Practical Challenges in Building Fake Services with the Record and Play -approach *

Jani Tammi
University of Turku
jasata@utu.fi

Sampsa Rauti
University of Turku
sjprau@utu.fi

Ville Leppänen
University of Turku
ville.leppanen@utu.fi

ABSTRACT

One way to learn more about how a malicious program functions and what its objectives are is to deceive it with fake services that provide responses containing fabricated data. We can try to achieve this goal with so called record and play -honeypot that learns what the normal communication between clients and a server looks like and then tries to mimic it, but fabricates the contents of the responses so that they contain fake data. This paper outlines and presents the challenges faced in practical development of such honeypot. Some solutions and recommendations that mitigate the identified problems are also considered.

Keywords

Honeypots, Protocol replay, Intrusion detection

1. INTRODUCTION

Cyber attacks and cyber intelligence are commonplace in computer networks today and their impact will keep increasing in future. However, when performing a cyber attack, it is not straightforward for a piece of malware to find out whether the service or interface it uses is really a genuine service. By creating fake services, we can deceive the adversary to give us valuable data about its actions and objectives.

A fake service generates responses in which the original content has been replaced with fake content. To be more specific, the response messages contain entities, pieces of data (such as dates, names and locations), that refer to different kinds of objects existing in real world. These original entities inside responses are replaced with *fake entities* to deceive the malware.

*The authors gratefully acknowledge the support of The Scientific Advisory Board for Defence (MATINE). This research is also funded by Tekes – the Finnish Funding Agency for Innovation, DIMECC Oy and CyberTrust research program.

ECSA, November 28-December 02, 2017, Copenhagen, Denmark
 ISBN 978-1-4503-4781-5/16/11…$15.00
DOI: http://dx.doi.org/10.1145/2993412.2993417

In this paper, we discuss several practical challenges faced in building a record and play -honeypot for deceiving the adversary with fake content. This discussion is based on both standpoints presented in the literature and our own practical experiences when implementing an experimental honeypot framework. We also present some recommendations and solutions to mitigate these problems.

2. RECORD AND PLAY -HONEYPOTS

The basic scheme of our "record and play" -approach is depicted in Figure 1. First, a typical sequence of messages between a client and a service is recorded. The client, in most cases, is an application requesting a resource (e.g. a web page) from the service, such as some public-facing web service. From the recorded exchanges, usual patterns used in the messages between the client and the service are learned. This information is used to send responses with modified deceptive contents to a malicious program that is trying to load a resource from our deceptive service.

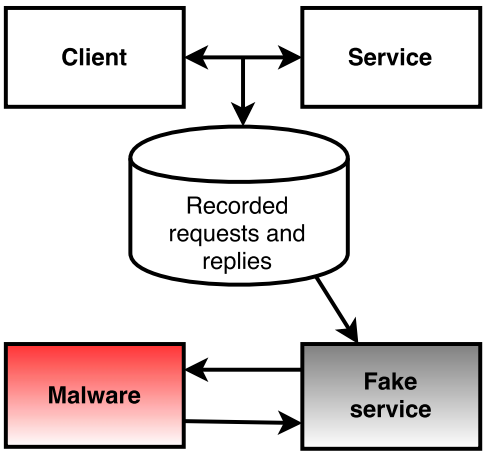

Figure 1. The record and play -approach.

During the record phase, several different message sequences are recorded. Entity recognition is then performed based on the recorded messages. In other words, we have to infer which parts of the message contents contain entities that we are going to modify and what are the relationships between these entities. This is naturally a challenging task. It is possible to infer some field semantics automatically [3] but this phase is likely to require some manual help. When the entities in the messages have been found, convincing fake versions of these entities are generated (e.g. believable employee names) to replace the original entities.

The recorded messages should not be used in their original form for several reasons. First off, we do not want to leak any private or critical data that the original messages may contain. Instead, we make up the data and lie to the attacker. Second, trying to force different reactions out of malware is one objective of deception, because this is useful for analysis. Third, by modifying the data fed to the malware, we can attempt to manipulate malware's actions [4, 1]. Finally, the contents of the responses should also be altered simply in order to deceive the adversary with fallacious information.

3. FEATURES OF AN IDEAL RECORD AND PLAY -HONEYPOT

The following are the desirable features that would describe an ideal record and play honeypot:

- *Creating convincing fake entities.* The entities in fake messages should look convincing for the malware and, whenever reasonably possible, also be believable in human scrutiny. This requires an adequate learning set consisting of typical values of an entity (or sometimes simply a ready-made list of values) in the original service and usually some kind of structural information about the entity in question.

- *Consistency.* The honeypot that produces deceptive replies has to keep the fake entities consistent. For example, the entities may depend on each other. If we have a list of addresses and number of items in this list, we have to remember to update both if we change the list in the fake response. Along with keeping messages internally consistent, we have to preserve consistency between messages belonging to the same session. If we have told the adversary about employee X in some message, we have to keep consistent about this in future messages. Arguably, the consistency of fake entities should even exceed mere sessions. The reason for this is that not all persistent attacks are conveniently contained in one session [9].

- *Protocol-independence.* A perfect record and play -honeypot would do a good job in recognizing the entities in any non-encrypted text-based protocol.

- *No data leaks.* Since the fake service uses the messages traveling between original non-malicious clients and the real service as its starting point, there is also a risk that the data in these messages might leak to the adversary if its not correctly replaced with fake data in the responses. Therefore the honeypot has to make sure critical data does not leak.

- *Learning from malicious requests.* Upon encountering a new malicious request (a new attack method), the honeypot should be able to respond in a way that is likely to keep the conversation with the malicious party going so that more valuable data can be collected. Although replying to previously unknown attacks perfectly is not possible, we can try to make educated guesses based on other previously recorded (benign and malicious) exchanges with the help of machine learning approaches. Some human help may be necessary in this learning process, but the amount of effort needed in teaching the tool should be reasonable.

- *Good performance.* Preferably, generating fake messages should not cause large performance penalties compared to the original service.

An ideal record and play honeypot would of course implement all of these qualities perfectly. As will became clear in the following section, however, in practice we have to make trade-offs between these features. For example, it is incredibly difficult to keep the tool fully protocol-independent and still at the same time recognize all context dependent entities in a message.

4. CHALLENGES

This section discusses the challenges faced in practical record and play -honeypot development. We will cover a few essential issues we have chosen here based on both related literature and our practical experiences with the experimental honeypot implementation presented in [7]. We also briefly address the implications and mitigations for each issue.

4.1 Recognizing entities

The first and essential task in creating deceptive response messages is to correctly recognize the entities that will be replaced with fake versions. An entity can be a numerical or temporal expression, or a reference to a real world object. For example, `PERSON`, `LOCATION`, `DATE` are some of the usual categories for typical named entities [2]. The goal of entity recognition is to find the entities and classify them into groups.

Entities with an easily detectable structure can be recognized using regular expressions. These include for example phone numbers and email addresses. However, unknown names, locations and other entities requiring semantic understanding cannot be recognized this way. With enough well crafted rules, however, satisfactory results can be achieved [8]. A more complex approach named named-entity recognition (NER) often gives better results. The NER systems are usually mixes of manually typed rules and statistical machine learning approach [6].

Supervised learning and manually annotated entities can be used to improve results. A human can annotate entities in the message payloads and give information on their semantics and dependencies. In practice, it is probably best to use a combination of automated and manual recognition. That is, the system could learn by first automatically trying to recognize entities and then letting a user correct the mistakes with manual annotations that explain entity types to the tool. There are always cases that are hard to handle with automatic recognition, like several alternative presentations for some entity, but these can be solved by manual annotation.

Naturally, a good accuracy can be achieved when the tool uses an annotated description of the message structure. However, this causes some manual work and somewhat undermines the generic, protocol-independent nature of the tool.

4.2 Creating convincing fake entities

Automatically creating believable fake entities – especially for human scrutiny – is not feasible without some kind of context related information and knowledge about the format of required fake entities.

However, when the tool is provided with information on the type and range of the entity, it is quite easy to create short fake entities. For example, age has to be a believable number on certain interval (range) and names consist of two strings (format). In the cases where it is hard to create believable entities automatically, entity generation can also be based on ready-made lists – e.g. a list of Finnish names. This gives the tool the contextual information it needs, since automatic context aware analysis is usually not feasible.

With this kind of human help, both entity recognition and creating convincing entities becomes easier. However, the amount of manual work involved in the process naturally becomes larger an the generality of approach suffers.

Of course, in the cases where the fake entities are only examined by a piece malware, we can probably safely assume the scrutiny is not that thorough and context-aware. In this case it becomes easier to generate entities automatically and make them more generic. When the examiner is a human, on the other hand, requirements and a lot higher. However, it can be argued that resorting to human scrutiny also wastes the attacker's resources much more effectively.

The requirements for the quality of created fake entities also depend on the length of period we want to keep the adversary under the false idea. Proving the data wrong also requires time and work. Of course, the adversary does not necessarily always expect the content of the entities to be fallacious.

The ability to create convincing entities also depends on the application area and the used protocol. We also have to accept that it is impossible automatically create large files or large chunks of believable natural language that would pass meticulous human scrutiny.

4.3 Faking entities with non-trivial presentations

The simplest example of non-trivial presentation could be text rendered into images. This is often done for email addresses as a counter measure against spam bots harvesting addresses on the web. Exotic gimmicks like server side rendering of text into HTML tables can also make automatic entity fabrication considerably more difficult. Humans have a creativity that defies generic and especially automatic solutions and therefore it is very likely that some extent of human attention is always required in fake entity creation. Moreover, not all entities are passive. For instance, there may be Flash based components embedded into the web application, dealing with data related to entities that need to be faked. Solutions to address the challenge of non-trivial presentation (like integrating an image rendering engine into the fake entity creator) can be developed but they still complicate matters and often have to be context-dependent.

4.4 Consistency

The record and play -approach requires consistency between messages to deceive the adversary. In order to generate believable fake responses, we have to keep entities and their relations consistent.

A simple approach to achieve some consistency is *one to one mapping*. When a new entity is found, a new entity mapping (mapping between original and fake value) is created. When an already recorded entity is encountered, the mapped value is recalled from the data storage. A drawback with one to one mapping is that the relations are exactly the same as in the original service. In practice, this means that the adversary can still tell relations between entities (e.g. that two employees are working at the same location). To avoid this problem, entity relations should not be directly derived from the real service. *n to n mapping* builds a fake counterpart for every tuple of entities, where n is the number of entities.

In addition to maintaining consistency in served content, some level of consistency with publicly known facts may also be required. In order to keep the perpetrator convinced that the service is genuine, it has to provide well known facts without alteration. For example, the illusion is likely to shatter if the perpetrator is trying to look up information on a certain employee in and finds the information completely different from expected. This level of semantic understanding will require human input. We therefore need to carefully consider which entities are really to be faked and which are not, while a human operator is instructing the solution on entities.

4.5 Data leaks

As outlined in Section 2, our honeypot scheme uses the original responses as a basis when it forms the fake replies for the malicious party. This, however, leads to a problem of information potentially leaking if the original entities are not replaced correctly with fakes.

The problem information leaks ultimately boils down to how well the tool recognizes the entities and replaces them. That is, both entity recognition and creation of convincing entities have to work well enough. As we saw, if the consistency mechanism for entities is created wrong way, it might also leak some information about the relationships of the entities involved.

Basically, when we have good enough parser for the protocol in question and do not strive for too much protocol independence, we should be able to prevent the leaks. Of course, in some less critical applications, some information leaking might not be a problem. Moreover, even if part of information would be authentic, it is not often apparent for the adversary which parts of information are fake and what parts are not.

4.6 Learning message sequences and responding to unknown attacks

One significant challenge in the record an play -approach is that when the honeypot encounters a new kind of (malicious) message, it is not apparent how to respond. In fact, many tools relying on recorded and analyzed dialogues have no clue how to respond to unfamiliar malicious request sent by the adversary [5].

If we only use recorded message exchanges from original benign interaction as learning material, the tool will not be able to provide good responses to unfamiliar attacks. However, this can probably be remedied by also making the tool learn typical malicious exchanges. This can include both ready-made patterns from typical attack and the real exchanges that the tool encounters while operating. The results can be improved by having human assist in classifying the message exchanges, much like with entity recognition and manual annotation.

In practice, many automated attacks are also encountered repeatedly (or the attacks are very similar to each other because their and methods are the same), which makes it pos-

sible for the honeypot to learn to achieve more satisfactory results by trying different kinds of strategies in communicating with the adversary.

Of course, it is questionable what satisfactory means in this context. For example, it can be taken to mean keeping the conversation with malware going as long as possible in order to gather more data. Also, in the cases where the same piece of malware or is encountered several times, a satisfactory result can mean trying different kinds of responses in order to learn how the malware reacts to different stimuli. Analysts could then go on to build a partial state graph for the malware.

4.7 Protocol independence

We have already seen in this section that recognizing entities accurately and in a context aware manner, creating convincing fake entities and preventing sensitive information from leaking does not fit well together with protocol-independence in a real-world honeypot implementation. Entity recognition and fake entity creation require context specific information, as does fully understanding the states of a specific protocol.

For instance, the nature of many web applications presents a challenge in this sense. Web applications build their application protocol on top of the HTTP protocol (transfer protocol), and whereas HTTP has limited and documented scope, web application protocols have no standards and are entirely application specific. An application protocol for web applications here means a request-response pairs over middleware managed session, consisting of verb + URI and parameters in query string or message body (GET/POST). As the correct response may be dependent on the internal state of the application – which the record and play -honeypot cannot know without in some way finding out this internal state and therefore hurting the protocol-independence – it is infeasible to create a totally universal protocol-independent honeypot.

All usable practical solutions most likely have to relax the requirement on protocol-independence at least to some extent, if the goal is to create even moderately convincing fake responses. After all, if the piece of malware analyzing the messages knows more about the protocol than our fake service, we are usually in trouble. Therefore, we believe the aim should be to make the underlying honeypot framework as protocol-independent as possible and then build the protocol-specific functionality on top of this foundation. This also makes the honeypot implementations more flexible and allows the testing an comparison of different approaches for example in entity recognition and fake entity creation.

4.8 Deception and potential escalation

Current malware generally does not bother with analyzing responses, especially when it is mainly concerned in exploiting some vulnerability in order to gain privileges access to the host system, after which data theft and other illicit actions occur.

If deceptive measures become more commonplace, however, adversaries may find it justifiable to implement some basic analytic measures in order to cut their losses and retreat, preferably not revealing too much about their activities once they have realized that a fake service is engaged. It would therefore be negligent not to at least consider the possibility that the usual cat-and-mouse game might also manifest in the capabilities of fake entity generation and detection.

5. CONCLUSIONS AND FUTURE WORK

This paper has discussed the challenges and complexities faced when designing and implementing a record and play -honeypot. We first proposed several desirable features for a record and play -honeypot – creation of convincing fake entities, consistency, protocol-independence, confidentiality of sensitive data, learning from malicious requests, good performance – and then proceeded to discuss the issues in incorporating these features in a practical implementations. Some conceptual solutions to mitigate these challenges were also proposed.

As a future work, we plan to implement a proof-of-concept implementation of a record and play honeypot framework that can be used in building deceptive services. We have seen that some trade-offs and human involvement are inevitable, deception can still be a valuable tool in analyzing behavior of malicious programs.

References

[1] M. Almeshekah and E. Spafford. Planning and Integrating Deception into Computer Security Defenses. In *Proceedings of the 2014 workshop on New Security Paradigms Workshop*, pages 127–138. ACM, 2014.

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

[3] J. Caballero and D. Song. Automatic protocol reverse-engineering: Message format extraction and field semantics inference. *Computer Networks: The International Journal of Computer and Telecommunications Networking archive*, 57(2):451–474, 2013.

[4] F. Cohen and D. Koike. Misleading attackers with deception. In *Proceedings from the Fifth Annual IEEE Information Assurance Workshop*, pages 30–37. IEEE, 2004.

[5] W. Cui, V. Paxson, N. Weaver, and R. Katz. Protocol-independent adaptive replay of application dialog. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium*, 2006.

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

[7] J. Papalitsas, S. Rauti, and V. Leppänen. A Comparison of Record and Play Honeypot Designs. Accepted for publication, 2017.

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

[9] F. Shafique, K. Po, and A. Goel. Correlating Multi-session Attacks via Replay. In *Proceedings of the 2nd Conference on Hot Topics in System Dependability - Volume 2*, HOTDEP'06, pages 3–8. USENIX Association, 2006.