Complex & Intelligent Systems (2021) 7:477–494
https://doi.org/10.1007/s40747-020-00209-5

ORIGINAL ARTICLE

A computational intelligence enabled honeypot for chasing ghosts in the wires

Nitin Naik[1] · Paul Jenkins[2] · Nick Savage[3] · Longzhi Yang[4]

Received: 31 March 2020 / Accepted: 26 September 2020 / Published online: 2 November 2020

Abstract

A honeypot is a concealed security system that functions as a decoy to entice cyberattackers to reveal their information. Therefore, it is essential to disguise its identity to ensure its successful operation. Nonetheless, cyberattackers frequently attempt to uncover these honeypots; one of the most effective techniques for revealing their identity is a fingerprinting attack. Once identified, a honeypot can be exploited as a zombie by an attacker to attack others. Several effective techniques are available to prevent a fingerprinting attack, however, that would be contrary to the purpose of a honeypot, which is designed to interact with attackers to attempt to discover information relating to them. A technique to discover any attempted fingerprinting attack is highly desirable, for honeypots, while interacting with cyberattackers. Unfortunately, no specific method is available to detect and predict an attempted fingerprinting attack in real-time due to the difficulty of isolating it from other attacks. This paper presents a computational intelligence enabled honeypot that is capable of discovering and predicting an attempted fingerprinting attack by using a Principal components analysis and Fuzzy inference system. This proposed system is successfully tested against the five popular fingerprinting tools Nmap, Xprobe2, NetScanTools Pro, SinFP3 and Nessus.

Keywords Cyberattack · Honeypot · Computational intelligence · Fingerprinting attack · Principal components analysis · Fuzzy inference system

Introduction

Security experts adapted their strategy due to the significant increase in cyberattacks, in particular, the increase in their complexity and resolution; which led to the application of both active and passive defence systems as a part of their defensive strategies [8]. As an active defence system, a honeypot functions as a decoy to entice cyberattackers to reveal information which can be utilised by security experts in updating their security procedures [28]. As a concealed system, it is essential to disguise its identity for its successful operation. Nonetheless, cyberattackers always attempt to uncover these honeypots and one of the most effective techniques for revealing their identity is a fingerprinting attack. Generally, for any unconcealed system fingerprinting is not of great concern, but for a honeypot it may be end of its life, resulting in significant consequences, for example, it can be exploited as a zombie by an attacker to attack others [35].

A Honeypot can be protected from a fingerprinting attack, however, this is not consistent with the principle of a honeypot, which is established with the purpose of gaining information about attackers. It would be beneficial if an attempted fingerprinting attack can be predicted timely. Unfortunately, no specific method is available to detect and predict an attempted fingerprinting attack in real-time as it is

✉ Nitin Naik
n.naik1@aston.ac.uk

Paul Jenkins
pjenkins2@cardiffmet.ac.uk

Nick Savage
nick.savage@port.ac.uk

Longzhi Yang
longzhi.yang@northumbria.ac.uk

1 School of Informatics and Digital Engineering, Aston University, Birmingham, UK

2 Cardiff School of Technologies, Cardiff Metropolitan University, Cardiff, UK

3 School of Computing, University of Portsmouth, Portsmouth, UK

4 Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, UK

challenging to distinguish it from other attacks. Therefore, this paper presents a Computational Intelligence (CI) enabled honeypot that is capable of discovering and predicting an attempted fingerprinting attack by using a principal components analysis (PCA) and fuzzy inference system (FIS). The proposed CI-enabled design is focused on the most common Operating System (OS) fingerprinting attack, which is performed on the target system to obtain specific information regarding the OS, services, device type and type of architecture [3]. The mechanism used is to send a stream of fabricated TCP/IP packets by an attacker to prompt a response in the form of TCP/IP packets containing fingerprint information of the target system [33]. Conversely, the proposed CI-enabled honeypot analyses this stream of TCP/IP packets sent by an attacker to obtain signs of an attempted fingerprint attack on the honeypot.

Initially, this paper performs a simulation of fingerprinting attacks on the honeypot to collect attack data (TCP/IP packets). The simulation is accomplished by employing a *KFSensor* honeypot, and *Nmap* and *Xprobe2* fingerprinting tools. The attack simulation data is captured in two different logs by the KFSensor honeypot and Wireshark analyser for forensic analysis. Subsequently, based on preliminary observations and empirical evidence, a number of important fields of collected TCP/IP packets are analysed to ascertain abnormalities or patterns as an indication of an attempted fingerprinting attack. Successively, it applies PCA to determine the most influential fields, which can be further utilised to develop an effective approach to predict the fingerprinting attack. Later, it proposes an FIS to correctly correlate the identified influential fields by the PCA and predict an attempted fingerprinting attack and its severity level on the honeypot. Finally, the proposed system is successfully tested against the five popular fingerprinting tools. This testing includes two previous tools Nmap and Xprobe2 and three new tools NetScanTools Pro, SinFP3 and Nessus; which were not involved in the development of the CI-enabled honeypot.

The rest of the paper is organised into the subsequent sections: "Background information" explains about honeypots, fingerprinting attack and an OS fingerprinting attack. "Simulation of various types of OS fingerprinting attacks" describes the fingerprinting attack simulation on the honeypot for the collection of attack data and its detailed analysis. "Identifying abnormalities/patterns in the various fields of TCP/IP protocols as signs of os fingerprinting attack" discusses a comprehensive examination of the chosen TCP/IP fields and their related abnormalities/patterns as signs of a fingerprinting attack. "Design and development of computational intelligence enabled honeypot for predicting fingerprinting attacks on honeypots" presents the design and development of a CI-enabled honeypot for predicting fingerprinting attacks on honeypots. For this design it performs PCA on the selected TCP/IP fields to establish the most significant fields to predict fingerprinting attacks on honeypots. Subsequently, it proposes a fuzzy inference system to predict fingerprinting attacks and their severity levels on honeypots. Finally, "Conclusion" concludes the paper and discusses the possible future enhancement of the CI-enabled honeypot.

Background information

Honeypots

A honeypot is a concealed security system that functions as a decoy to entice cyberattackers to reveal their information [31]. It deceives, detects and diverts cyberattackers, whilst contemporaneously gathering their information [11]. Honeypots are designed to represent themselves as a potential target for cyberattackers; subsequently, deployed in an isolated manner and closely monitored to uncover vulnerabilities and new attacks to the network and to develop an enhanced defensive strategy [37]. Most honeypots are used to imitate the functionalities of a real network to entice cyberattackers to attack them assuming them to be a real network and revealing their information.

Honeypots are categorized into different categories on the basis of their design and level of interaction with cyberattackers: low-interaction honeypots, medium-interaction honeypots and high-interaction honeypots [1,29,36]. Low-interaction honeypots normally imitate real systems and have restricted communication with cyberattackers; whereas, high-interaction honeypots are normally real systems that have unrestricted communication with cyberattackers [38]. Medium-interaction honeypots have greater ability to communicate with cyberattackers than low-interaction honeypots, however, they have less functionality than high-interaction honeypots [7]. Honeypots can be a crucial active defence tool for organisations or researchers in an attempt to discover advanced attacks and related techniques which are not possible through other security tools; however, for effective operation, additional cost and skills are required in their design and management. Honeypots are effective security systems but should not be used as the only defensive system or an alternative to replace other security systems to protect the network.

Fingerprinting attack

In a fingerprinting attack, an attacker usually sends a sequence of fabricated packets to the target system to provoke a response in the form of packets containing fingerprint information with the intention of its identification. Fingerprinting attacks are categorized into two categories on the basis of the activity of cyberattackers: active and passive fingerprinting attacks. In an active fingerprinting attack, cyberattackers send

carefully constructed packets to the target system, analysing their response packets to extract fingerprinting information [31]. In a passive fingerprinting attack, cyberattackers do not send any packets to the target system rather they sniff, capture and analyse traffic from the target system to extract fingerprinting information [31]. Active fingerprinting attacks are more accurate than passive fingerprinting attacks as the result is based on the direct response from the target system. Therefore, in this design of CI-enabled honeypot, only an active fingerprinting attack is considered.

OS fingerprinting attack

OS fingerprinting attacks are the most prevalent fingerprinting attack, which is performed on the target system to obtain specific information regarding the OS, services, device type and type of architecture [3]. The mechanism is to send a stream of fabricated TCP/IP packets from the attacker to elicit response TCP/IP packets containing fingerprint information of the target system [33]. After analysing a number of the fields of certain TCP/IP protocols of the response packets, a fingerprint is constructed and compared against the fingerprint database to find the exact or closest matched fingerprint of the target system. Cyberattackers are highly successful in performing OS fingerprinting attacks as the same TCP/IP protocol suite is implemented by every OS distinctively, as a result, distinct responses are produced for the same TCP/IP query. Consequently, different responses generated by different operating systems divulge substantial information about that system to cyberattackers. The complete process of an OS fingerprinting attack is dependent on TCP/IP protocol suite; therefore, it is sometimes referred to as TCP/IP stack fingerprinting. After obtaining precise information about the OS and target system, cyberattackers can launch more complex attacks with greater severity against the target system.

Simulation of various types of OS fingerprinting attacks

Each OS implements the TCP/IP protocol suite distinctively, therefore, acquiring a fingerprint of any OS requires an analysis of the TCP/IP packets sent by that OS, which can offer significant information to construct an accurate fingerprint of that OS. This process of finding a fingerprint of a particular OS is primarily based on the analysis of the TCP, ICMP and UDP protocols as every fingerprinting tool/technique sends and receives these three protocol-based packets to the target machine. However, certain fingerprinting tools/techniques primarily employ TCP packets to perform the fingerprinting attack and certain primarily employ ICMP packets, thus, the development of any successful method to detect this attack should involve examination of both categories (TCP and ICMP) of tools/techniques. Thus, this simulation covers both TCP-based and ICMP-based OS fingerprinting attacks to acquire the OS fingerprint (and information about associated services) of a honeypot system for resolving its identity. This information about OS and associated services of a honeypot system would assist an attacker to identify and exploit that honeypot system and possibly transform it into a zombie to attack others. The simulation of all OS fingerprinting attacks are accomplished by employing a *KFSensor* honeypot, and *Nmap* and *Xprobe2* fingerprinting tools. *KFSensor* is a commercially available graphical user-interface honeypot for the Windows platform [14], *Nmap* is TCP-based and *Xprobe2* is an ICMP-based OS fingerprinting tool. This experimental simulation generated fingerprinting data for analysis and, abnormalities and patterns detection as symptoms of an OS fingerprinting attack.

TCP-based OS fingerprinting attack using Nmap

Nmap is the most powerful and reliable scanning tool which is very effective in performing an OS fingerprinting attack (mainly TCP-based). Many Nmap scripts use heuristics and fuzzy signature matching to reach conclusions about the target host OS or services [19]. During an OS fingerprinting attack, Nmap sends a stream of TCP/IP packets (approximately 16 or more), to identified open and closed ports on the target machine [18]. This stream of TCP/IP packets contains TCP, UDP, and ICMP packets, however, this counting does not include all the retransmitted packets. These packets/probes are aimed at several existing ambiguities and their exploitation in the standard protocol Request for Comments (RFCs). When the target machine sends a reply back to the Nmap machine for these packets/probes, Nmap analyses values of various parameters of TCP, ICMP and UDP packets and constructs an OS fingerprint to match against its database of OS signatures [10]. Depending on the OS signature matching result, it predicts the possible OS of the target machine, when there is no exact match it can use its fuzzy technique to predict the result [15].

Table 1 shows the five different Nmap scripts for an OS fingerprinting attack. The first Nmap script is the basic OS fingerprinting command that reveals the OS fingerprint and several other details such as OS version numbers, device type and architectural information [20]. The second Nmap script offers more descriptive fingerprinting information such as OS type, device type, host script and traceroute. The third Nmap script utilises the fuzzy approach to predict the closest matched OS (in percentages) in event that it does not find exact match [15]. The fourth Nmap script is used to perform OS fingerprinting continuously for the given number of attempts to improve the accuracy of prediction. The fifth and last Nmap script is completely different from other four scripts and utilises a different signature database for

Table 1 Nmap OS fingerprinting attack scripts

No.	Nmap OS Fingerprinting Attack Script
1	$nmap\ -O\ <Honeypot\ IP>$
2	$nmap\ -A\ <Honeypot\ IP>$
3	$nmap\ -O\ --fuzzy\ --osscan-guess\ <Honeypot\ IP>$
4	$nmap-O--max-OS-tries\ n\ <HoneypotIP>$
5	$nmap-sV--version-intensity\ n\ <HoneypotIP>$

Table 2 Xprobe2 OS fingerprinting attack scripts

No.	Xprobe2 OS fingerprinting attack script
1	$xprobe2\ <Honeypot\ IP>$
2	$xprobe2-D/-M\ <ModuleName><HoneypotIP>$
3	$xprobe2\ -B\ <Honeypot\ IP>$
4	$xprobe2-T/-U\ <Port\ Range>\ <HoneypotIP>$
5	$xprobe2\ -p\ <Protocol:Port:Status>\ <Honeypot\ IP>$

matching any fingerprint. It discovers information relating to various services running on different ports such as HTTP, FTP, SMTP, SSH, Telnet and DNS. This script can be executed with different intensity from 0 to 9, where 9 is the highest intensity which improves the accuracy of prediction [17], [16]. The first four Nmap scripts use Nmap database called *nmap-os-db* [22], and the fifth Nmap script uses Nmap database called *nmap-services* [21]. For accuracy and to discount any outlier data, each Nmap script (with various sub-options) is executed 100 times to record the results in various network conditions and observe the retransmitted packet pattern.

ICMP-based OS fingerprinting attack using Xprobe2

Nmap is a powerful and reliable fingerprinting tool, however, its results largely rely on TCP packets; consequently, an ICMP-based fingerprinting simulation and analysis is essential to propose a generic solution. Xprobe2, is one of the first ICMP-based fingerprinting tools, which utilises ICMP packets and is based on the signature engine and fuzzy signature matching [5]. During an OS fingerprinting attack, Xprobe2 sends a stream of TCP/IP packets (approximately 10 or more), to identified open and closed ports on the target machine [40]. This stream of TCP/IP packets contains ICMP, TCP, and UDP packets, not including all the retransmitted packets. Xprobe2 consists of 13 modules, and Xprobe2++ or Xprobe2-ng consists of an additional 3 modules (fingerprint: icmp_info, app: ftp, app: http), which it utilises to find an OS fingerprint [40]. This tool is both more effective and quicker than Nmap due to the utilisation of fewer number of TCP/IP packets. Nevertheless, it is obsolete and not updated, as a result of this, it is not able to ascertain latest OSs including Windows 7 on the honeypot system [2]. Nonetheless, this paper is focused on the counter strategy of identifying and predicting an OS fingerprinting attack, and for this, Xprobe2-based simulation is imperative to analyse the ICMP-based OS fingerprinting attack as it is one the very first ICMP-based OS fingerprinting tools and the basis for all ICMP-based tools.

Table 2 shows the five different Xprobe2 scripts for an OS fingerprinting attack. The first Xprobe2 script is a basic OS fingerprinting command that determines a fingerprint of an OS running on an intended system as per its basic operation [5]. The second Xprobe2 script determines a fingerprint of an OS depending on the utilisation of specific modules, which can provide different results based on the selected modules. The third Xprobe2 script determines a fingerprint of an OS by sending more traffic to an intended system because parameter *-B* sends consecutive TCP handshake requests to any open TCP port such as 80, 443, 23, 21, 25, 22, 139, 445 and 6000 on an intended system and expects a SYN ACK reply [6]. The fourth Xprobe2 script determines a fingerprint of an OS by utilising an internal port scanning module that performs a port scanning of indicated TCP and/or UDP port(s) [6]. The fifth Xprobe2 script determines a fingerprint of an OS by utilising additional details regarding a protocol, port and the current status via parameter *-p*. The protocol can be chosen from TCP or UDP, the port number from 1 to 65,535, and the current status (Open or Closed) of a port. In case of a closed port, an intended system may reply with RST packet for a TCP port, and may reply with ICMP Port Unreachable packet for a UDP port. In case of an open port, an intended system may reply with SYN ACK packet for a TCP port, and may not reply (send a packet) for a UDP port [6]. Similar to the Nmap experiment, to obtain accurate results and removing any outliers in the data, each Xprobe2 script (with various sub-options) is executed 100 times to record the results in various network conditions and observe the pattern of retransmitted packets.

Identifying abnormalities/patterns in the various fields of TCP/IP protocols as signs of OS fingerprinting attacks

The experimental simulation data for both TCP and ICMP based OS fingerprinting attacks collected in the previous section is analysed in this section. Each stream of TCP/IP packets received from the attacker is analysed to reveal any observed abnormalities/patterns in the various fields of TCP/IP protocols (i.e. TCP, ICMP, UDP and IP) [34]. This analysis identifies the ten indicator fields of TCP/IP protocols based on their detected discrepancies in the attack simulation data. These ten indicator fields mostly include TCP and

IP fields as shown in Figs. 1 and 5. Additionally, these ten TCP/IP fields are analysed to emphasis their weight based on the literature and the core attack principles of popular OS fingerprinting tools/techniques.

Discovering abnormalities/patterns in TCP Flags

TCP is comprised of six standard flags (SYN, ACK, URG, PSH, RST, FIN) that controls the nature and flow of the transmission. There are several flags or combination of flags which are considered as illegal/abnormal flags based on the RFCs of TCP, however, it does not explain the handling of such illegal/abnormal flags. As a result, it is managed by the OS and thus, different OSs generate different responses for an illegal/abnormal flag or combination of flags. This is a significant concern for the security community as attackers exploit these responses to determine the OS of the target machine. A number of these illegal/abnormal TCP flags can be utilised as a good indicator of an OS fingerprinting attack, which relatively straightforward to find as they are renowned. Some OS fingerprinting tools utilise additional control flags (CWR, ECN) and three Reserved Bits in their attack techniques. This analysis includes all the possible illegal/abnormal flags or combination of flags which can be an indication of an OS fingerprinting attack. The inclusion of these additional flags are to ensure that the proposed approach is a generic approach, however it explains the findings of the experiment regarding illegal/abnormal flags.

URG/PSH/FIN probing

This is one of the well-known abnormal flag combinations (called Xmas probe), which exploits flaws in the TCP RFC 793 to determine the open and closed ports. This URG/PSH/-FIN probing is only effective with those operating systems that conform to the TCP RFC 793. Nevertheless, an attacker can send this URG/PSH/FIN packet to understand the status of any port. Upon receiving this packet on the port of the target machine, a specific OS-based response is generated by that machine, which can give an indication the OS of a target machine. This is an important probe in an OS fingerprinting attack; however, it is an abnormal flag combination that can be combined with other indicators as a sign of an OS fingerprinting attack. The captured TCP packet with URG/PSH/FIN probing during an OS fingerprinting attack is shown in Fig. 2.

NULL packet

The NULL probe is another abnormal packet wherein no flag is set but contains a packet sequence number. Nevertheless, an attacker can send this NULL packet to understand the status of any port. Upon receiving this packet on the port of the target machine, a specific OS-based response is generated by that machine, which can give an indication of the OS of a target machine. This is an important probe in an OS fingerprinting attack; however, it is an abnormal flag combination that can be combined with other indicators as a sign of an OS fingerprinting attack. The captured TCP packet with NULL probing during an OS fingerprinting attack is shown in Fig. 3.

Reserved Bit Probing

There are 3 reserved bits in the TCP header for future use. These reserved bits should not be used and are always set to zero. The captured TCP packet utilising reserved bits during an OS fingerprinting attack is shown in Fig. 4. This symptom can be combined with other indicators as a sign of an OS fingerprinting attack.

ECN-echo probing

The Explicit Congestion Notification (ECN) flag offers added functionality for notifying hosts about network congestion without dropping packets. It is an additional feature that may be used for the two hosts if they are ECN-enabled. The captured TCP packet with ECN-Echo probing during an OS fingerprinting attack is shown in Fig. 4. This symptom can be combined with other indicators as a sign of an OS fingerprinting attack (Fig. 5).

FIN Probing

The FIN flag is used to close the connection and, it should only be sent when the connection was initiated previously. Therefore, this FIN packet is violating the rules of TCP that would never occur in the real world. Nevertheless, an attacker can send this FIN packet as an unconnected packet for knowing the status of any port. Upon receiving this packet on the port of the target machine, a specific OS-based response is generated by that machine, which can give an indication of the OS of a target machine. This is a very important probe in an OS fingerprinting attack; however, it is an abnormal flag that can be combined with other indicators as a sign of an OS fingerprinting attack.

SYN/FIN probing

This pair of flags are reciprocally exclusive and usually not used in the same packet. Therefore, this SYN/FIN packet is violating the rules of TCP that would never occur in the real world. Nevertheless, an attacker can send this SYN/FIN packet to understand the status of any port. Upon receiving this packet on the port of the target machine, a specific OS-based response is generated by that machine, which can

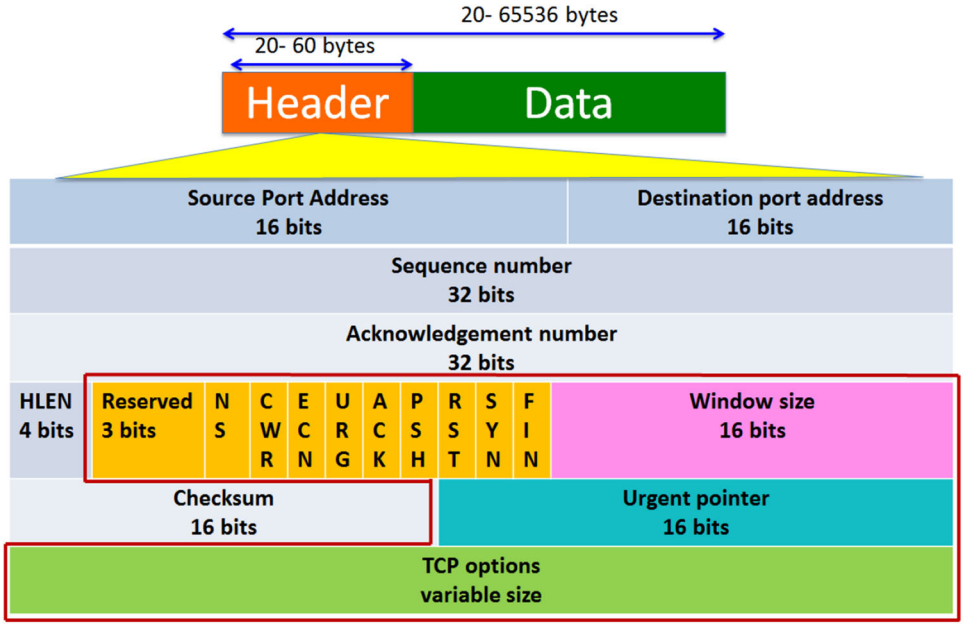

Fig. 1 Investigated fields of TCP Header for the attempted OS fingerprinting attack

```
Flags: 0x029 (FIN, PSH, URG)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Nonce: Not set
   .... 0... .... = Congestion Window Reduced (CWR): Not set
   .... .0.. .... = ECN-Echo: Not set
   .... ..1. .... = Urgent: Set
   .... ...0 .... = Acknowledgment: Not set
   .... .... 1... = Push: Set
   .... .... .0.. = Reset: Not set
   .... .... ..0. = Syn: Not set
   .... .... ...1 = Fin: Set
```

Fig. 2 Captured TCP packet with URG/PSH/FIN probing during an OS fingerprinting attack

```
Flags: 0x000 (<None>)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Nonce: Not set
   .... 0... .... = Congestion Window Reduced (CWR): Not set
   .... .0.. .... = ECN-Echo: Not set
   .... ..0. .... = Urgent: Not set
   .... ...0 .... = Acknowledgment: Not set
   .... .... 0... = Push: Not set
   .... .... .0.. = Reset: Not set
   .... .... ..0. = Syn: Not set
   .... .... ...0 = Fin: Not set
```

Fig. 3 Captured TCP packet with NULL probing during an OS fingerprinting attack

```
Flags: 0x8c2 (SYN, ECN, CWR, Reserved)
    100. .... .... = Reserved: Set
    ...0 .... .... = Nonce: Not set
    .... 1... .... = Congestion Window Reduced (CWR): Set
    .... .1.. .... = ECN-Echo: Set
    .... ..0. .... = Urgent: Not set
    .... ...0 .... = Acknowledgment: Not set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
```

Fig. 4 Captured TCP packet with Reserved Bit and ECN-Echo probing during an OS fingerprinting attack

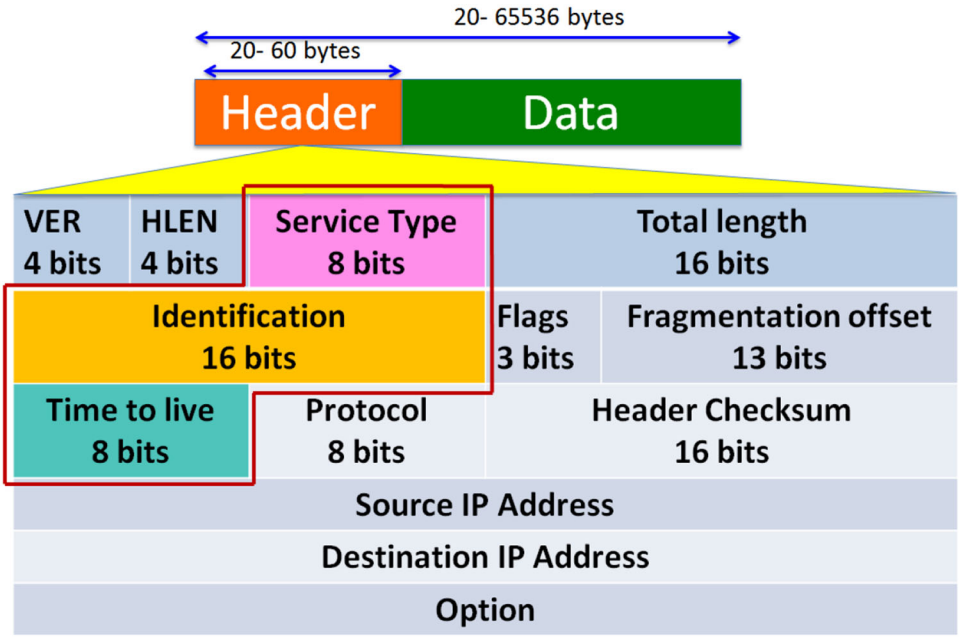

Fig. 5 Investigated fields of IP Header for the attempted OS fingerprinting attack

give an indication of the OS of a target machine. This is an important probe in an OS fingerprinting attack; however, it is also an abnormal flag combination that can be combined with other indicators as a sign of an OS fingerprinting attack.

Discovering abnormalities/patterns in TCP Options

The majority of fingerprinting tools utilise TCP Options field of the TCP header because it is an adaptable field and can be of any size from 0 to 40 bytes. The TCP options field may contain some or all attributes: Maximum Segment Size (MSS), Window Scaling, Selective Acknowledgements (SACK), Timestamps, and Nop. Therefore, every OS customises this TCP Options field based on its implementation which can be identified as a pattern of that OS. Conversely, the TCP options field can be used to identify an OS fingerprinting attack by finding abnormalities/patterns in the packets received from an attacker. This can be combined with other indicators as a sign of an OS fingerprinting attack.

Discovering abnormal/frequent uses of TCP urgent pointer

TCP provides the facility to mark certain amount of data as urgent, which is indicated by setting the URG flag. This Urgent Pointer field indicates how much of the data in the segment is urgent. This field and URG flag jointly allow an application to forward urgent data immediately by creating a secondary out of band channel without waiting in sequential send queue. Nonetheless, most users are uncertain

Table 3 IP service type/Type of service (TOS) specifications

Bit positions	Descriptions
Bits 0–2	Precedence (e.g., 000 (0) - Routine, 001 (1) - Priority, 010 (2) - Immediate)
Bit 3	Delay (0 = Normal Delay, 1 = Low Delay)
Bit 4	Throughput (0 = Normal Throughput, 1 = High Throughput)
Bit 5	Reliability (0 = Normal Reliability, 1 = High Reliability)
Bit 6	Cost (0 = Normal Cost, 1 = Low Cost)
Bit 7	Reserved for Future Use

about using this field correctly. Thus, this ambiguity offers a possible opportunity to attackers to exploit this field for a fingerprinting attack. At the same time, the improper use of this Urgent Pointer may reveal a potential OS fingerprinting attack.

Discovering abnormalities/variations in TCP Window Size

TCP Window Size is important field to decide the total amount of bytes that can be sent successfully without waiting for an acknowledgement. TCP Window Size is maintained by both sender and receiver due to the bidirectional nature of TCP, however, fixed limit is determined by receiver [27,32]. This field is mainly used for network troubleshooting, application baselining or preventing network congestion at the receiver end. This is the important field for flow control and could be exploited for a fingerprinting attack. Equally, this TCP Window Size can be looked at for finding substantial discrepancies and repetitive cases of zero windows that could reveal a potential OS fingerprinting attack.

Discovering abnormal/frequent uses of IP service type/Type of service (TOS) Field

This is an IP datagram field that is used to describe its various quality of services. It is an 8-bit field consisting of several quality parameters, namely, Precedence, Speed, Throughput, Reliability and Cost as shown in Table 3 [4]. Some of the QoS parameters may not be frequently used in regular communications; therefore, their frequent or anomalous use may reveal irregular actions and perhaps the probability of an OS fingerprinting attack.

Discovering abnormalities/commonalities in IP identification (IPID) field

In a TCP/IP network, the maximum size of a datagram is limited to the processing capacity of that network, which is called the Maximum Transmission Unit (MTU). Therefore, the successful data transmission process requires fragmentation of all those datagrams, which are greater than the MTU. The IPID field facilitates fragmentation (and later reassembly) of IP datagrams with a unique ID, which is incremented whenever an IP datagram is sent from source to the destination. This IPID is used to reassemble all fragmented IP datagrams (which will have the same IPID) at the receiver end. The exact order of the fragmented datagrams during the reassembly is determined by the fragment offset. The More Fragments (MF) flag is used to determine if fragmentation is allowed, and whether more fragments are pending. Similarly, the Don't Fragment (DF) flag is used to deny fragmentation, resulting the drop of packets greater than the MTU size.

The updated specification of the IPID Field (RFC 6864) states that it must not be utilised for any purpose other than fragmentation (and reassembly) [39]. However, it is not uncommon to set its value to zero while using it for numerous pings, and for numerous SYN-ACKs from the same source. Irrespective of IPID standard guidelines, its implementation is still ambiguous, which leads to its exploitation by attackers for various types of attacks and possibly a fingerprinting attack. Similarly, this field can be analysed for various sequences of IPID or commonality of fragmented packets of the same IPID number for finding a sign an OS fingerprinting attack.

Discovering abnormalities in IP time-to-live (TTL) value

The IP TTL field is used to determine the lifetime of an IP datagram in the network. It can be defined as a counter or timestamp and once it is elapsed, the corresponding IP datagram is discarded or revalidated. This field was added to the IP header to restrict the time an IP datagram can spend on any network due to the connectionless nature of IP. This field can be exploited to perform various kinds of attacks including an OS fingerprinting attack, where an abnormal TTL value or a TTL value of less than or equal to one can be used. Conversely, these TTL abnormalities may provide a sign of an OS fingerprinting attack.

Discovering abnormalities/patterns in UDP Requests

UDP is a very useful protocol in many probing techniques due to its connectionless nature. All OS fingerprinting tools use UDP packets in conjunction with TCP and/or ICMP packets to collect fingerprinting information from the target machine. An attacker sends UDP packets to a port of the target machine and may or may not receive response depending on the open/closed port. The target machine replies with an ICMP error message- *Destination Unreachable (ICMP Type 3)* if the port is closed, otherwise, receives no reply for

an open or filtered port. Generally, the UDP packet used in OS fingerprinting is either empty or set to a fixed payload. An attacker can also set IP DF flag in the UDP packet that can prompt the target machine to reply with an ICMP error message. These symptoms can be found in the UDP packets received from an attacker to identify an OS fingerprinting attack. This can be combined with other indicators as a sign of an OS fingerprinting attack.

Discovering abnormalities/patterns in ICMP requests

ICMP is an error announcing protocol that is used for troubleshooting, control and error message services. It is used by network devices (e.g. routers, gateways, hosts) to announce error messages when there is an issue in delivering packets. As a result of this, an attacker can use legitimate ICMP request packets, ICMP Echo Request (Type 8), ICMP Router Solicitation Request (Type 10), ICMP Timestamp Request (Types 13), ICMP Information Request (Type 15-Deprecated) and ICMP Address Mask Request (Type 17-Deprecated) to collect significant information about an OS of the target machine [25,26,30]. However, most OS fingerprinting tools/techniques utilise abnormal ICMP requests by changing some of the parameters of these ICMP requests. For example, an abnormal ICMP Echo request (Type 8) can be easily determined by examining its *Code* value which should always be *Code* 0, however, some OS fingerprinting tools use the invalid *Code* value in their attacks. These abnormalities can be found in the ICMP request packets received from an attacker to identify an OS fingerprinting attack. This can be combined with other indicators as a sign of an OS fingerprinting attack.

Discovering abnormalities/patterns in ICMP packet size

ICMP packets are normally used to report errors in the standard format and therefore, their size is relatively stable with respect to particular OS, and it is in the predictable range [25,26,30]. When the common size of an ICMP packet is determined as a network baseline, it is relatively straightforward to compare normal and abnormal ICMP packets without investigating their contents in a detailed way. For example, in an Nmap-based experimental simulation, the baseline size was 74 bytes (i.e. most common ICMP packet size in Windows), and the size of collected ICMP packets by KFSensor Honeypot was 149 and 179 respectively. The recorded size of these two ICMP packets for all the Nmap experimental iterations was the same. This is one clear indication of pattern/abnormality found in the ICMP request packets received from an attacker to identify an OS fingerprinting attack. This can be combined with other indicators as a sign of an OS fingerprinting attack.

Design and development of computational intelligence enabled honeypot for predicting fingerprinting attacks on honeypots

The design of Computational Intelligence (CI) enabled honeypots to utilise two approaches, namely a Principal Components Analysis (PCA) and a Fuzzy Inference System (FIS). The PCA is utilised to determine only the most influential TCP/IP fields from the previously observed several TCP/IP fields, which can be further utilised to develop an effective approach to predict an OS fingerprinting attack. Then FIS is designed to utilise and correctly correlate the identified influential fields by PCA and predict an attempted OS fingerprinting attack and its severity level on the honeypot. The complete working procedure of this CI-enabled honeypot is shown in Fig. 6.

Principal components analysis to determine the most influential TCP/IP fields for predicting fingerprinting attacks on honeypots

Principal components analysis is one of the most effective computational techniques for dimensionality reduction by feature extraction while retaining most of the information. The primary reasons for the preferred choice of PCA over other techniques are:

- A very efficient technique for smaller dimensions which is the case here
- The decreased requirements for capacity and memory which makes the proposed design, a lightweight system
- The low noise sensitivity which is a great advantage for the volatile network traffic
- It uses simple statistical calculations which is available with most of the ordinary tools that avoids the need of complex programming or machine learning tasks
- A lack of redundancy of data due to orthogonal components
- A synchronized low-dimensional representation of the variables

Based on the comprehensive research on the exploitation of various TCP/IP fields in several attacks, and subsequent experimental simulation of an OS fingerprinting attack in this work concluded the ten evidential TCP/IP fields that may reveal an OS fingerprinting attack. To aid prediction of an OS fingerprinting attack, it is worthwhile to select only the most significant fields out of the ten chosen fields and also establish their corresponding relationships with each other. This can

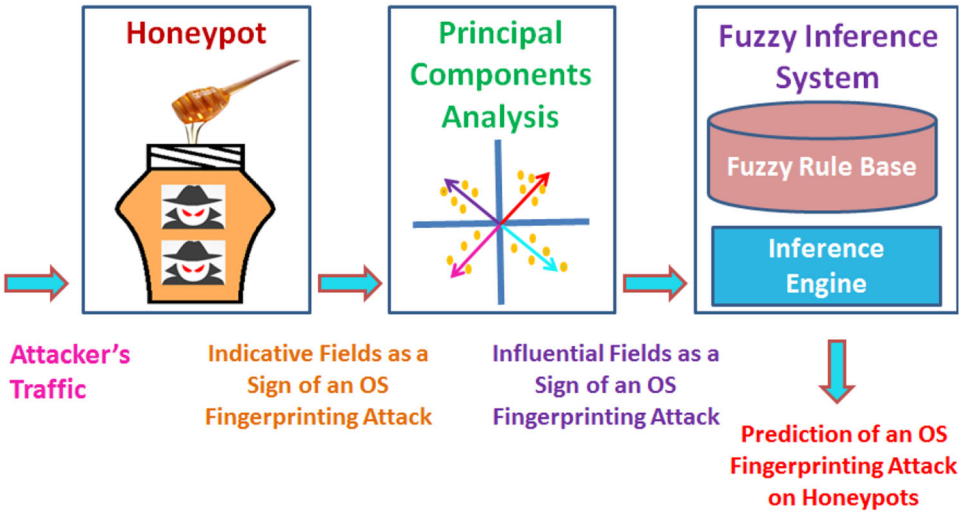

Fig. 6 Computational Intelligence enabled honeypot for predicting OS fingerprinting attacks on honeypots

Table 4 Principal components analysis of targeted TCP/IP fields of collected fingerprinting data

Importance of components	Standard deviation	Proportion of variance	Cumulative proportion
Principal Component1 (PC1)	1.8632531	0.3471712	0.3471712
Principal Component2 (PC2)	1.3222377	0.1748313	0.5220025
Principal Component3 (PC3)	1.2714464	0.1616576	0.6836601
Principal Component4 (PC4)	1.0285482	0.1057911	0.7894512
Principal Component5 (PC5)	0.9629541	0.0927281	0.8821793
Principal Component6 (PC6)	0.7542089	0.0568831	0.9390624
Principal Component7 (PC7)	0.5752591	0.0330923	0.9721547
Principal Component8 (PC8)	0.5200373	0.0147899	0.9869446
Principal Component9 (PC9)	0.3845766	0.0111769	0.9981216
Principal Component10 (PC10)	0.137055	0.0018784	1.0000000

be accomplished using a PCA, where principal components with higher variances will be considered the best components, showing extra information about the data. Based on this analysis, only the best components are selected for the subsequent analysis as they practically signify the complete data, and rest of the components can be ignored based on the pre-decided threshold values, namely, Cumulative Proportion of Variance, Eigenvalue and/or Loading (contribution of each variable to the principal component). The traditionally accepted threshold values considered for this experiment are: $Cumulative\ Proportion\ of\ Variance$ >85%, $Eigenvalue$ >1 (from Kaiser's rule [13]) and $Loading$ for any variable should be relatively higher than other variables or at least $Loading^2 > 1/Total\ \ Number\ of\ Variables$ [9], [41], [12]. Prior to the PCA, data profiling of the collected data is required, converting mostly categorical fields into numerical fields for the analysis purposes.

Table 4 illustrates the standard deviation, variances and cumulative proportion of variances for the chosen ten principal components. The cumulative proportion of variance for the first five components is 0.8821793 ($\approx$ 88%), which is higher than the pre-decided threshold value of 85%, and thus, this PCA analysis suggests, the first five components are the best components based on the collected attack simulation data. The contribution of the rest five components to the data is very low because their cumulative value is only around 12%. Nonetheless, it is also crucial to further evaluate the first five best components, such as examining their eigenvalues >1 (see Fig. 7), which is true for the first four components, however, the fifth component is marginally smaller than 1 ($\approx$ 1), but the inclusion of the fifth component is crucial to constitute 85% value of cumulative proportion of variance as mentioned earlier [12].

The final evaluation and selection of influential variables are additionally based on the $Loading$, which shows the correlation between an original variable and a principal component. In this analysis, the $Loadings$ of the first five most significant principal components are computed as

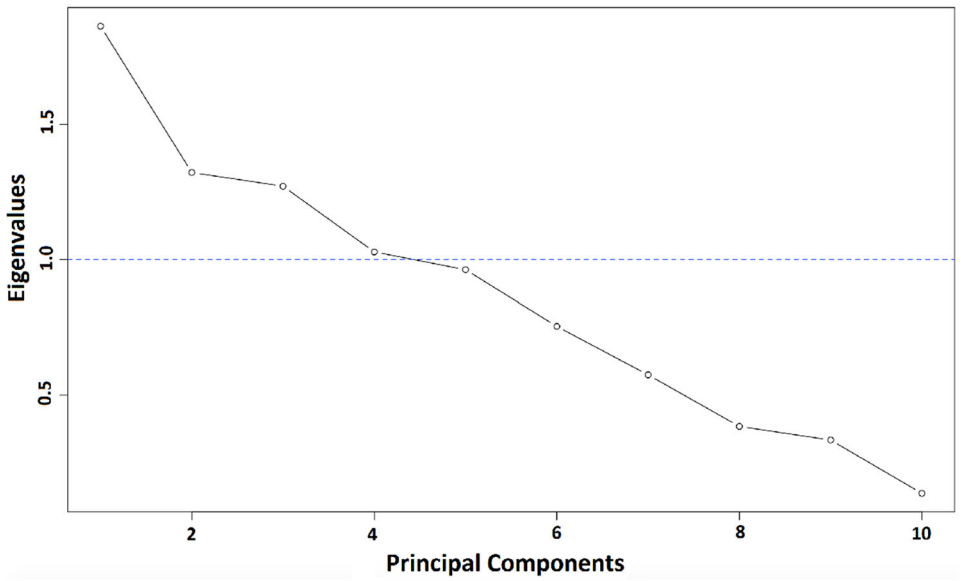

Fig. 7 PCA graph demonstrating Eigenvalues for the principal components

Table 5 Loading/rotation matrix of the selected most significant principal components

Fields	PC1	PC2	PC3	PC4	PC5
TCP Flags	0.46395	0.66868	0.10247	0.20834	0.43479
TCP Options	0.44029	0.58625	0.11341	0.22636	0.51578
ICMP Requests	0.42547	0.13763	0.77126	0.50812	0.21459
ICMP Packet Size	0.41512	0.12637	0.50132	0.64353	0.22927
UDP Requests	0.35459	0.33253	0.21872	0.26654	0.32253
TCP Window Size	0.12435	0.10501	0.01978	0.24361	0.29201
IP Time-To-Live	0.08543	−0.12875	−0.25983	0.13523	−0.30287
IPID Value	0.07653	−0.10182	0.10455	0.23156	−0.27128
IP Type Of Services	−0.09123	0.11764	0.04627	0.10138	0.28026
TCP Urgent Pointer	−0.27362	0.10763	−0.01774	−0.13982	0.11261

shown in Table 5, wherein, the five variables (TCP Flags, TCP Options, ICMP Requests, ICMP Packet Size and UDP Requests) have greater *Loadings* than the rest of the five variables (TCP Window Size, IP Time-To-Live, IPID Value, IP Type Of Services and TCP Urgent Pointer), which indicate that the first five variables have greater correlation with the five most significant principal components. Moreover, an in-depth analysis of *Loadings* of the first five variables for five components reveals some new attributes emerging from these five principal components.

The *Loadings* of the first principal component highlight the higher weighting and importance of the first five variables in the data. The PC2 and PC5 are mainly represented by the two variables TCP Flags and TCP Options due their greater *Loadings*, thus, the two variables can be grouped as a new TCP attribute (TCP Flags + TCP Options). Similarly, the PC3 and PC4 are mainly represented by the two variables ICMP Requests and ICMP Packet Size due their greater *Loadings*, thus, the two variables can be grouped as a new ICMP attribute (ICMP Requests + ICMP Packet Size). The fifth variable UDP requests has consistently greater *Loadings* in all the five principal components, which highlights its higher weighting and importance in the data, but as a separate networking protocol, thus, it can be considered as a separate UDP attribute to combine with TCP attribute and ICMP attribute for representing any principal components from PC1 to PC5. Eventually, these three derived attributes from PCA collectively represent the data and can be used for the subsequent operation.

Fuzzy inference system for predicting OS fingerprinting attacks and their severity levels on honeypots

In the previous analysis, the three new attributes (related to TCP, ICMP and UDP) are derived from the five most significant principal components as a sign of OS fingerprinting attack which can be used to predict the potential attack. However, it is not feasible to use the precise value of these attributes for developing a generic prediction approach due

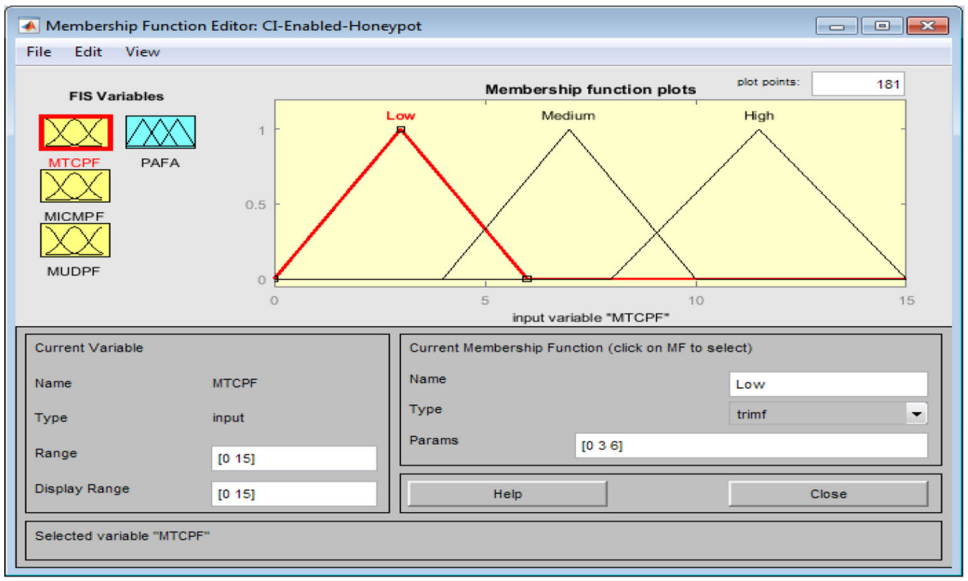

Fig. 8 Fuzzy input variable MTCPF and its fuzzy sets

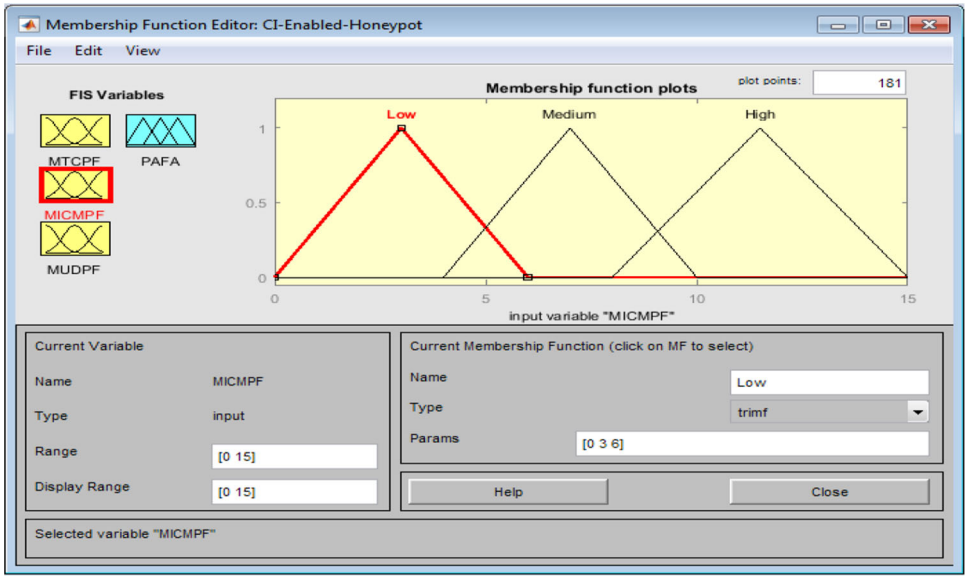

Fig. 9 Fuzzy input variable MICMPF and its fuzzy sets

to the coverage of several OS fingerprinting tools/techniques. Additionally, it is equally important to correlate these attributes in a way that the proposed prediction approach can predict most OS fingerprinting attacks accurately irrespective of tools/techniques. Fuzzy logic can address both problems effectively by offering a value range for each attribute to cover most OS fingerprinting tools/techniques in the prediction range and correlating attributes in a way that fuzzy rules can cover majority of the OS fingerprinting tools/techniques accurately.

Fuzzy input and output variables

In designing the fuzzy inference system, the three influential attributes are employed and their corresponding fuzzy input variables are derived. Here, TCP flags and TCP options are merged as a single attribute called Malicious TCP Field (MTCPF); ICMP requests and ICMP packet size are merged as a single attribute called Malicious ICMP Field (MICMPF); and the last variable UDP requests is kept unchanged with renaming as Malicious UDP Field (MUDPF). Therefore, the three derived fuzzy input variables are: MTCPF, MICMPF and MUDPF. This PCA-based evolution of only three attributes could offer a highly optimised and effective rule base for better prediction accuracy of the proposed system.

The value ranges for these three fuzzy input variables are determined based on the analysis of thousands of TCP/IP packets collected from Nmap and Xprobe2 experimental simulations and on the main principles of fingerprinting tools.

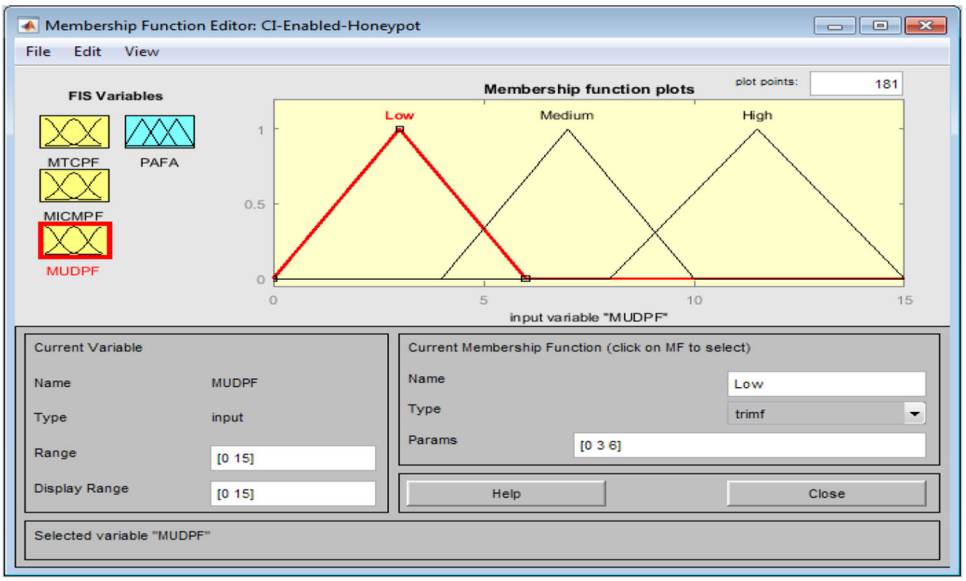

Fig. 10 Fuzzy input variable MUDPF and its fuzzy sets

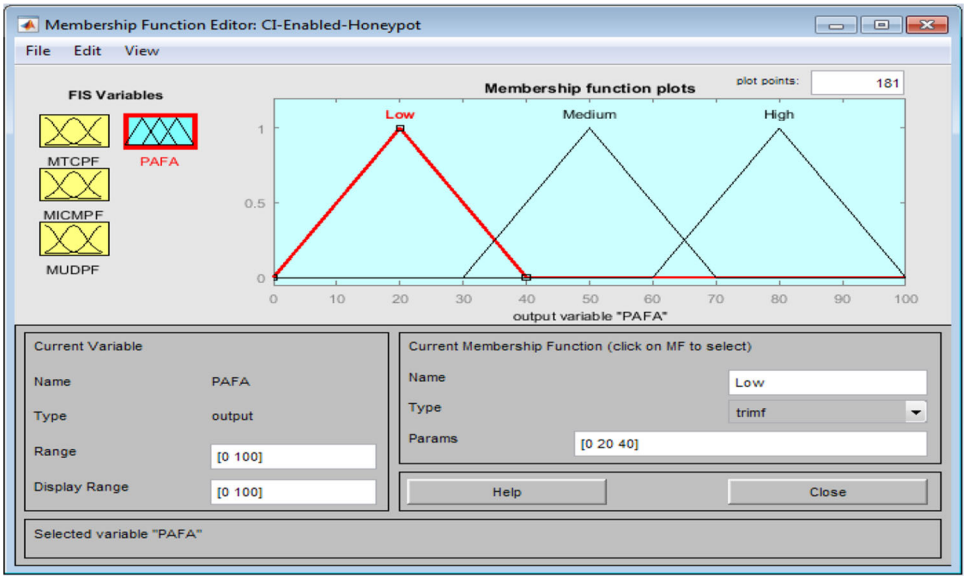

Fig. 11 Fuzzy output variable PAFA and its fuzzy sets

The common value range is set 1–15 packets for all three input variables based on the observation of various streams of TCP/IP packets. Subsequently, this value range is split into three fuzzy sets Low, Medium and High to represent three severity levels of an OS fingerprinting attack in the prediction. The corresponding value ranges set for Low is 0-6-packets, Medium is 4–10 packets, and High is 8–15 packets. Matlab is used to simplify this design. Figures 8, 9 and 10 illustrate three fuzzy input variables MTCPF, MICMPF and MUDPF in Matlab. As a preliminary design, a triangular membership function is selected, however, any other function can be selected and is very straightforward to adapt and analyse in Matlab.

Finally, the fuzzy output variable Probability of an Attempted Fingerprinting Attack (PAFA) is derived to represent the future correlation of three fuzzy input variables as a result. This variable is represented in percentage (0–100%) and split into three fuzzy sets Low, Medium and High to represent three severity levels of OS fingerprinting attack in the prediction. The corresponding value ranges set for Low is 0–40%, Medium is 30–70%, and High is 60–100%. Its Matlab design based on the similar triangular membership function is shown in Fig. 11 (Fig. 12).

Fuzzy rules and fuzzy rule base system

The fuzzy rules are created based on the correlation of three fuzzy input variables and their corresponding results in the form of a fuzzy output variable. The relation among these three input variables is established in a way that the fuzzy

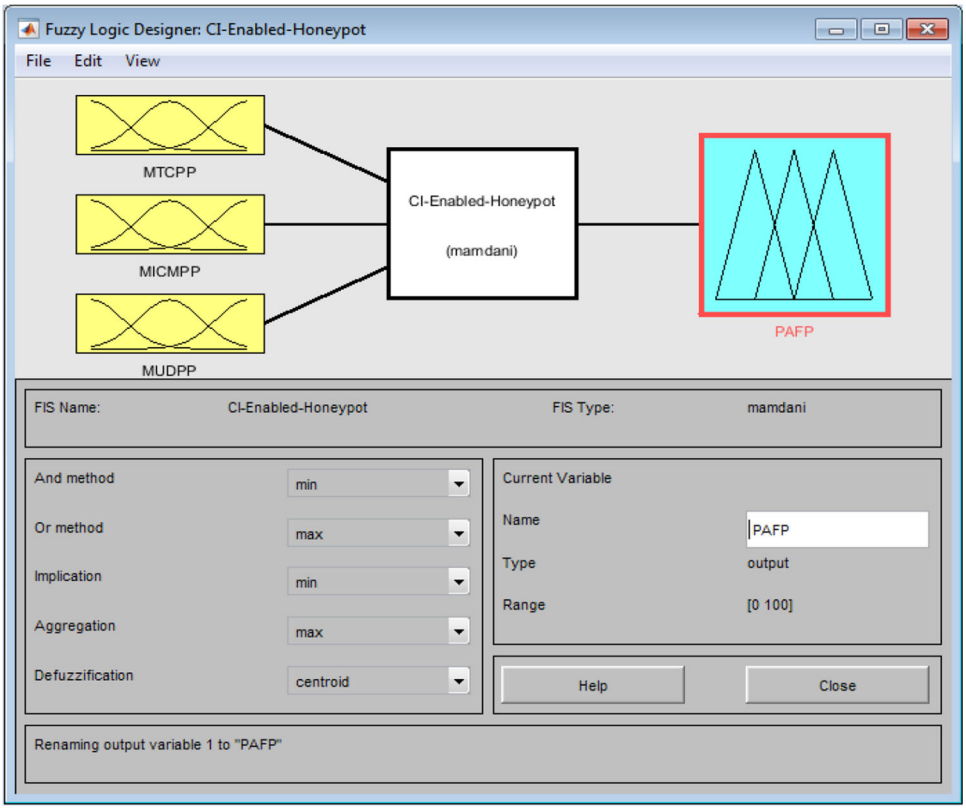

Fig. 12 Fuzzy inference system consisting of input and output variables for the proposed system

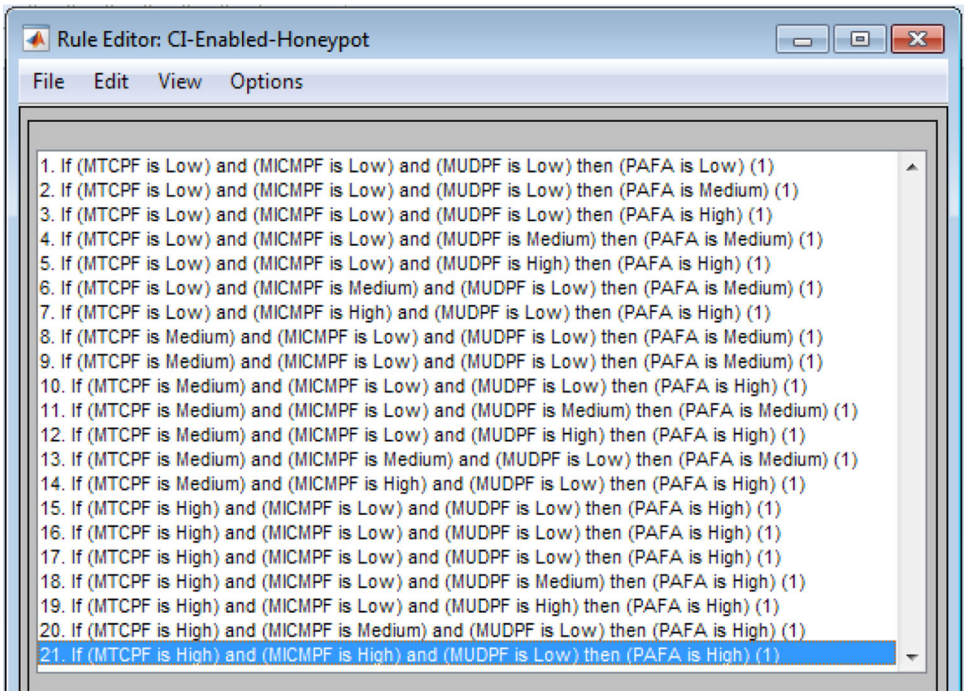

Fig. 13 Fuzzy rules of the proposed system to predict the fingerprinting attack on honeypots

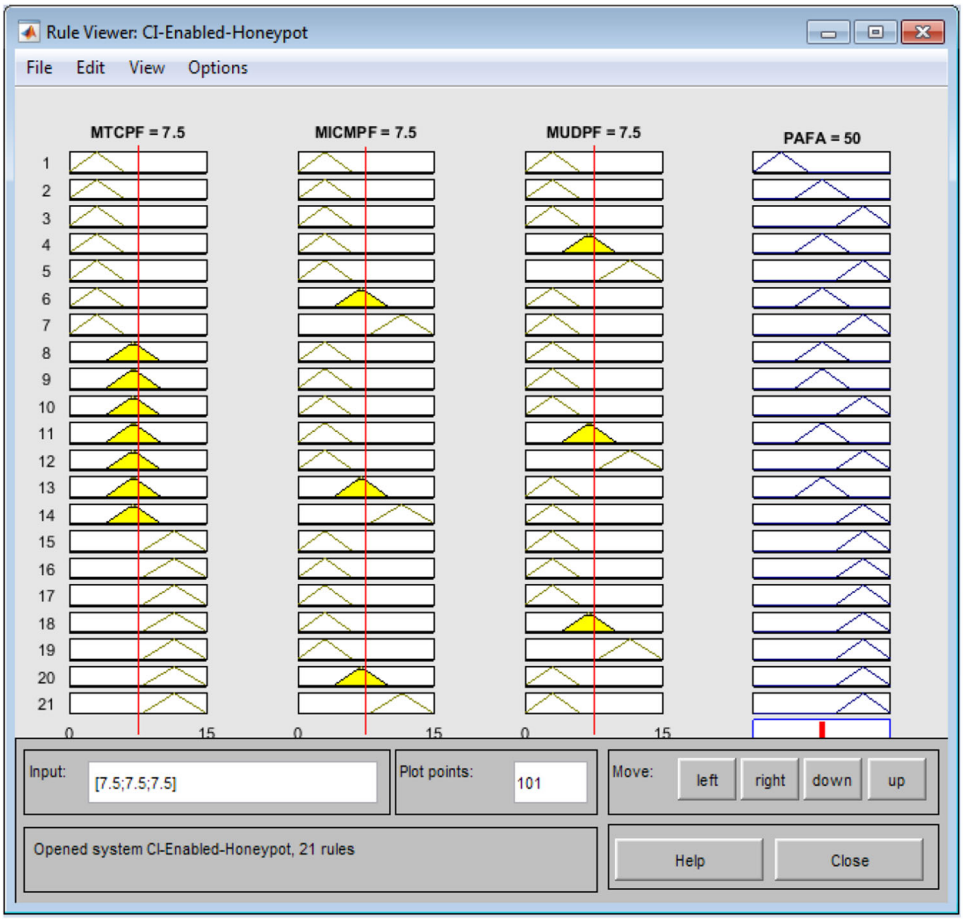

Fig. 14 Fuzzy rule base of the proposed system to predict the fingerprinting attack on honeypots

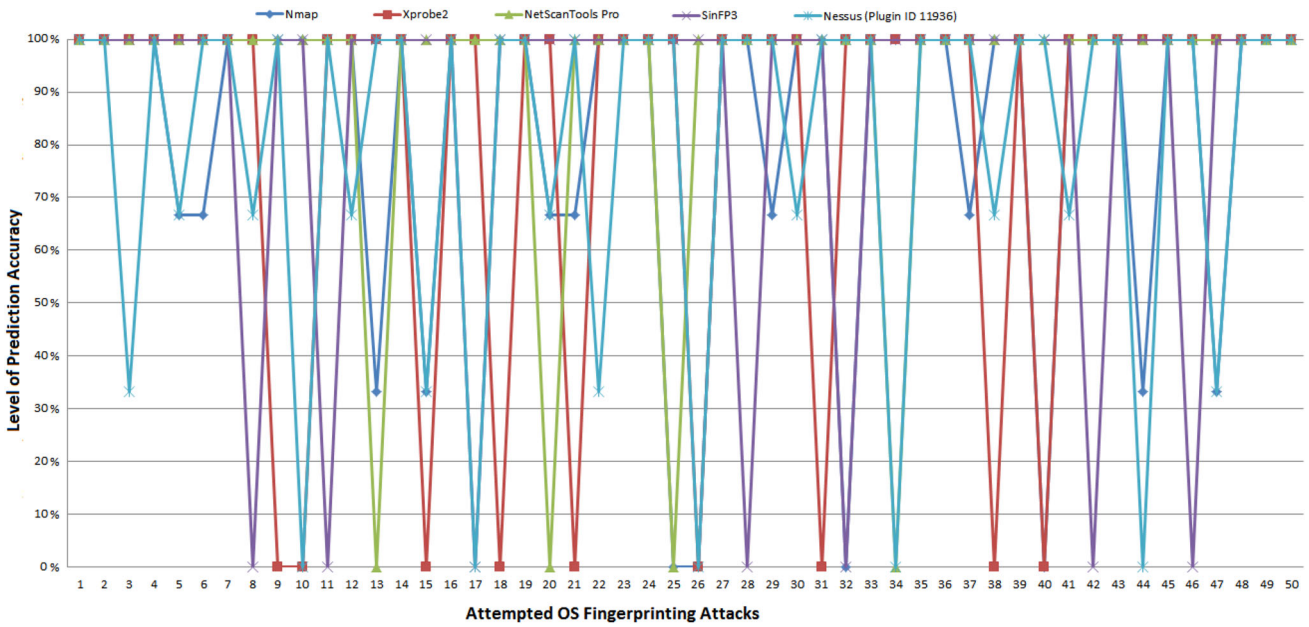

Fig. 15 Testing results of the CI-enabled honeypot based on the level of accuracy for each prediction for the five selected fingerprinting tools

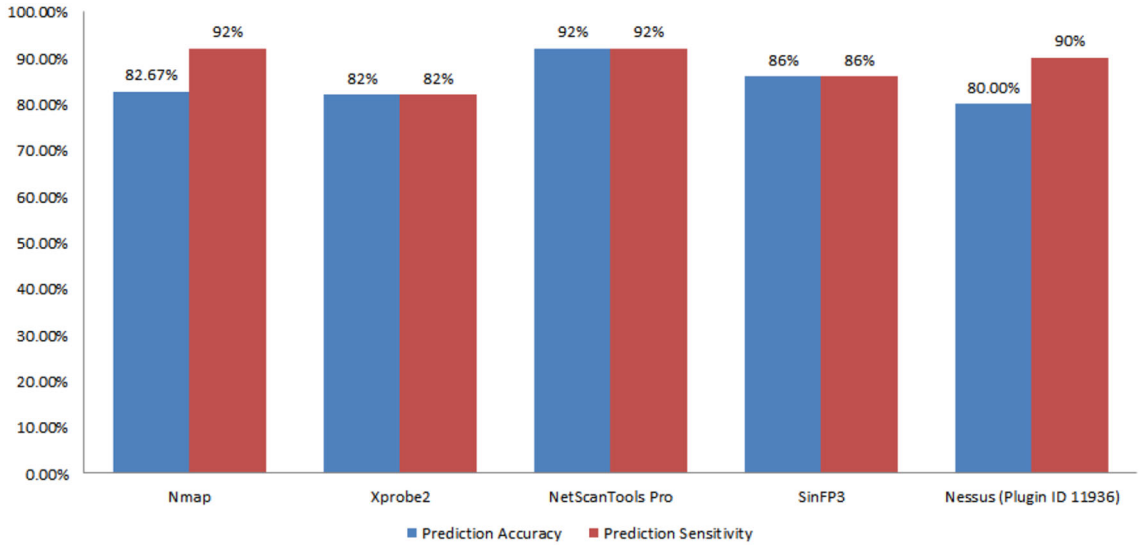

Fig. 16 Testing results of prediction accuracy and prediction sensitivity of the CI-enabled honeypot for the five selected fingerprinting tools

rule base system should be the generic rule base for several OS fingerprinting tools/techniques. The created sample rules are shown in Fig. 13 and the corresponding fuzzy rule base system is shown in Fig. 14. This designed fuzzy inference system (see Fig. 12) is based on Mamdani's inference method [23].

Test results of the computational intelligence enabled honeypot

The proposed CI-enabled honeypot system is employed to test the prediction results for an attempted OS fingerprinting attack from the five different OS fingerprinting tools: Nmap, Xprobe2, NetScanTools Pro, SinFP3 and Nessus. A total of 250 OS fingerprinting attacks using different attack scripts were carried out, i.e., 50 attacks from each tool. The testing results for all the tools and their level of prediction accuracy are shown in Fig. 15. In calculating the prediction accuracy of the proposed system, the prediction of attack levels as High, Medium and Low are translated into their corresponding percentage accuracy as 100%, 66.7% and 33.3% for the purpose of evaluation. The failure to detect an attempted attack is considered as 0%. The prediction accuracy of the proposed system was 82.67% for Nmap, 82% for Xprobe2, 92% for NetScanTools Pro, 86% for SinFP3 and 80% for Nessus, which is shown in Fig. 16. The overall prediction accuracy of the proposed system was 84.53%. Alongside prediction accuracy, the prediction sensitivity of the proposed system was also calculated based on the total True Positive (TP) and False Negative (FN) to determine whether the system can detect an attempted attack or not. Out of 50 attacks from each tool, the proposed system predicted attempted attacks 46 times for Nmap, 41 times for Xprobe2, 46 times for NetScanTools Pro, 43 times for SinFP3 and 45 times for Nessus. The prediction sensitivity of the proposed system was 92% for Nmap, 82% for Xprobe2, 92% for NetScanTools Pro, 86% for SinFP3 and 90% for Nessus, which is shown in Fig. 16. The overall prediction sensitivity of the proposed system was 88.4%. The prediction accuracy and sensitivity of the proposed system demonstrates its success for the five different types of fingerprinting tools (TCP-based, ICMP-based and combination of both).

Finally, Table 6 illustrates the summary of prediction results for the five OS fingerprinting tools, where the proposed system can predict the severity level of an attempted OS fingerprinting attack as *HIGH* for all the five tools with some exceptions as discussed in Table 6. These are related to Nmap and Nessus as they can perform a wide range of OS fingerprinting attacks, some of which rely on HTTP and other application layer protocols, however, this investigation concentrated on the core protocols of the network and transport layer (TCP, ICMP, UDP and IP). As a result of using HTTP and some other application layer protocols, there is a reduced reliance on the core TCP/IP protocols to obtain OS fingerprinting information leading to the generation of lower TCP/IP traffic and fewer abnormalities/patterns for the prediction of the system. However, HTTP and some application layer protocols can be included, with each protocol targeting a very specific attack, significantly increasing the complexity and overheads of the method. Whereas core TCP/IP protocols are included in all tools/attacks based on the TCP/IP stack fingerprinting technique, offering a lightweight generic approach that can predict all TCP/IP based fingerprinting attacks.

Table 6 Fuzzy inference system (FIS) based prediction for attempted OS fingerprinting attacks and their severity levels for various tools

OS Fingerprinting Tool	Main Protocol for Reconnaissance	Prediction of a Severity Level of Attempted OS Fingerprinting Attacks by FIS	Exception in the Prediction of Attempted OS Fingerprinting Attacks by FIS
Nmap	TCP-Based	*HIGH* Severity Level*	The protocols affected are where HTTP-based attack scripts (e.g. *nmap -sV*) are used, where it may not be able to predict *HIGH* severity levels
Xprobe2	ICMP-Based	*HIGH Severity Level*	
NetScanTools Pro	ICMP-Based	*HIGH Severity Level*	
SinFP3	TCP-Based	*HIGH Severity Level*	
Nessus	Both TCP and ICMP-Based	*HIGH* Severity Level*	The protocols affected are where SMB, NTP, SNMP, SSH and HTTP-based attack scripts are used, where it may not be able to predict *HIGH* severity levels

Where * means that FIS predicts *HIGH* severity for the majority of OS Fingerprinting Attacks, with the *Exception* of some application layer protocol-based attacks (see the last column)

Conclusion

This paper presented a computational intelligence enabled honeypot for discovering and predicting an attempted fingerprinting attack by using a Principal Components Analysis and Fuzzy Inference System. The proposed CI-enabled design was focused on the most common OS fingerprinting attack. The simulation of fingerprinting attacks and data (TCP/IP packets) collection was accomplished by employing - *KFSensor* honeypot tool and *Nmap* and *Xprobe2* fingerprinting tools. Subsequently, based on preliminary observations and empirical evidence, some of the important fields of collected TCP/IP packets were analysed to establish abnormalities or patterns as a sign of an attempted fingerprinting attack. Successively, it applied a PCA to determine the most influential fields, which were further utilised by the proposed FIS to predict the fingerprinting attack and its severity levels. Finally, the proposed system was successfully tested against the five popular fingerprinting tools. This included two previous tools Nmap and Xprobe2 and three new tools NetScanTools Pro, SinFP3 and Nessus; which were not involved in the development of CI-enabled honeypot. Notwithstanding, the CI-enabled honeypot being promising and encompassing several types of TCP/IP based fingerprinting attacks, it may omit some fingerprinting attacks which exploit some of the application layer protocols HTTP, SMB, NTP, SNMP and SSH. Thus, in the future, it is essential to enhance this CI-enabled honeypot and incorporate these fingerprinting attacks. Additionally, the developed fuzzy inference system can be improved to cover new attacks by utilising an adaptive rule base through dynamic fuzzy rule interpolation approach [24].

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Alata E, Nicomette V, Kaâniche M, Dacier M, Herrb M (2006) Lessons learned from the deployment of a high-interaction honeypot. In: 2006 Sixth European Dependable computing conference, pp 39–46. IEEE
2. Alex: Backtrack 5: Information gathering: network analysis: OS fingerprinting: Xprobe2 (2012, March 31). https://www.question-defense.com/2012/03/31/backtrack-5-/information-/gathering-network-analysis-os-/fingerprinting-xprobe2. Accessed Dec 2017
3. Allen JM (2008) OS and application fingerprinting techniques. https://www.sans.org/reading-room/whitepapers/authentication/os-application-/fingerprinting-techniques-32923. Accessed Dec 2017
4. Almquist P (1992) Type of service in the internet protocol suite. https://tools.ietf.org/html/rfc1349. Accessed Dec 2017
5. Arkin O, Yarochkin F (2003) A fuzzy approach to remote active operating system fingerprinting. http://www.sys-security.com/archive/papers/Xprobe2.pdf. Accessed Dec 2017

6. Arkin O, Yarochkin F (2018) Xprobe2(1)–Linux man page. https://linux.die.net/man/1/xprobe2. Accessed Dec 2017
7. Cabral W, Valli C, Sikos L, Wakeling S (2019) Review and analysis of cowrie artefacts and their potential to be used deceptively. In: 2019 International Conference on computational science and computational intelligence (CSCI), pp 166–171. IEEE
8. Cantrell C, Willebeek-LeMair M, Cox D, McHale J, Smith B, Kolbly D (2008) Active network defense system and method. US Patent 7,454,499
9. Costello AB, Osborne JW (2005) Best practices in exploratory factor analysis: Four recommendations for getting the most from your analysis. Pract Asses Res Eval 10(7):1–9
10. Greenwald LG, Thomas TJ (2007) Toward undetected operating system fingerprinting. WOOT 7:1–10
11. Grimes RA (2017) Honeypots, in hacking the hacker: learn from the experts who take down hackers. Wiley, Indianapolis. https://doi.org/10.1002/9781119396260.ch19
12. Horn JL (1965) A rationale and test for the number of factors in factor analysis. Psychometrika 30(2):179–185
13. Kaiser HF (1960) The application of electronic computers to factor analysis. Educ Psychol Measur 20(1):141–151
14. Keyfocus.net (2018) KFSensor: advanced windows honeypot system–enhanced intrusion and insider threat detection for your network (2018). http://www.keyfocus.net/kfsensor/. Accessed Dec 2017
15. Lyon GF (2009) Chapter 15. Nmap Reference Guide. https://nmap.org/book/man-os-detection.html. Accessed Dec 2017
16. Lyon GF (2009) Chapter 15. Service and version detection. https://nmap.org/book/man-version-detection.html. Accessed Dec 2017
17. Lyon GF (2009) Chapter 7. Service and application version detection. https://nmap.org/book/vscan.html. Accessed Dec 2017
18. Lyon GF (2009) Chapter 8. Remote OS detection: TCP/IP fingerprinting methods supported by Nmap. https://nmap.org/book/osdetect-methods.html. Accessed Dec 2017
19. Lyon GF (2009) Chapter 9. Nmap scripting engine. https://nmap.org/book/nse-usage.html. Accessed Dec 2017
20. Lyon GF (2009) Nmap network scanning: the official Nmap project guide to network discovery and security scanning. Insecure, Sunnyvale CA, United States, pp 468. https://doi.org/10.5555/1538595. Accessed Dec 2017
21. Lyon GF (2011) Nmap service DB. https://svn.nmap.org/nmap/nmap-services. Accessed Dec 2017
22. Lyon GF (2017) Nmap OS fingerprinting 2nd generation DB. https://svn.nmap.org/nmap/nmap-os-db. Accessed Dec 2017
23. Mamdani EH, Assilina S (1975) An experiment in linguistic synthesis with a fuzzy logic controller. Int J Man-Mach Stud 7(1):1–13
24. Naik N, Diao R, Shen Q (2018) Dynamic fuzzy rule interpolation and its application to intrusion detection. IEEE Trans Fuzzy Syst 26(4):1878–1892
25. Naik N, Jenkins P (2016) Enhancing windows firewall security using fuzzy reasoning. In: IEEE International Conference on dependable, autonomic and secure computing, pp 263–269
26. Naik N, Jenkins P (2016) Fuzzy reasoning based windows firewall for preventing denial of service attack. In: IEEE International Conference on fuzzy systems, pp 759–766
27. Naik N, Jenkins P (2016) Web protocols and challenges of web latency in the web of things. In: 2016 Eighth International Conference on ubiquitous and future networks (ICUFN), pp 845–850. IEEE
28. Naik N, Jenkins P (2018) Discovering hackers by stealth: predicting fingerprinting attacks on honeypot systems. In: 2018 IEEE International Symposium on systems engineering (ISSE)
29. Naik N, Jenkins P (2018) A fuzzy approach for detecting and defending against spoofing attacks on low interaction honeypots. In: 21st International Conference on information fusion, pp 904–910. IEEE
30. Naik N, Jenkins P, Cooke R, Ball D, Foster A, Jin Y (2017) Augmented windows fuzzy firewall for preventing denial of service attack. In: 2017 IEEE International Conference on fuzzy systems (FUZZ-IEEE), pp 1–6
31. Naik N, Jenkins P, Cooke R, Yang L (2018) Honeypots that bite back: a fuzzy technique for identifying and inhibiting fingerprinting attacks on low interaction honeypots. In: 2018 IEEE International Conference on fuzzy systems (FUZZ-IEEE)
32. Naik N, Jenkins P, Davies P, Newell D (2016) Native web communication protocols and their effects on the performance of web services and systems. In: 16th IEEE International Conference on computer and information technology (CIT), pp 219–225. IEEE
33. Naik N, Jenkins P, Savage N (2018) Threat-aware honeypot for discovering and predicting fingerprinting attacks using principal components analysis. In: IEEE Symposium Series on computational intelligence (SSCI)
34. Naik N, Jenkins P, Savage N, Katos V (2016) Big data security analysis approach using computational intelligence techniques in R for desktop users. In: IEEE Symposium Series on computational intelligence (SSCI)
35. Naik N, Shang C, Jenkins P, Shen Q (2020) Building a cognizant honeypot for detecting active fingerprinting attacks using dynamic fuzzy rule interpolation. Expert Syst. https://doi.org/10.1111/exsy.12557
36. Naik N, Shang C, Shen Q, Jenkins P (2018) Vigilant dynamic honeypot assisted by dynamic fuzzy rule interpolation. In: IEEE Symposium Series on computational intelligence (SSCI)
37. Rowe NC (2006) Measuring the effectiveness of honeypot counter-counter deception. In: System sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on, vol. 6, p 129c. IEEE
38. Spitzner L (2003) Honeypots: tracking hackers, vol 1. Addison-Wesley Reading, Boston
39. Touch J (2014) Updated specification of the IPv4 ID Field. https://tools.ietf.org/html/rfc6864. Accessed Dec 2017
40. Yarochkin FV, Arkin O, Kydyraliev M, Dai SY, Huang Y, Kuo SY (2009) Xprobe2++: low volume remote network information gathering tool. In: Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on, pp. 205–210. IEEE
41. Zwick WR, Velicer WF (1986) Comparison of five rules for determining the number of components to retain. Psychol Bull 99(3):432