A Comparison of Record and Play Honeypot Designs

[placeholder for review]

Abstract: *Record and play -honeypots mimic the normal TCP traffic and fool the adversary with fake data while simultaneously keeping the setting realistic. In this paper, we propose several designs for such honeypots. Two important aspects of honeypot design are considered. First, we compare named entity recognition systems in order to recognize the entities in the messages the honeypot modifies. Second, we consider methods to fake these entities consistently. Pros and cons of each approach – varying from the better accuracy of the fake responses to the possibility of causing side effects on the real services – are discussed.*

Key words*: honeypot, named entity recognition, proxy*

INTRODUCTION

Cyber attacks and cyber intelligence have become common in computer networks. Novel approaches to defend computer systems against sophisticated threats such as malicious insiders and advanced persistent threats (APT) are needed [16]. One such approach is deceiving the attacker by creating fake services that feed the attacker fabricated information. At the same time, the deceptive service can also collect valuable information on the attacker's behavior in the system.

It appears a large part of research on fake services and fake entities they contain simply aims to detect the malware and to get rid of the malicious attacker as soon as possible [13]. While this is understandable in many settings, our objective here is to find out how we can keep deceiving and interacting with the attacker as long as possible in order to learn more about malicious programs' behavior and goals. Therefore, we keep giving the adversary fake data in order to lure out the malicious program's real intent. In other words, we concentrate on *high-interaction honeypots* that typically interact with the malware in order to gain a better understanding of the methods and technologies malicious adversaries use [11].

To accomplish this goal, a tool has to learn what the typical communication between a client and a service (for example, the exchange of messages related to some public-facing web service) looks like and then try to mimic it. We call this kind of honeypot design the *record and play -approach*.

In this paper, we present different designs for record and play -honeypots. The discussion is divided in two main parts. First, we consider named entity recognition (NER) systems that automatically identify the entities to be replaced for the adversary. We discuss different approaches presented in the literature and provide a comparison between them. Second, once entities we want to fake are identified, we need to create believable and consistent fake data that the real data is replaced with. We present different approaches to achieve this and discuss their strengths and shortcomings.

ENTITY RECOGNITION

A named entity (NE) is an entity that refers to a real word object with a proper name, a numerical expression or a temporal expression. The commonly used categories for NEs are ORGANIZATION, PERSON, LOCATION, DATE, TIME, MONEY, PERCENT, FACILITY (man-made artifacts such as monuments) and GPE (geo-political entities such as cities) [3], but they are not limited to those either. Named entity recognition first tries to recognize

named entities and then classify them into groups. We start by looking into the most basic record and play honeypot designs and move into the more advanced with more advanced NER technologies.

Systems based on simple static rules

The first and simplest way is to implement an in-place honeypot system as a simple proxy as shown in Figure 1. The proxy would modify the traffic in real time by following manually created rules or already known list of entities. The rules can be created for example by using regular expressions or even simple if statements.

Regular expressions can easily detect entities that have an easily detectable structure such as phone numbers, email addresses and web addresses and of course the predetermined words. Unknown names, locations and other entities that require semantic understanding are way harder to detect this way. Some heuristics and rules such as "the first letter of the word is capital if it is not appearing after the dot" can be made to detect that the word is a named entity but the quality and amount of rules as well as the other information available determines the final quality of the recognition. For example, in a study by Sekine *et al.* [15] about 1400 manually created rules were used.

A process called named-entity recognition (NER) should yield better results. There are two major different approaches for NER systems [10]: one that relies only on manually typed grammar rules and another using rules alongside statistical, machine learning approach. The grammar based approach needs a lot of manual work by professionals to create the rules and as such, plain rule based systems are not common any more [10] even though the systems having machine learning components use some grammatical rules as well.

The NER is a pipeline comprised of multiple components. One common pipeline used by for example NLTK [3] is presented in Figure 2. We can see that the pipeline is much more complex than plain regular expressions even though regular expressions, word lists (gazetteers) and other manually created rules can be used with machine learning.

As an example of the importance of other methods, in [3] a plain regular expression -based parser was used in part-of-word (POS) tagging that is one of the pre-phases of NER as it is shown in Figure 2. They evaluated different taggers against Brown gold standard annotated corpus' news category and calculated the accuracy by using the evaluation function. The evaluation function counts correctly identified tags out of all tags and calculates the percentage. Correctly identified tags are previously manually annotated. The tests in [3] resulted accuracy of 0.203264 in regular expression tagger and 0.844513 in backoff tagger that combines multiple taggers such as default tagger that marks everything as nouns, unigram tagger and a bigram tagger. As NER tagging requires POS tags, difference of 64,1 percentage points between the methods has a notable effect on the NER tagging performance.

Systems based on trained NER systems

As noted earlier, using plain manually created rules for faking NEs is inadequate or at least a major workload. To overcome this limitation, we could simply replace the static rules with modern named entity recognition as it is done in Figure 3. The statistical systems need to be trained for them to work. The system in Figure 3 has been trained with a generic corpus such as one created of news stories. The text domain and genre do affect the NER but the effect varies across the domains and genres [7].

There are multiple libraries and programs with many annotated corpora to provide NER support. There is Natural Language toolkit (NLTK) [3] for Python, General Architecture for Text Engineering (GATE) [5] with Nearly-New Information Extraction System (ANNIE) [5] and Stanford Named Entity Recognizer [6] to list a few.

We tested two common natural language tools to perform NER tagging with the de-

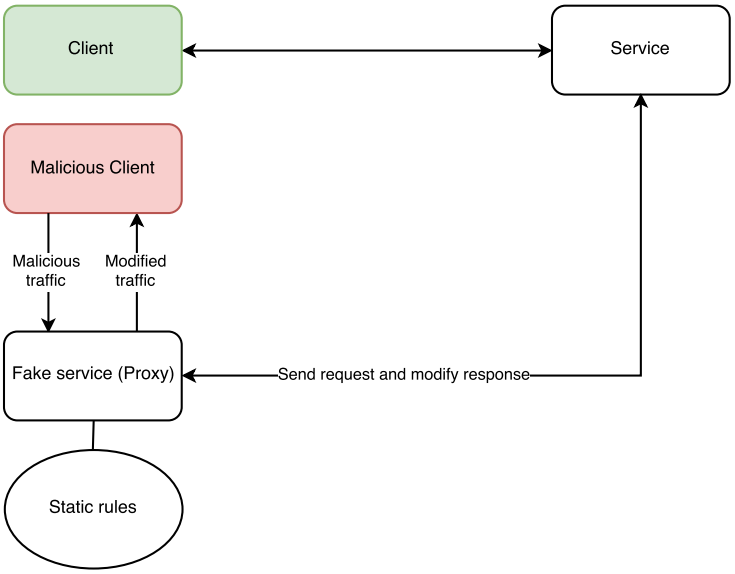

Figure 1: A simple proxy that contains static rules

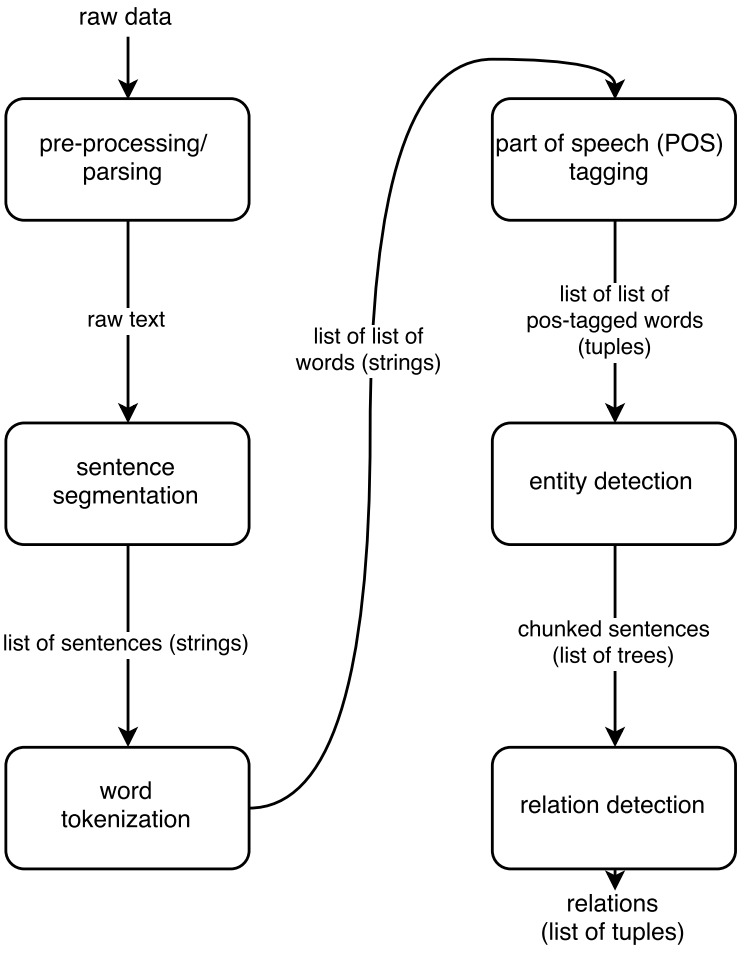

Figure 2: An example of a named entity recognition/information extraction pipeline

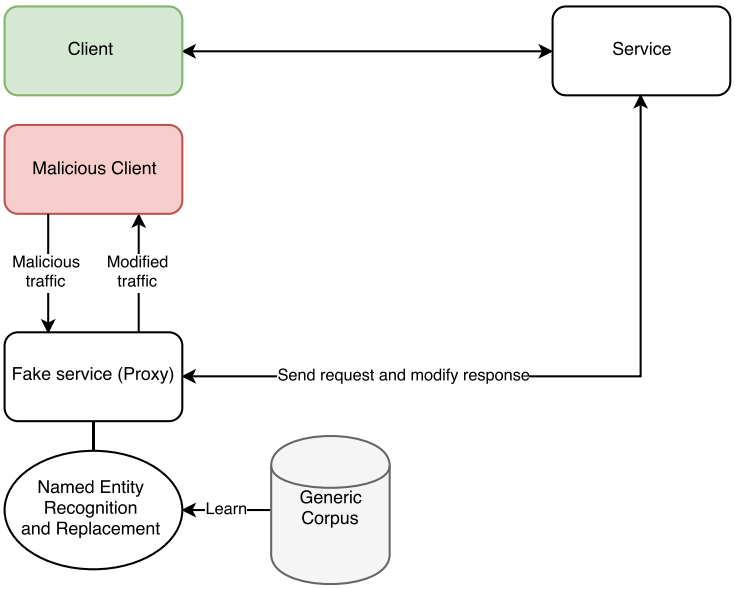

Figure 3: A simple proxy that uses non-domain corpus to identify named entities

fault parsers and corpora to noisy HTML data to see how it performs with personal information recognition. The tested tools were NLTK NER binary classifier and GATE with ANNIE pipeline. Two kind of source materials were used: an extract from student registry administration web site and a plain HTML list of people with personal information. These pages were modified into templates, filled with fake information generated from database dump from generatedata.com and later the natural language tools were evaluated against the manually annotated version.

The database with the fake data had the following attributes with every entity: FNAME, LNAME, BDATE, ADDRESS, ZIP, CITY, PHONE, EMAIL, NID (as in SSN), ORGNUM (as in organizational number), SDATE (start date) and EDATE (end date). And these were used on the templates. The templates did not include any other NEs itself. The true positives are marked only if the system got the entity itself marked and not other noise inside. The entities were not categorized in this test, it was enough that the system marked the entity as named. The entities were the same for every system and test. Five different entities were used in test rounds. Results for precision and recall are presented in Table 1.

Table 1: Comparison of precision and recall with and without pre-parsing using variable types of source material

		Precision	Recall
Unmodified HTML (NLTK)	Student record	0.0942	0.1565
	List of people	0.3696	0.2833
Pre-parsed HTML (NLTK)	Student record	0.0468	0.0800
	List of people	0.3617	0.2833
Unmodified HTML (GATE)	Student record	0.0871	0.4348
	List of people	0.8108	0.5000
Pre-parsed HTML (GATE)	Student record	0.2370	0.3200
	List of people	0.8529	0.4833

As can be seen from Table 1, the systems do not perform that well with noisy data having no normal textual structure. There were some observations during the testing: The NLTK NER tagger does not support numerical or temporal expressions as is and as such will perform worse with data containing lots of dates. Moreover, removing HTML tags from the data, making it less noisy, actually makes NLTK perform worse than without the operation. GATE on the other hand increases its precision score with a small drop in recall. This can

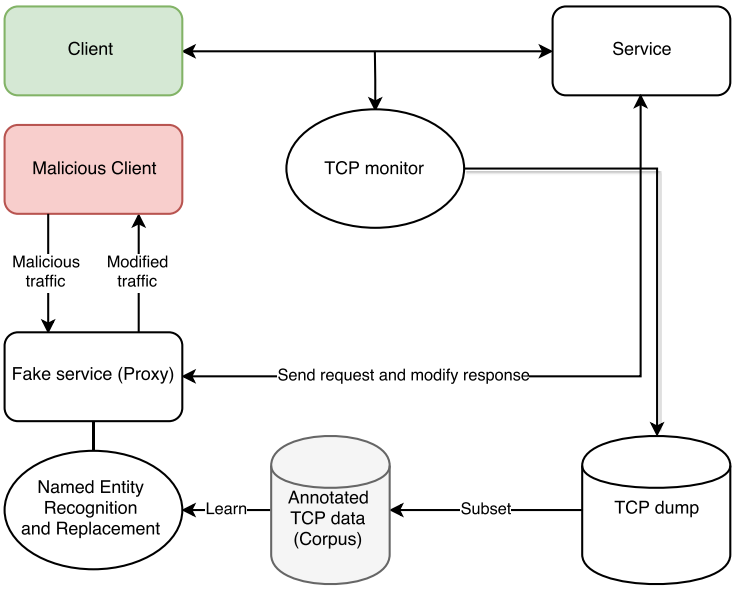

Figure 4: A fake service that gathers TCP data to have a domain specific corpus

be explained with smaller amount of false positives. NLTK appears to combine NEs with non-NEs more when the HTML was stripped and as such failed more on true positives that will affect both, recall and precision.It should be noted that NER systems are not designed to be used this way and the modest results in many cases are partly caused by this. Still, GATE manages to achieve promising results in some cases as the parsed list of people gets the precision of 0.8529 with 0.4833 recall. Machine learning can be used to considerably improve these results.

Systems based on machine learning

If the accuracy of entity recognition is not satisfactory in the model introduced in the previous section, the system can be improved by monitoring and storing legit TCP traffic and manually annotating it with POS and NER tags, using a more suitable gazetteer and also using better pre-parser to clean the data. This annotated TCP data can be used to teach the NER tagger to detect the domain and textual genre specific information better [7, 9, 12]. This kind of system is introduced in Figure 4.

The annotated TCP data can be learned by the system. Currently supervised learning is the dominant technique that includes machine learning algorithms such as Hidden Markov Models [2], Decision Trees [14], Maximum Entropy Models [4], Support Vector Machines [1] and Conditional Random Fields [8]. In addition to the supervised learning, semi-supervised learning and unsupervised learning techniques are being researched.

While the accuracy of NER can be improved, there is one major flaw in the design introduced in Figures 1, 3 and 4: there is an active link to the real service. Proxying request to the real service to get back authentic responses guarantees that the information is somewhat sane. At the same time however, it might have some undesired side-effects in the real system. For example, in context of HTTP, malicious client might send POST request to the server and while the server would respond accordingly, it could also change the state of the server application.

There are different ways to approach this problem. First, we could see whether the real service supports a simulation mode where the requests are processed but no permanent changes are made. This approach would be the easiest to use but would require such support from the service, which is unlikely. Also, such support would mean that the real service in

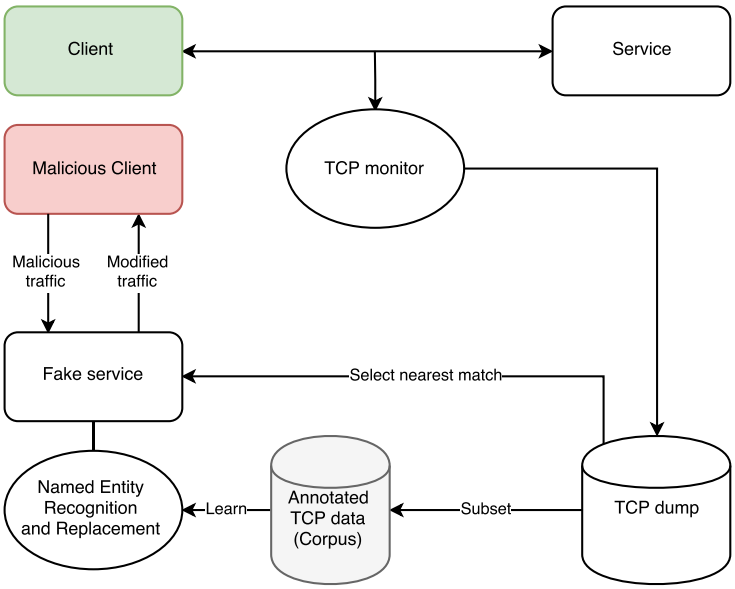

Figure 5: Generating the response by finding nearest match from the history and using NER to fake information

question should save the simulated actions to remember the state it offers to the fake service provider. Implementing such support in a real service would render the honeypot functionality non-universal and nullify the idea behind the record and play -system.

Other way to approach this problem would be to modify the requests in the proxy so that no harmful requests are passed to the service. This would require a lot of knowledge in the real service itself and also would not guarantee that the real service is fully protected from the malicious client. In Figure 5, a model without direct two-way communication to the real service is introduced. The idea is that the TCP monitor would collect TCP data into the database and the fake service would later search the TCP dump database for the best match for the request and modify it accordingly.

FAKING INFORMATION CONSISTENTLY

In the previous section, we discussed the technical designs and implementations of entity recognition systems. However, a system still needs the fake data that the real data is replaced with. Next we will look into some of the possible options as well as their advantages and shortcomings.

Randomly selected fake entities

The simplest way to generate fake data is to replace the recognized entities with a randomly selected entity from the fake entity database. This approach would not require any understanding of the relations between entities nor would it need storing of the entity mappings. Also, it would not reveal the relations present in the real service. This approach also has some severe shortcomings, however: For example, the entities and their relations would not have any consistency in the text. As the record and play -approach requires consistency between messages to deceive the adversary, this approach is not an option here.

One to One mapping

In one to one mapping every time a new entity is encountered, a new entity mapping is created. When an already known entity is encountered, the mapped country is recalled from the data storage using hash mapping or similar technique.

The one to one mapping is a fairly easy and robust way to achieve consistent relations

between fake entities. Every entity that is recognized in NER phase can be mapped to a different entity while keeping the relations sane. The problem with one to one mapping is that while the relations are sane, they are also exactly the same as in the real service. In practice, this means that even though a malicious user could not tell that John and Bill are working in Texas, he or she would know that they are working in the same location. To overcome this, relations between entities should be extracted from the real service.

n-to-n mapping with relation detection

In one to one mapping, every entity had their corresponding fake entity. In n to n entity mapping, every tuple of entities has its fake counterpart where n is the length of a tuple (amount of entities). For example, a double (John, Texas) will resolve into (Peter, Minnesota) but (Bill, Texas) into (Ross, Alabama). Notice how the mapping is consistent between every instance of tuples but different between single entities.

The n-to-n mapping has its problems though. First, the relation extraction systems do not always detect relations and even if they would, not all relations are visible from every context. In an ideal situation where relation extraction would work perfectly and find every relation and there would be only isolated simple identities with attributes (a tree), the system would also work. Secondly, it is possible to change only NEs with methods previously presented. This will introduce a problem into our system: If the text concludes that entity a and entity b are both related via common entity that is not named, a contradiction follows. As an example, the text might imply that entities a and b both live in the same area but when queried by name, the system would give them different locations.

CONCLUSIONS AND FUTURE WORK

This paper has discussed technical design of record and play -honeypot systems, including named entity recognition systems, as well as consistently creating the fake data the recognized entities are replaced with.

We have seen the problems the NER systems have with noisy and uncommonly structured text. We have concluded that simple regular expression parsers work fine for entities with easily detectable syntax such as e-mail addresses. Based on the comparison of the designs, combination of pre-parser cleaning the data, simple regular expression based parser and advanced NER parser with proxy based model would provide best reliability for our record and play -honeypot scheme. At the same time, this setup would not require as much manual work as self-tagged TCP-data.

We believe that of the alternatives presented, most consistent deception will be achieved with one to one mapping. Relation extraction would work by not exposing the relations in real data but as the system can only replace named entities without further understanding of the language, the system would introduce contradictions.

Future work will focus on building an implementation of record and play -honeypot with the presented techniques and performing practical experiments with it. Different pre-parsers for NER-systems, other consistent deception methods and methods to send responses for requests based on collected TCP data instead of relaying the requests to external server are also possible research topics.

REFERENCES

[1] Asahara, M., and Matsumoto, Y. Japanese Named Entity Extraction with Redundant Morphological Analysis. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1* (2003), Association for Computational Linguistics, pp. 8–15.

[2] Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R. Nymble: A High-performance Learning Name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing* (1997), ANLC '97, Association for Computational Linguistics, pp. 194–201.

[3] Bird, S., Klein, E., and Loper, E. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[4] Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. NYU: Description of the MENE Named Entity System as Used in MUC-7. In *Proceedings of the Seventh Message Understanding Conference* (1998).

[5] Cunningham, et al. *Developing Language Processing Components with GATE Version 8*. University of Sheffield Department of Computer Science, 11 2014.

[6] Finkel, J. R., Grenager, T., and Manning, C. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (2005), ACL '05, Association for Computational Linguistics, pp. 363–370.

[7] Maynard, D., Tablan, V., Ursu, C., Cunningham, H., and Wilks, Y. Named entity recognition from diverse text types. In *Recent Advances in Natural Language Processing 2001 Conference* (2001), pp. 257–274.

[8] McCallum, A., and Li, W. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons. In *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4* (2003), Association for Computational Linguistics, pp. 188–191.

[9] Minkov, E., Wang, R. C., and Cohen, W. W. [extracting personal names from email: Applying named entity recognition to informal text.

[10] Nadeau, D., and Sekine, S. A survey of named entity recognition and classification. *Lingvisticae Investigationes 30*, 1 (2007), 3–26.

[11] Nawrocki, M., Wahlisch, M., Schmidt, T., Keil, C., and Schonfelder, J. A Survey on Honeypot Software and Data Analysis, 2016. *arXiv preprint* (2016).

[12] Poibeau, T., and Kosseim, L. Proper name extraction from non-journalistic texts. *Language and computers 37*, 1 (2001), 144–157.

[13] Rauti, S., and Leppänen, V. A survey on fake entities as a method to detect and monitor malicious activity. In *25th Euromicro International Conference on Parallel, Distributed and Network-based Processing* (2017).

[14] Sekine, S., et al. NYU: Description of the Japanese NE system used for MET-2. In *Proceedings of Message Understanding Conference* (1998).

[15] Sekine, S., and Nobata, C. Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy. In *LREC* (2004), pp. 1977–1980.

[16] Virvilis, N., and Gritzalis, D. The Big Four – What we did wrong in Advanced Persistent Threat detection? In *Proceedings of Eighth International Conference on Availability, Reliability and Security (ARES)* (2013), IEEE, pp. 248–254.

ABOUT THE AUTHORS

[placeholder for review]

ACKNOWLEDGEMENTS

[placeholder for review]