International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-5, August 2015
ISSN: 2395-3470
www.ijseas.com

# Advanced Honeypot Architecture for Network Threats Quantification

**Mr. Susheel George Joseph M.C.A, M.Tech, M.Phil(CS)**
*(Associate Professor, Department of M.C.A, Kristu Jyoti College of Management and Technology, Changanassery, Kerala. susheelgj@gmail.com)*

***Abstract:*** Today's world is increasingly relying on computer networks. The increase in the use of network resources is followed by a rising volume of security problems. New threats and vulnerabilities are discovered everyday and affect users and companies at critical levels, from privacy issues to financial losses. Monitoring network activity is a mandatory step for researchers and security analysts to understand these threats and to build better protections. Honeypots were introduced to monitor unused IP spaces to learn about attackers. The advantage of honeypots over other monitoring solutions is to collect only suspicious activity. However, current honeypots are expensive to deploy and complex to administrate especially in the context of large organization networks. This study addresses the challenge of improving the scalability and flexibility of honeypots by introducing a novel hybrid honeypot architecture. This architecture is based on a Decision Engine and a Redirection Engine that automatically filter attacks and save resources by reducing the size of the attack data collection and allow researchers to actively specify the type of attack they want to collect. For a better integration into the organization network, this architecture was combined with network flows collected at the border of the production network.By offering an exhaustive view of all communications between internal and external hosts of the organization, network flows can 1) assist the configuration of honeypots, and 2) extend the scope of honeypot data analysis by providing a comprehensive profile of network activity to track attackers in the organization network. These capabilities were made possible through the development of a passive scanner and server discovery algorithm working on top of network flows. This algorithm and the hybrid honeypot architecture were deployed and evaluated.This study marks a major step toward leveraging honeypots into a powerful security solution. The contributions of this study will enable security analysts and network operators to make a precise assessment of the malicious activity targeting their network.

***Keyword:*** *Honeypots, Network Security, Network Attack, Network Threat, Darknet*

## *1. Introduction*

Today's world increasingly relies on computer networks. The use of network resources is growing and network infrastructures are gaining in size and complexity. This increase is followed by a rising volume of security problems. New threats and vulnerabilities are found every day, and computers are far from being secure. In the first half of 2008, 3,534 vulnerabilities were disclosed by vendors, researchers and independents. Between 8 and 16% of these vulnerabilities were exploited the day they were released by malicious programs. The consequences affect users and companies at critical levels, from privacy issues to financial losses.

To address this concern, network operators and security researchers have developed and deployed a variety of solutions. The goal of these solutions is two-fold: first to monitor, and second to protect network assets. Monitoring allows researchers to understand the different threats. Data are being collected to better characterize and quantify malicious activity. The goal of this dissertation is to introduce an innovative framework to better measure malicious threats in the organization network. The framework is based on a flexible hybrid honeypot architecture that we integrate with the organization network using network flows.

## *2. Background*

### 2.1. Network Security

Network malicious activity can be quantified and characterized through two distinct approaches: the first is to monitor production networks, where live hosts and devices are actually used by people; the second is to monitor an unused address space that nobody uses. The advantage of the second approach over the first is that there is no user traffic to filter out. Indeed, the traffic received by unused addresses falls into three categories: malicious activity, misconfiguration, and backscatter from spoofed addresses. On the other hand, the disadvantage of the second approach is to rely on the assumption that malicious activity destined to unused addresses is similar to the one targeting production machines.

Tools used in these two different approaches can be divided into two groups: passive and active tools. When monitoring production networks, passive security tools include intrusion detection systems (IDSs) such as Snort,

and network traffic sniffers such as Tcpdump tc or Netflow. Active tools include firewalls such as Netfilter, intrusion prevention systems (IPSs) such as Snort Inline, and vulnerability scanners such as Nessus. When monitoring an unused address space, passive tools are similar,but active tools are specific sensors developed with the only goal of better investigating the malicious activity received. Historically, unused address spaces were only passively monitored. Then researchers had the idea of actively replying to the traffic received to discover the exact threat behind each connection attempt. To understand the research challenges introduced with this new idea, we will now describe the different existing types of active sensors.

**2.2. Honeypots**

**2.2.1. Definitions**

_ We define a **network sensor** as an unused IP address instrumented to collect information about suspicious traffic. We separate sensors into two categories: *passive sensors*, which simply collect data without any interaction with the source of traffic; and *active sensors*, which can interact with the source of traffic to collect additional information.

_ We define a **honeypot** as a network device that provides a mechanism for completing network connections not normally provided on a system and logging those connection attempts. We note that honeypot and active network sensor are synonyms.

_ We define a **darknet** as a network of passive sensors.

_ Similarly, we define a **honeynet** as a network of honeypots.

_ By **honeypot architecture**, we mean a specific combination of software solutions to administrate a honeynet.

_ Finally by **honeypot framework**, we mean the combination of a honeypot architecture and a data processing solution to analyze malicious network activity.

**2.2.2. Honeypot Attributes and Classification**

The main goal of honeypots is to provide information about network attacks. A large variety of honeypots have been proposed by researchers to collect various types of security threats. These honeypots can be organized according to three main attributes:

_ *Fidelity*: honeypots have different levels of interaction, whether they offer emulated or real services to attackers. The more interactions a honeypot has with an attacker, the more knowledge is gained about the attack. Hence, three different levels of interaction are defined:

1. A *high-interaction honeypot* is a conventional network resource, such as a computer or router, with no active user and no specific task other than getting attacked. From an attacker's point of view, this type of honeypot can hardly be differentiated from another production machine. The advantage is to gain as much information as possible about the attack. Of course, with such a genuine exposure, the risk of being effectively compromised is real. Consequently, these honeypots should be closely monitored and data control mechanisms, such as a reverse firewall, should be configured to prevent an attacker from using the honeypot to damage other production resources. The Honeynet Project provides tools and documentation to deploy and administrate this type of honeypot.
2. A *low-interaction honeypot* provides limited interaction with the attacker by emulating a set of services. The goal of low-interaction honeypots is to gather information about the first steps of an attack. Information about the motivation of the threats received is rarely captured because the level of interaction is too low for the honeypot to be effectively compromised. A well-known implementation of a low-interaction honeypot is Honeyd.
3. A *zero-interaction sensor* is no longer a honeypot but a passive sensor that does not respond to attackers. Such sensors are called darknets and are nonetheless able to collect important information about how attackers probe networks and what services they target.

_ *Scalability*: the level of interaction of honeypots affects the number of IP addresses on which the honeypots can be deployed as well as the maximum bandwidth they can sustain. Indeed, a darknet is more scalable than a set of highinteraction honeypots because, from a resource perspective, passively monitoring thousands of network addresses is less demanding than deploying and administrating a few high-interaction honeypots. As a result, current honeypot architectures offer either large scalability or high interaction but not both.

_ *Security*: as explained for high interaction honeypots, deploying honeypots to actively collect malicious traffic is not a safe activity. Honeypots can be compromised and so several protection systems currently exist to avoid attackers from using honeypots to relay malicious activity.

Honeypots are governed by these three contending attributes: *scalability*, *fidelity* and *security*. Researchers have to balance these attributes according to their needs and their resources. They can either study the first steps of an attack by deploying a large number of low interaction honeypots. Or they can study the full attack process by deploying high interaction honeypots. This last option requires constant monitoring and important software and hardware resources.

**2.2.3. Honeypots and Network Attack Processes**

To better understand how honeypots can be used, it is important to first describe how network attacks proceed to spread and to compromise computers. For this purpose, we define an attack process as a sequence of network communications between an attacker and a victim, with a malicious purpose. We divide the network attack process in three phases:

1. The first phase is to reach a victim, which means to send a communication attempt to a specific service hosted on a network device. For example, a technique for an attacker to discover a large number of victims is to scan incrementally all network addresses within a specific subnet. The attacker goes to the next phase only if the victim replies and the service is open to the attacker.
2. The second phase is to exploit the service found on the victim's machine by launching an attack payload. There is not always a clear boundary between the first and second phase, because some attacks are made of a single network packet , so communication attempts and attack payload overlap. Moreover, attackers often use the connection initialized during the scan to send the attack payload.

The attacker goes to the next phase only if the service is successfully compromised by the attack launched.

3. The third phase is to use the newly corrupted victim's machine. The attacker can

be someone who wants to gain access to a specific resource, or a worm that is simply spreading from one vulnerable machine to another. In such case, the worm installed on the newly corrupted machine will start probing for other victims and will create a new attack process starting with phase one again.

From this model we can map the different phases of network attack with the different types of honeypots. Figure 1 details this mapping and explains how honeypots attributes are related.

The probing phase of an attack can be detected by all types of sensors (zerointeraction sensors, low and high-interaction honeypots). However, to gather significant statistical results about scanning techniques and services targeted, one need to monitor large address space. Therefore, darknets are the most suitable solution to study this attack phase because of their high scalability.

The second phase of an attack can be detected only by sensors which can reply to

probes. The reason is that an attack payload can be sent by the attacker to the sensor only if a network connection is correctly established between the two peers. As we just saw when explaining the second phase of an attack, we can find some exceptions to this requirement, because some attacks are made of a single network packet that does not need first an acknowledgment from the victim to be sent. Low interaction honeypots are well suited to gather exploits sent during the second phase of an attack, because they are scalable and provide enough interaction for the attacker to send its attack payload.

However, emulated scripts hosted by low interaction honeypots will not always satisfy the level of interaction required by complex attacks. This threshold between simple and complex attacks is represented by the level of emulation on Figure 1. Furthermore, low interaction honeypots cannot be compromised by attackers, so the third phase of the attack process is never collected by this type of architecture.

As a result, the full attack process requires high interaction honeypots to be analyzed

in detail. High interaction honeypots are not only able to collect complex exploits, they can also collect the third phase of an attack, which is how the attacker will use the compromised resource. This phase gives information regarding the motivation of the attack. For example, attacks can lead to the installation of a rogue software to provide illicit services to the attacker community, such as a botnet client, illegal file sharing or hidden remote control. Of course, high interaction honeypots should be closely monitored to learn enough of the attacker's actions while staying under control. The risk is to have an attacker being able to use the honeypot to attack external production resources. Thus, the amount of information gathered on the attack will depend on the level of control deployed in the honeypot architecture. This level of control is represented on Figure 1 as the boundary between high interaction honeypots and vanilla systems such as live hosts. This requirement to closely monitor and control honeypots directly reduces the scalability. Moreover, high interaction honeypots, even if ran on virtual machines such as VMWare, need important hardware resources.

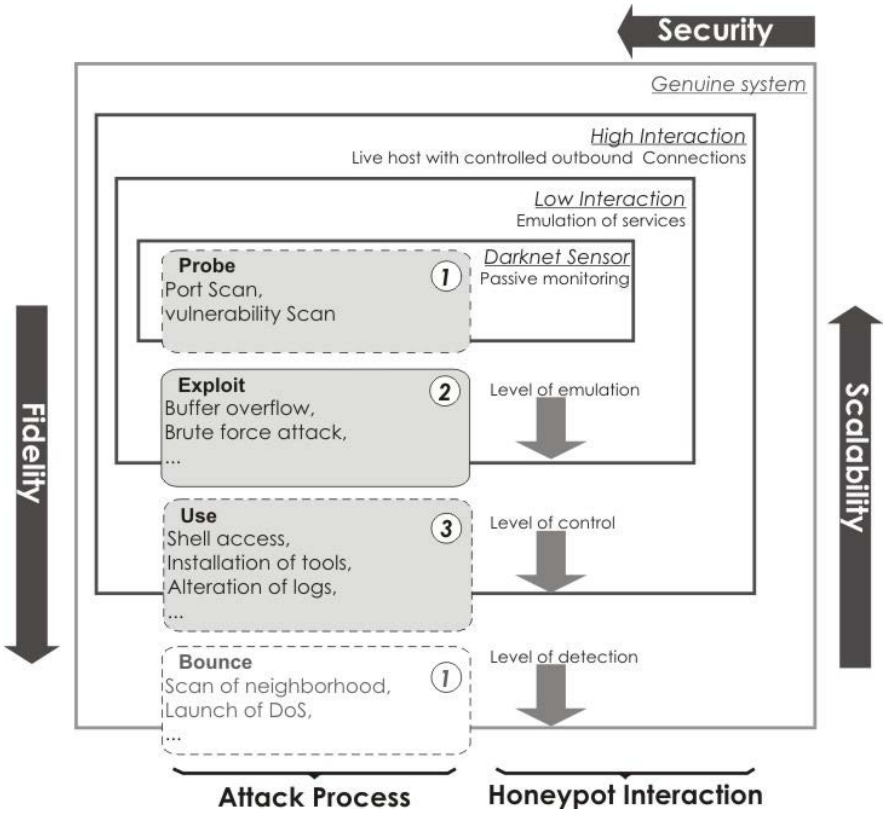


*Figure 1: Different honeypot architectures to collect different phases of network attack process*

## *3. Related Work*

This section provides a broad overview of relevant solutions to collect attack processes using passive sensors and honeypots.

### 3.1. Darknets

The idea of passively monitoring unused IP space to learn about suspicious traffic has spread through several research projects with various names. First, the Network

Telescope has pioneered the use of darknets to learn about denial of service attacks . Another project called Blackholes was able to collect traffic from 16.8 million IP addresses (1/256th of the Internet) to study global trends of worm activity. The Darknet project [29] provides a full guide to learn how to configure a darknet and start monitoring malware traffic. While darknets offer the greatest scalability of all monitoring solutions, they are greatly limited by the lack of active responders. The Internet Motion Sensor and iSink are two research projects that implemented stateless active responders to darknets.

As a result, they were able to keep a high scalability while capturing the first attack

payloads sent by attackers. These two architectures provided important discoveries on the attributes of unused IP space to understand the differences in traffic collected. They are at the transition between passive darknets and active honeypots.

### 3.2. Low Interaction Honeypots

The most widely used low interaction honeypot is Honeyd. Honeyd can create a

population of virtual hosts on a network using unassigned IP addresses. Each host can be configured with a set of emulated services and a specific operating system behavior. The simplicity and flexibility of Honeyd makes it a relevant solution to host a complete low interaction honeynet. However, attacks collected depend on the interaction provided by the emulated services, and developing these services is often a difficult challenge.

Another well-known low interaction honeypot is Nepenthes. Nepenthes was designed to automatically capture malware that spread from one computer to another. It consists of a set of emulated vulnerabilities that give enough interaction to capture the infection attempt of malware. Then Nepenthes examines the attack payload and tries to download the remaining part of the malware.

### 3.3. High Interaction Honeypots

The Honeynet Project has developed a variety of tools to help researchers deploying their honeynet and analyzing suspicious network traffic. One of these tools called Honeywall was especially designed to administrate high interaction honeypots. It provides a web interface to monitor the data collection, and a reverse

firewall to control outgoing connections from potentially compromised honeypots. Honeywall also integrates system monitoring capabilities through the Sebek kernel module.

The more cost efficient solution to host high interaction honeypots is to use virtual machines. Compared to genuine systems, virtual environments have the important advantage of being easier to monitor, to save and to clean after a successful compromise.

### 3.4. Hybrid Honeypots

The need to collect detailed attack processes on large IP spaces has pushed researchers to invent more scalable and intelligent architectures. Collapsar simplifies the deployment and administration of high interaction honeypots on large IP spaces by using GRE tunnels to route traffic from distributed networks into a centralized farm of honeypots. The limitation of Collapsar is to not provide any filtering mechanism that can prevent high interaction honeypots from being overloaded.

Another project called Potemkin is based on the idea that idle high interaction honeypots do not even need to run. As a result, the architecture saves resources by starting a new virtual machine for each active IP address. As soon as an IP address becomes inactive, the virtual machine is destroyed to save physical memory and CPU resources. Such a system allows hundreds of virtual machines to run on a single physical host.

## *4. Problem Statement*

When deploying honeypots, researchers have to precisely define three elements: a location, an architecture, and a configuration. Data collected by honeypots is critically affected by these three keys. Therefore, they need to be carefully selected. We will now detail the different problems related to each of these elements.

The **location** is the set of IP addresses used by honeypots to receive and collect network traffic. The current addressing protocol deployed on the Internet is IPv4, which is made of 4.3 billions unique addresses. The volume and the nature of attacks can greatly change from one IP address to another. Some attack threats such as the Slammer worm are globally distributed, while others such as Denials of Service target precise locations. So the location of honeypots can greatly affect the data it will receive.

Recent studies started to compare attack data from different locations and defined network characteristics such as reachability or proximity to production networks that could partially explain the differences observed. Moreover, not only the location but the size of the network of honeypots is important to collect significant attack results.

The honeypot **architecture** refers to the type of honeypot. We saw in the previous section that the different types of honeypots were governed by three attributes: fidelity, scalability and security. There is currently no solution available that offers both scalability and a high level of interaction. As a result, researchers and network operators who want to deploy honeypots cannot collect and analyze datasets which have both detailed attack processes and large network space coverage.

The **configuration** defines the set of services offered to attackers and thus the behavior of the honeypot. By set of services we mean the set of opened ports and software listening for network connections on the honeypot. These services can be emulated or real. They can be host-specific resources or vulnerabilities to study specific categories of attack. The problem when deploying honeypots in a large organization network is that there is a very large number of possible configurations to choose from.

There is currently no solution to determine whether the configuration of a network of honeypots is optimal to collect malicious threats; and to make sure that the fingerprint of the network of honeypots is small enough to prevent attackers from detecting it.

The last major issue of current honeypots is that even if they actively reply to attackers with more or less interaction, they do not allow researchers to select the type of attack they want to study. This means that because honeypots collect attacks randomly, the information collected is not often the information researchers were really looking to analyze. From such point of view, existing honeypots are collecting attack traffic passively. We believe that if honeypots adopt a more active approach when receiving illegitimate connections, they could 1) provide better results on the exact threat expected to be studied, and 2) reduce the resources spent to analyze and filter data collected.

## *5. Approach*

The purpose of our study is to develop efficient solutions to overcome current honeypot limitations. We addressed the issue of the size and the location of honeynets by correlating network flows with darknet data. We solved the problem of scalability of high interaction honeypot by implementing an advanced hybrid honeypot architecture called Honeybrid. We solved the problem of configuring honeynets in large organization network by using a server and scanner discovery program based on network flows.

Finally we addressed the challenge of cost effectively analyzing large volumes of malicious data by implementing an aggregation process that integrates network flows and honeypot data. These solutions are integrated into a complete framework to facilitate honeypot deployment and attack data analysis.The overall goals are 1) to provide to the security community an advanced honeypot solution that can be better integrated

into the landscape of security tools used by researchers and network operators, and 2) to deploy such architecture to better quantify malicious activity occurring on the campus network. The cornerstone of this architecture is a hybrid gateway that offers both advantages of high and low interaction honeypots: fidelity and scalability.

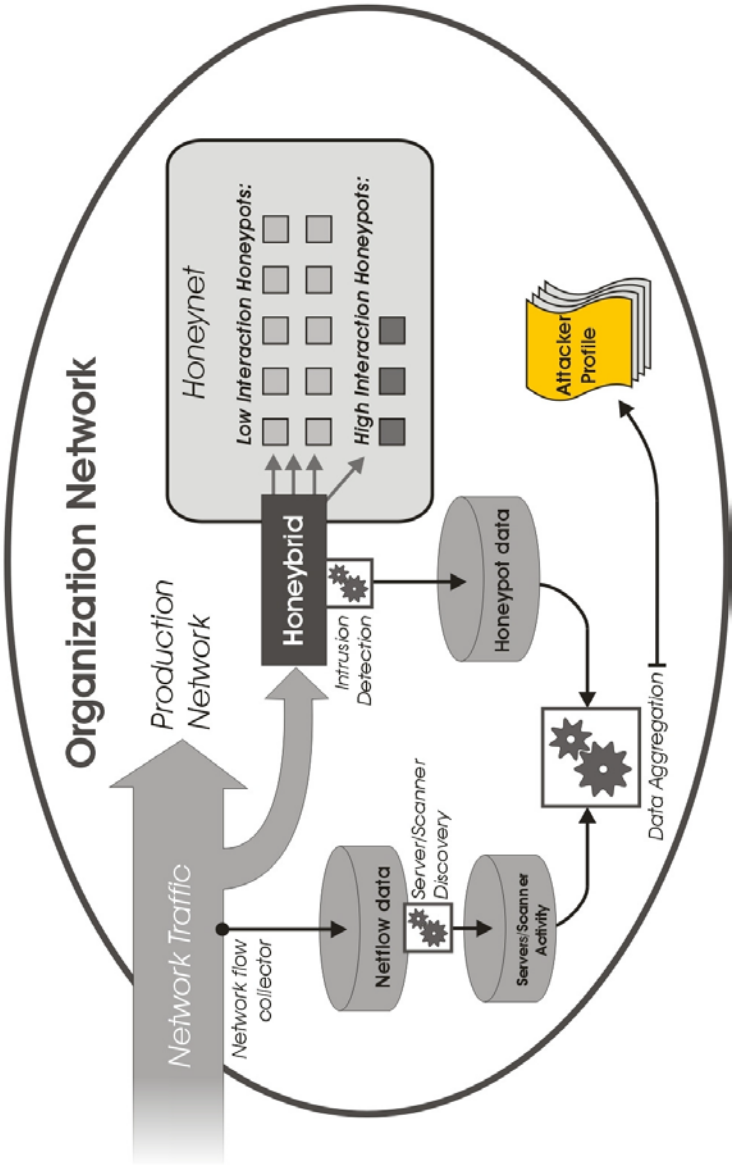


*Figure 2: Overview of our malicious traffic analysis framework*

***6. Contributions***

The contributions of this work are:

_ *Honeypot classification and mapping with attack process*: we provide a detailed classification of current honeypot solutions and we link this classification with the different phases of attack collected. We outline the different properties and limitations of honeypots.

_ *Hybrid architecture*: we describe an innovative honeypot solution that provides both a high scalability and a high level of interaction. We also introduce the concept of an attack event, to differentiate network attacks worth of analysis from the noise of malicious traffic. Our architecture is designed to be able to harvest large IP spaces while actively filtering attack events from attack traffic for detailed focused analysis.

_ *Dynamic configuration engine*: we address the problem of honeypot configuration by combining network flows and automated honeypot management. From an exhaustive monitoring of the existing attack patterns targeting the organization network, we infer the required honeypot configuration to assess the malicious activity.

_ *Architecture integration*: we provide the first open source implementation of an hybrid honeypot architecture. We integrate this architecture with network flows to provide a complete attack assessment framework. Finally, we deploy this framework and we show how it can be used to accurately detect compromised computers inside the organization network.

**Bibliography**

[1] E. Alata, V. Nicomette, M. Kaâniche, M. Dacier, and M. Herrb, "Lessons learned from the deployment of a high-interaction honeypot," *edcc*, 2014, pp. 18-20.
[2] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, "The nepenthes platform: An efficient approach to collect malware," *Lecture Notes in Computer Science*, vol. 4219, 2013, p. 165.
[3] M. Bailey, E. Cooke, T. Battles, and D. McPherson, "Tracking global threats with the Internet Motion Sensor," *32nd Meeting of the North American Network Operators Group*, 2014.
[4] M. Bailey, E. Cooke, F. Jahanian, A. Myrick, and S. Sinha, "Practical darknet measurement," *Ann Arbor*, vol. 1001, pp. 48109-2122.
[5] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson, "The Internet Motion Sensor: A distributed blackhole monitoring system," *Proceedings of the 12th Annual Network and Distributed System Security Symposium*, 2014.
[6] M. Bailey, E. Cooke, F. Jahanian, N. Provos, K. Rosaen, and D. Watson, "Data reduction for the scalable automated analysis of distributed darknet traffic," *Proceedings of the USENIX/ACM Internet Measurement Conference, New Orleans, LA*, 2015.
[7] M. Bailey, E. Cooke, D. Watson, F. Jahanian, and N. Provos, "A hybrid honeypot architecture for scalable network monitoring," *TechnicalReportCSE-TR-499-04, UniversityofMichigan*, 2014.
[8] M.D. Bailey, "A scalable hybrid network monitoring architecture for measuring, characterizing, and tracking internet threat dynamics," 2012.
[9] BaitnSwitch and Snort, *http://doc.emergingthreats.net/bin/view/Main/BaitnSwitch, 2011*.
[10] BaitnSwitch Project, *http://baitnswitch.sourceforge.net/, 2012*.
[11] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, 2013, pp. 164-177.