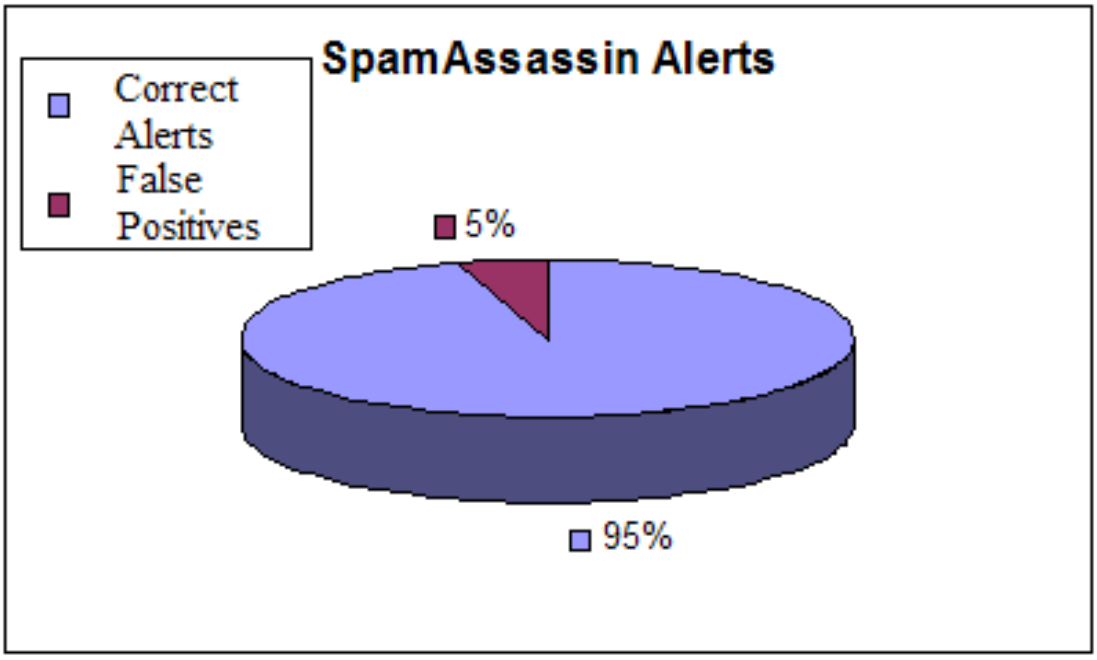


Table 6. SpamAssassin performance numbers (hits vs. false positives)

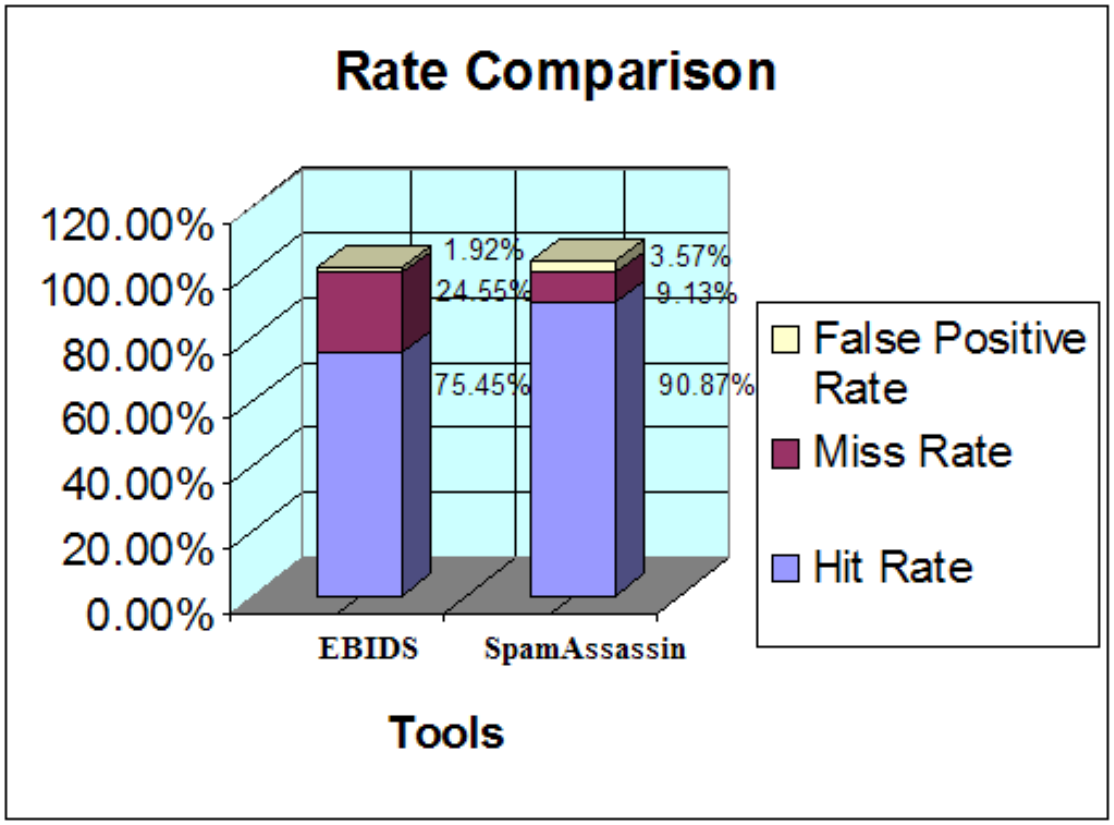


Table 7. EBIDS and SpamAssassin performance numbers (Hit/Miss/False Percentages)

In the big picture, however, as demonstrated here in Table 7, SpamAssassin's gains in hit rate are significant enough to override the concern about a higher false positive rate, and it is not unreasonable to assume that the false positive rate of EBIDS might climb slightly as improvements were made to increase its hit rate. So, for the time being, EBIDS has more accurate alerts, but SpamAssassin has alerts that catch more of the malicious traffic, and that is what is most important in IDS at the end of the day. EBIDS held its own in this test, and it can be discerned that this methodology can be made to bear fruit with the right amount of attention and future work.

Chapter 4

Discussion and Future Work

3.1 Conclusions

With the current state of IDS, it appears that SpamAssassin represents a viable solution to combating social engineering, in addition to Personnel Training, but EBIDS has shown here that it is also a fairly effective tool and has done so with a very small signature set relative to SpamAssassin. In order for this work to move forward and become applicable in the real world, its signature set must be updated/expanded to look for more than just phishing, and its interaction with OntoSem must either be sped up dramatically, achieved through some new addition to OntoSem, or dropped altogether for a newer, more efficient and focused Natural Language Processing system, the latter of which would in most cases be entirely infeasible. Many standard IDS packages come with hard-coded signature sets, the way that most antivirus software does, but this project was borne under the same philosophy that Snort takes to IDS, that the rule set should be customizable by the user, in order to tailor the set to each user's specific network. Thus, hard-coding is not the long-term answer, and so OntoSem must be updated to include this very specific function. However, the overarching conclusion that has been realized from this work is that it can be done effectively. EBIDS may be the start of something new in the world of Network Security.

3.2 Research Questions and Future Work

EBIDS raises many interesting research questions. With only four signatures, the system produces a high hit rate on a phishing corpus. Would expanding or fine tuning the signature set raise that hit rate significantly? How would this current signature set fare against another phishing corpus? Most importantly, how much larger does the signature set need to grow to make EBIDS achieve the level of success of a SpamAssassin? Does it save the user on the size of the signature set over time, as was intended in its design? Even though it is assumed that data acquisition can be performed on OntoSem to expand its capabilities to interact with EBIDS, can it in fact be expanded as such, and if it can, is that really the optimal way to handle such a thing, or should a specialized process/ontology be gleaned from OntoSem to more efficiently perform this task? As has been mentioned a few times throughout the paper, can the idea of signature writing be abstracted from the specifics and internal mechanics of OntoSem, so that future users can more easily write signatures? With the EBIDS-OntoSem interaction, can the speed of execution be increased dramatically to put EBIDS on par with industry standard products?

EBIDS has proven through this research that it is at least well along its way in providing a viable solution to the social engineering problem where phishing is concerned. It would behoove security minds to experiment with expanding the signature set to incorporate more than phishing and to test this project against a new corpus. Outside of phishing, social engineering is used to infiltrate networks and

machines through worm spread emails and via other tactics, such as targeted impersonation, pretending to be management to get users to open word documents that compromise their systems. There are other motivations than money which exist, and there are many other media to deliver social engineering messages electronically, such as system messages (pop-ups that tell the user that they need to update or clean their computer) and websites. This project also has the capability to grow into a full-fledged signature-based IDS, since it can match on literal strings. Work could be done to expand EBIDS to do so, probably requiring a faster string match algorithm than just the internals of Java working on plain text.

Works Cited

1. Cooper, Mark, Stephen Northcutt, Matt Fearnow, and Karen Frederick. Intrusion Signatures and Analysis. Thousand Oaks, CA: New Riders Publishing, 2001.

2. Denning, Dorothy E. "An Intrusion-Detection Model." Symposium on Security and Privacy in Oakland, California (1986).

3. English, Jesse. "DEKADE II: An Environment for Development and Demonstration in Natural Language Processing." Unpublished Master's Thesis, University of Maryland Baltimore County, May 2006.

4. Ertoz, L., E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar, P. Dokas. "The MINDS – Minnesota Intrusion Detection System." Data Mining: Next Generation Challenges and Future Directions. Hillol Kargupta, Anupam Joshi, Kirshnamoorthy Sivakumar, and Yelena Yesha. AAAI Press, October 1, 2004.

5. Gao, Y., and G. Zhao. "Knowledge-based Information Extraction: A Case Study of Recognizing Emails of Nigerian Frauds." 10th International Conference on Applications of Natural Language to Information Systems, NLDB 2005, held in Alicante, Spain (June 2005).

6. Garfinkel, Simson, and Gene Spafford. Practical UNIX Security. Sebastopol, CA: O'Reilly & Associates, Inc., 1991.

7. Kerremans, Koen, Yan Tang, Rita Temmerman, and Gang Zhao. "Towards Ontology-based E-mail Fraud Detection." 12th Portuguese Conference on Artificial Intelligence, Workshop on building and applying ontologies for the semantic web (2005)

8. Mahesh, Kavi, and Sergei Nirenburg. "A Situated Ontology for Practical NLP." Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligence in Montréal, Canada (1995).

9. McClure, Stuart, Joel Scambray, and George Kurtz. Hacking Exposed: Network Security Secrets and Solutions, Third Edition, 3rd Edition. McGraw-Hill Professional, 2001.

10. Mitnick, Kevin D. and William L. Simon. The Art of Deception. Indianapolis, IN: Wiley Publishing, Inc., 2002.

11. Mukherjee, B., L. T. Heberlein, and K. N. Levitt. "Network Intrusion Detection." IEEE Network. Vol. 8, no. 3 (May-June 1994): 26-41.

12. Nirenburg, S., M. McShane, and S. Beale."Operative Strategies in Ontological Semantics." Proceedings of HLT-NAACL-03 Workshop on Text Meaning. Edmonton, Alberta Canada (June 2003).

13. Nirenburg, S., V. Raskin and S. Sheremetyeva. "Lexical Acquisition." Proceedings of NATO Advanced Science Institute on Lesser Studied Languages in Ankara, Turkey (2000)

14. Northcutt, Stephen. Network Intrusion Detection: An Analyst's Handbook. Thousand Oaks, CA: New Riders Publishing, 1999.

15. Peikari, Cyrus, and Anton Chuvakin. Security Warrior. Sebastopol, CA: O'Reilly & Associates, Inc., 2004.

16. Roesch, Martin. "Snort – Lightweight Intrusion Detection For Networks." Proceedings of the 13th Large Installation System Administration Conference, (1999): 229-238

17. Schiffman, Mike, Bill Pennington, David Pollino, and Adam J. O'Donnell. Hacker's Challenge 2: Test Your Network Security & Forensic Skills, 2nd Edition. McGraw-Hill Osborne Media, 2002.

18. Schwartz, Alan. SpamAssassin. O'Reilly Media, Inc., 2004.

19. Stevens, W. Richard. TCP/IP Illustrated, Volume 1: The Protocols. Addison Wesley, 1994.

20. Stevens, W. Richard. TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols. Addison Wesley, 1996.

21. Wright, Gary R., and Richard W. Stevens. TCP/IP Illustrated, Volume 2: The Implementation. Addison Wesley, 1995.

22. Zhao, G., J. Kingston, K. Kerremans, F. Coppens, R. Verlinden, R. Temmerman, and R. Meersman, "Engineering an Ontology of Financial Securities Fraud", 10th International Conference on Applications of Natural Language to Information Systems, NLDB 2005, held in Alicante, Spain (June 2005).