D-FRI-Honeypot: A Secure Sting Operation for Hacking the Hackers Using Dynamic Fuzzy Rule Interpolation

Nitin Naik, Changjing Shang, Paul Jenkins, and Qiang Shen

***Abstract*—As active network defence systems, honeypots are commonly used as a decoy to inspect attackers and their attack tactics in order to improve the cybersecurity infrastructure of an organisation. A honeypot may be successful provided that it disguises its identity. However, cyberattackers continuously endeavour to discover honeypots for evading any deception and bolstering their attacks. Active fingerprinting attack is one such technique that may be used to discover honeypots by sending specially designed traffic. Preventing a fingerprinting attack is possible but doing that may hinder the process of dealing with the attackers, counteracting the purpose of a honeypot. Instead, detecting an attempted fingerprinting attack in real-time can enhance a honeypot's capability, uninterruptedly managing any immediate consequences and preventing the honeypot being identified. Nevertheless, it is difficult to detect and predict an attempted fingerprinting attack due to the challenge of isolating it from other similar attacks, particularly when imprecise observations are involved in the monitoring of the traffic. Dynamic fuzzy rule interpolation (D-FRI) enables an adaptive approach for effective reasoning with such situations by exploiting the best of both inference and interpolation. The dynamic rules produced by D-FRI facilitate approximate reasoning with perpetual changes that often occur in this type of application, where dynamic rules are required to cover new network conditions. This paper proposes a D-FRI-Honeypot, an enhanced honeypot running D-FRI framework in conjunction with Principal Component Analysis, to detect and predict an attempted fingerprinting attack on honeypots. This D-FRI-Honeypot works with a sparse rule base but is able to detect active fingerprinting attacks when it does not find any matching rules. Also, it learns from current network conditions and offers a dynamically enriched rule base to support more precise detection. This D-FRI-Honeypot is tested against five popular fingerprinting tools (namely, Nmap, Xprobe2, NetScanTools Pro, SinFP3 and Nessus), to demonstrate its successful applications.**

***Index Terms*—Dynamic fuzzy rule interpolation, D-FRI, Honeypot, D-FRI-Honeypot, fingerprinting attack, sparse rule base, principal components analysis.**

I. Introduction

CYBERATTACKERS are becoming more sophisticated and devious in their attacks involving artificial intelligence (AI) [1]–[3]. This requires more and better countermeasures from cyber experts exploiting the same technological developments in AI [2], in addition to conventional passive or active defence systems as a part of defensive strategies [4]. Honeypots, as a form of active defence systems, are commonly used as a decoy to inspect attackers and their sophisticated attacking tactics, in an effort to improve the cybersecurity infrastructure of an organisation [5], [6].

A. Research Motivations

A honeypot may be successful as long as it disguises its identity. However, cyberattackers continuously endeavour to discover honeypots for evading any deception and bolstering their attacks. Active fingerprinting attack is one such technique that can be used to discover honeypots by sending specially designed traffic. If it is identified by an attacker, it can be abused as a bot to attack others and even its own network [5], [7], [8]. Preventing a fingerprinting attack is possible but such an action may hinder the process of its dealing with the attackers, thereby counter-acting the purpose of a honeypot [9]. Instead, detecting an attempted fingerprinting attack in real-time can enhance the capability of a honeypot. This is because it helps ensure uninterrupted dealing of the honeypot while managing any immediate consequences, including prevention of its own function from other malicious honeypots. Unfortunately, little work exists to detect and predict an attempted fingerprinting attack on honeypots because it is very difficult to isolate such an attack from many of the similar ones, especially when imprecise observations regarding the network traffic are involved. Therefore, this research is set to investigate various fingerprinting attacks. It proposes an intelligent and adaptive technique for detecting and predicting an attempted fingerprinting attack on honeypot, in an effort to enhance the capability of a honeypot.

In the context of present application studies, both the observations of and the knowledge available for detecting fingerprinting attacks are often described in imprecise terms. This means that a system built for such applications must be able to handle ill-defined variable values and vague statements. Fuzzy rule-based reasoning can help in this regard, being one of the most popular techniques for designing rule-based intelligent systems that meet such a requirement. The efficacy of a fuzzy rule-based system is largely dependent on its underlying rule base. Typically, such a system infers the conclusion based on any matching rules in

Manuscript received June 17, 2020; revised September 6, 2020; accepted September 7, 2020. *(Corresponding author: Nitin Naik.)*

Nitin Naik is with the School of Informatics and Digital Engineering, Aston University, Birmingham B4 7ET, U.K. (e-mail: n.naik1@aston.ac.uk).

Changjing Shang and Qiang Shen are with the Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3FL, U.K. (e-mail: cns@aber.ac.uk; qqs@aber.ac.uk).

Paul Jenkins is with the Cardiff School of Technologies, Cardiff Metropolitan University, Cardiff CF5 2YB, U.K. (e-mail: pjenkins2@cardiffmet.ac.uk).

Digital Object Identifier 10.1109/TETCI.2020.3023447

the given rule base. Yet, this type of system can only be used effectively when the rule base is dense (i.e., it covers all, or at least almost all input conditions). Practically speaking, it may not be possible to attain such a dense rule base, especially when coping with novel networks. Even if it is attainable to devise a dense rule base, the use of such a rule base may cause adverse effects on computational overheads and redundancy. Fuzzy rule interpolation (FRI) [10]–[12] has been proposed to perform approximate reasoning with just a sparse rule base. However, in both cases of dense and sparse rule bases, largely the rule base is static with no mechanism to update it, which can make it ineffectual in long run if it is not updated over time. This forms a significant challenge for the potential successful applications of fuzzy systems to cybersecurity problems.

To tackle the aforementioned specific challenge, dynamic fuzzy rule interpolation (D-FRI) has been developed, enabling the induction and exploitation of the most updated and dynamic rule base during the FRI process [13]. Indeed, D-FRI offers an integrated framework for inference and interpolation and can be used with any fuzzy intelligent system irrespective of the type of its underlying rule base. It benefits the cybersecurity applications through its ability of assisting in the monitoring of perpetual changing traffic conditions within a given network. This is evident from its initial applications in the area, such as D-FRI-Snort [14], D-FRI-WinFirewall [15], D-FRI-CiscoFirewall [16], ID-Honeypot [17] and VD-Honeypot [18]. Building on these original attempts, D-FRI is herein utilised again to develop a D-FRI-Honeypot for detecting or predicting active fingerprinting attacks on honeypots.

B. Main Contributions

The work proposed herein helps enhance the capability of a honeypot to conceal its own identity, thereby enabling itself to successfully perform its function. In particular, it makes the following major contributions to the relevant literature:

- A methodology to detect and prevent fingerprinting attacks to (cyber security) honeypots that can both run and also learn online. The methodology offers an automated means for dealing with the challenging problem of only having a sparse, and imprecisely described, rule base for the detection of fingerprinting attacks.
- Active fingerprinting attacks are simulated on a given honeypot to collect attack data (i.e., TCP/IP packets). The simulation is accomplished by employing the *KFSensor* honeypot and *Nmap* and *Xprobe2* fingerprinting tools, with the simulated attack data captured in two different logs, respectively by the use of KFSensor directly and with Wireshark analyser for forensic analysis.
- A number of the important fields of collected TCP/IP packets are empirically examined to ascertain abnormalities or patterns as an indication of an attempted fingerprinting attack, with the classical principal component analysis (PCA) exploited to determine the most influential fields, which are further utilised to develop an effective approach to predicting fingerprinting attacks.
- D-FRI is applied to correctly correlate the identified influential fields and to predict fingerprinting attacks and their severity level. The resulting system, dubbed D-FRI-Honeypot hereafter, is successfully tested against five popular fingerprinting tools, including both Nmap and Xprobe2 (that have been used in the development of D-FRI-Honeypot) and three other tools: NetScanTools Pro, SinFP3 and Nessus, neither of these new tools has been involved in building the D-FRI-Honeypot previously.

C. Content Organisation

The rest of this paper is organised into the subsequent sections: Section II explains the basic concepts regarding D-FRI, honeypots and fingerprinting attacks, including operating system fingerprinting. Section III describes the simulation and analysis of fingerprinting attacks on honeypots for the collection of empirical data. Section IV discusses a comprehensive examination of the chosen TCP/IP fields and their related abnormalities/patterns as signs of a fingerprinting attack. Section V presents the design and development of a D-FRI-Honeypot for discovering and predicting fingerprinting attacks on honeypots. Section VI shows the testing results of the implemented D-FRI-Honeypot system. Section VII discusses the main limitations of D-FRI-Honeypot, and Section VIII concludes the paper and points out possible improvements of this work.

II. Background

This section introduces the basic concepts upon which to develop the work to be presented in the subsequent sections.

A. Dynamic Fuzzy Rule Interpolation (D-FRI)

The conventional fuzzy reasoning systems infer any outcomes using a dense rule base that covers the problem domain. When it is not possible for the rules available to cover the complete domain then a situation involving the use of a sparse rule base results. The most effective way to draw outcomes under such scenarios is the utilisation of an FRI system. These two procedures, of inference and interpolation, can be combined for designing more effective fuzzy systems using the sparse rule base. The integrated system can offer several benefits such as performing conventional inference with the sparse rule base and hence, minimising computational overheads (as rule interpolation requires more computation). Nonetheless, the effectiveness of such an integrated system may be affected due to the static nature of the sparse rule base as it demands an additional mechanism for intelligent learning and adaptation of the rule base as the problem domain evolves.

The providential, dynamic fuzzy rule interpolation (D-FRI) is particularly developed to address this issue, which offers the most updated rule base during a problem-solving process, through the utilisation of FRI [13]. It can also be used to gradually develop a dense rule base from an original sparse rule base if needed. The D-FRI system is the result of an integration of fuzzy inference, fuzzy rule interpolation, and dynamic rule learning and adaptation techniques. As such, it

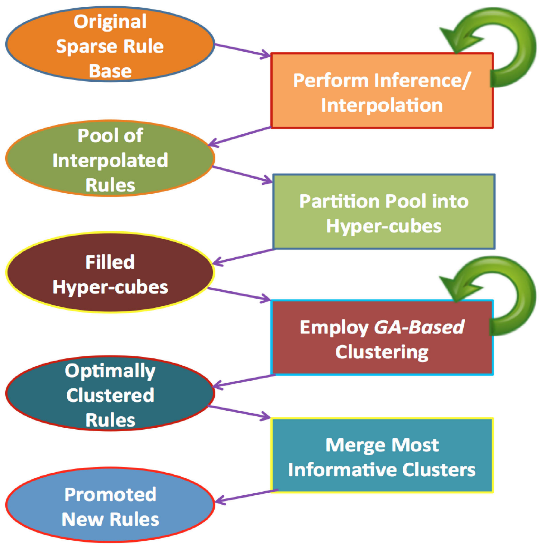

Original Sparse Rule Base
Perform Inference/ Interpolation
Pool of Interpolated Rules
Partition Pool into Hyper-cubes
Filled Hyper-cubes
Employ GA-Based Clustering
Optimally Clustered Rules
Merge Most Informative Clusters
Promoted New Rules

Fig. 1. Dynamic fuzzy rule interpolation (D-FRI) [13].

offers a generic framework to encompass a wide range of fuzzy inference and interpolation mechanisms. However, in this work, the Mamdani's fuzzy inference [19] and transformation-based rule interpolation (T-FRI) [20] techniques are utilised due to their popularity and availability. The design and working stages of D-FRI are illustrated in Fig. 1 and its further details can be found in [13].

B. Honeypots

A honeypot is a concealed security system that functions as a decoy to entice cyberattackers to reveal their information [21]. It deceives, detects and diverts cyberattackers, whereas concurrently gathering their information [7]. Most honeypots are used to imitate the functionalities of a real network so that cyberattackers may interpret such a system as the real network to carry out attacks, thereby revealing information regarding the attacker [8]. They are installed to entail constantly observations to uncover the vulnerabilities and new attacks to the underlying real network and to develop an enhanced defensive strategy [22]. However, honeypots should not be used as a defensive system to protect the network. Honeypots are categorized into two categories on the basis of their design and level of activity with cyberattackers [23]: low-interaction honeypots and high-interaction honeypots. Low-interaction honeypots normally imitate real systems and have restricted communication with cyberattackers; whereas, high-interaction honeypots are normally real systems and have unrestricted communication with cyberattackers [18]. Low-interaction honeypots are most commonly used honeypots due to their benefits of limited resources, overheads and interaction with attackers, and therefore, this research work has also employed it.

C. Fingerprinting Attack

In a fingerprinting attack, an attacker typically sends a sequence of fabricated packets to a target system or network [24]. It does so to provoke a response in the form of packets containing fingerprint information with the intention of obtaining the target's identification. Fingerprinting attacks are categorised 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, analysing their response packets to extract fingerprinting information [21]. In a passive fingerprinting attack, cyberattackers do not send any packets to the target, instead they sniff, capture and analyse traffic from the target to extract fingerprinting information [21]. Active fingerprinting attacks are more accurate than passive fingerprinting attacks as the result is based on the direct response from the target. Therefore, in this design of D-FRI-Honeypot, only an active fingerprinting attack is considered.

D. OS Fingerprinting Attack

Operating system (OS) fingerprinting is the most prevalent fingerprinting type of attack, which is performed on a target system or network to obtain specific information regarding its OS, services, device type and type of architecture [25]. 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 [26]. 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. Cyberattackers are highly successful in performing OS fingerprinting attacks as the same TCP/IP protocol suite is implemented by every OS differently, resulting in different responses for the same TCP/IP query. Consequently, different responses generated by different operating systems divulge substantial information about a given system to cyberattackers. The complete process of an OS fingerprinting attack is dependent on TCP/IP protocol suite. As such, it is sometimes referred to as TCP/IP stack fingerprinting. Most low-interaction honeypots simulate several OSs in order to presents an illusion of real OSs, which makes this OS fingerprinting attack more crucial to discover any low-interaction honeypot easily. Moreover, having obtained precise information about the OS of the target, cyberattackers can launch more complex attacks with grater severity against the target system or network.

III. Simulation of OS Fingerprinting Attacks

Every OS implements the TCP/IP protocol suite differently. This, acquisition of a fingerprint of any OS requires the analysis of the TCP/IP packets sent by that OS. The process of finding a fingerprint of a particular OS is therefore, primarily based on the examination of the TCP, ICMP and UDP protocols as in general, every fingerprinting tool or technique sends and receives these three protocol-based packets differently to a target system. 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)

TABLE I
NMAP AND XPROBE2 OS FINGERPRINTING ATTACK SCRIPTS

(a) Nmap OS Fingerprinting Attack Scripts

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

(b) Xprobe2 OS Fingerprinting Attack Scripts

No.	Xprobe2 OS Fingerprinting Attack Script
1	$xprobe2\ < Honeypot\ IP >$
2	$xprobe2\ -D/-M\ < Module\ Name >< Honeypot\ IP >$
3	$xprobe2\ -B\ < Honeypot\ IP >$
4	$xprobe2\ -T/-U\ < Port\ Range >\ < Honeypot\ IP >$
5	$xprobe2\ -p\ < Protocol : Port : Status >\ < Honeypot\ IP >$

of tools/techniques. Reflecting this practical observation, the present simulation covers both TCP-based and ICMP-based OS fingerprinting attacks, in an effort to acquire the type of OS fingerprint (and also, information about the associated services) of a honeypot that may reveal its identity.

A. TCP-Based OS Fingerprinting Attacks Using Nmap

Nmap is the most powerful and reliable scanning tool which is very effective in delivering an OS fingerprinting attack, being mainly TCP-based. Many Nmap scripts use heuristics and fuzzy signature matching to derive conclusions about a target host OS or services [27]. During an OS fingerprinting attack, Nmap sends a stream of TCP/IP packets, to identified open and closed ports on the target. This stream of TCP/IP packets contains TCP, UDP, and ICMP packets. These packets/probes are aimed at several existing ambiguities and their exploitation in terms of standard protocol Request-for-Comments (RFCs). When the target sends a reply back to Nmap regarding these packets, the fingerprinting tool analyses the values of various parameters of the TCP, ICMP and UDP packets and constructs an OS fingerprint to match against the database of OS signatures it contains [28]. Depending on the OS signature matching result, it predicts the possible OS of the target. If there is no exact match then an integrated fuzzy representation and inference mechanism is used to perform such a prediction [29].

Table I(a) shows the five different Nmap scripts that may be used for an OS fingerprinting attack. The first Nmap script is the basic OS fingerprinting command that reveals the OS fingerprints and other details such as OS version numbers, device type and architectural information. The second script offers more descriptive fingerprinting information such as OS type, device type, host script and traceroute. The third utilises fuzzy techniques to predict the closest matched OS (in percentage), in the event that it does not find any exact match. The fourth is used to perform OS fingerprinting continuously for the given number of attempts to improve the accuracy of prediction. The fifth, and the final Nmap script is completely different from the other four, utilising a different signature database for 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 a different intensity (ranging from 0 to 9 with 9 being the highest intensity) to improve the accuracy of prediction [30].

Particularly, the first four Nmap scripts employ the Nmap database called *nmap-os-db* [31], and the fifth Nmap script utilises the Nmap database called *nmap-services* [32]. For accuracy and to discount any outlier data, each Nmap script (with various sub-options) is executed 100 times to record the results under various network conditions while observing retransmitted packet patterns.

B. ICMP-Based OS Fingerprinting Attacks Using Xprobe2

Nmap is a powerful and reliable fingerprinting tool, however, its results are largely dependent upon the TCP packets sent. An ICMP-based fingerprinting simulation and analysis becomes an essential alternative in order to propose a generic solution. Xprobe2 is one of the first ICMP-based fingerprinting tools, which is herein employed for such simulation. It utilises ICMP packets and is based on the notion of signature engine supported by fuzzy signature matching [33].

During an OS fingerprinting attack, Xprobe2 sends a stream of TCP/IP packets, to identified open and closed ports on the target [34]. This stream of TCP/IP packets contains ICMP, TCP, and UDP packets. In particular, Xprobe2 consists of 13 modules (whilst versions such as Xprobe2++ and Xprobe2-ng consist of an additional 3 modules (fingerprint:icmp_info, app:ftp, and app:http), when used to find an OS fingerprint [34]. This tool is both more effective and quicker than Nmap due to the utilisation of a fewer number of TCP/IP packets. Unfortunately, it is obsolete and not updated. Thus, it is unable to ascertain newer OSs including Windows 7 on the honeypot system. Nonetheless, the present work is focused on the development of a counter strategy for identifying and predicting an OS fingerprinting attack. Having recognised that Xprobe2 is the very first ICMP-based OS fingerprinting tool, forming the basis for all subsequent ICMP-based OS fingerprinting tools, it is imperative to investigate Xprobe2-based simulation.

Table I(b) 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 [33]. 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 switch $-B$ sends consecutive TCP handshake requests to any open TCP

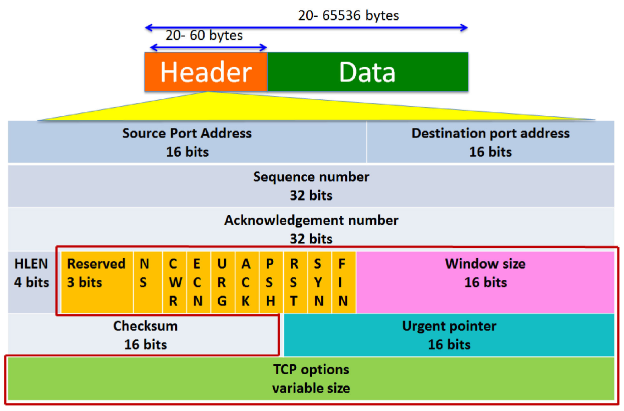


Fig. 2. Investigated fields of TCP Header for attempted OS fingerprinting attack.

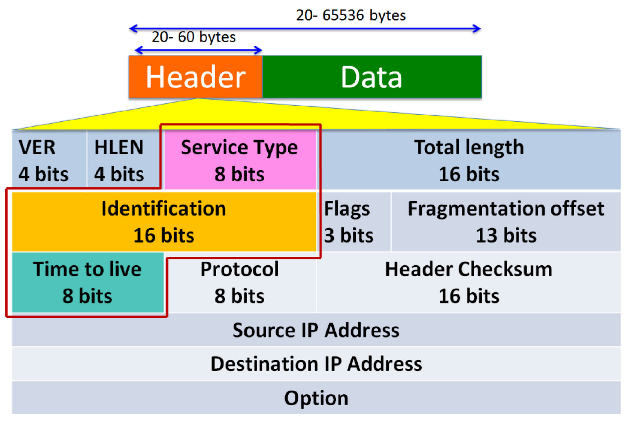


Fig. 3. Investigated fields of IP Header for attempted OS fingerprinting attack,

port such as 80, 443, 23, 21, 25, 22, 139, 445 and 6000 on an intended system and expects a SYN ACK reply [35]. 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) [35]. The fifth Xprobe2 script determines a fingerprint of an OS by utilising additional details regarding a protocol, port and the current status via switch $-p$. The protocol can be chosen from TCP or UDP, the port number from 1 to 65535, 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 [35].

Note that similar to the simulation with Nmap, to obtain accurate results while removing any outliers in the data, each Xprobe2 script (with various sub-options) is executed 100 times. This helps record the results under different network conditions and observe patterns of retransmitted packets.

IV. Identification of OS Fingerprinting Attacks

The 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 an attacker is analysed to reveal any observed abnormalities/patterns in the various fields of TCP/IP protocols (i.e., TCP, ICMP, UDP and IP). This analysis identifies ten indicator fields of TCP/IP protocols with respect to the detected discrepancies in the attack simulation data (see Figs. 2 and 3). Additionally, these ten TCP/IP fields are analysed to emphasise their weight based on the literature and the core attack principles of popular OS fingerprinting tools/techniques.

A. Abnormalities/Patterns in TCP Flags

TCP comprises six standard flags (SYN, ACK, URG, PSH, RST, FIN), controlling the nature and flow of the transmission. There are several flags or combinations of flags which are considered as abnormal or illegal ones based on the RFCs of TCP. Yet, there is no explanation regarding the handling of

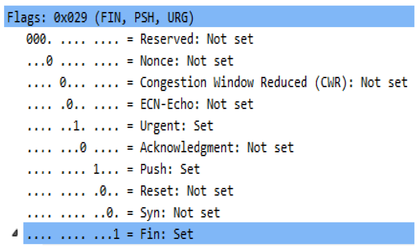

```
Flags: 0x029 (FIN, PSH, URG)
  000. .... .... = Reserved: Not set
  ...0 .... .... = Nonce: Not set
  .... 0... .... = Congestion Window Reduced (CWR): Not set
  .... .0.. .... = ECN-Echo: Not set
  .... ..1. .... = Urgent: Set
  .... ...0 .... = Acknowledgment: Not set
  .... .... 1... = Push: Set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

Fig. 4. Captured TCP packet with URG/PSH/FIN probing during OS fingerprinting attack.

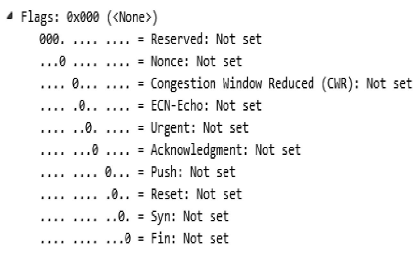
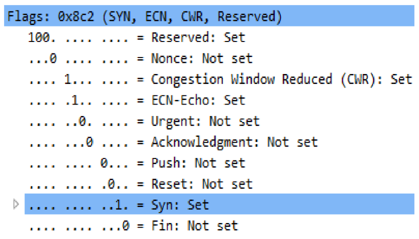

```
Flags: 0x000 (<None>)
  000. .... .... = Reserved: Not set
  ...0 .... .... = Nonce: Not set
  .... 0... .... = Congestion Window Reduced (CWR): Not set
  .... .0.. .... = ECN-Echo: Not set
  .... ..0. .... = Urgent: Not set
  .... ...0 .... = Acknowledgment: Not set
  .... .... 0... = Push: Not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

Fig. 5. Captured TCP packet with NULL probing during OS fingerprinting attack.

such abnormal/illegal flags. Different OS's generate different responses for an abnormal/illegal flag or combination of such flags. This is a significant concern for the security community as attackers generally exploit these responses to determine the OS of the target. Thus, identification of a number of these abnormal/illegal TCP flags can be utilised as a good indicator of an OS fingerprinting attack, which is relatively straightforward to find as they are well known. Certain OS fingerprinting tools also utilise additional control flags (e.g., CWR, ECN) and Reserved Bits in their attack techniques. Examples of these abnormal/illegal TCP flags are:

- *URG/PSH/FIN Probing* – See Fig. 4
- *NULL Packet* – See Fig. 5
- *Reserved Bit Probing* – See Fig. 6
- *ECN-Echo Probing* – See Fig. 6
- *FIN Probing*
- *SYN/FIN Probing*



```
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. 6. Captured TCP packet with Reserved Bit and ECN-Echo probing during OS fingerprinting attack.

B. Abnormalities/Patterns in TCP Options

Most fingerprinting tools exploit the TCP Options field of a TCP header because it is an adaptable field and can be of any size ranging from 0 to 40 bytes. A TCP options field may contain a subset of or all the following attributes: Maximum Segment Size (MSS), Window Scaling, Selective Acknowledgements (SACK), Timestamps, and Nop. Therefore, every OS customises the 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. Of course, this can be combined with other indicators as a sign of an OS fingerprinting attack.

C. Abnormal Use 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 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.

D. Abnormalities 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. 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.

E. Abnormal Use of IP Type of Service (TOS)

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. The TOS field is commonly redefined as the Differentiated Services Code Point (DSCP). 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.

F. Abnormalities/Commonalities in IP Identification (IPID)

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) [36]. 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.

G. 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.

H. 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/ICMP packets to collect fingerprinting information from the target. An attacker sends UDP packets to a port of the target and it may or may not receive any response, depending on the state of a port

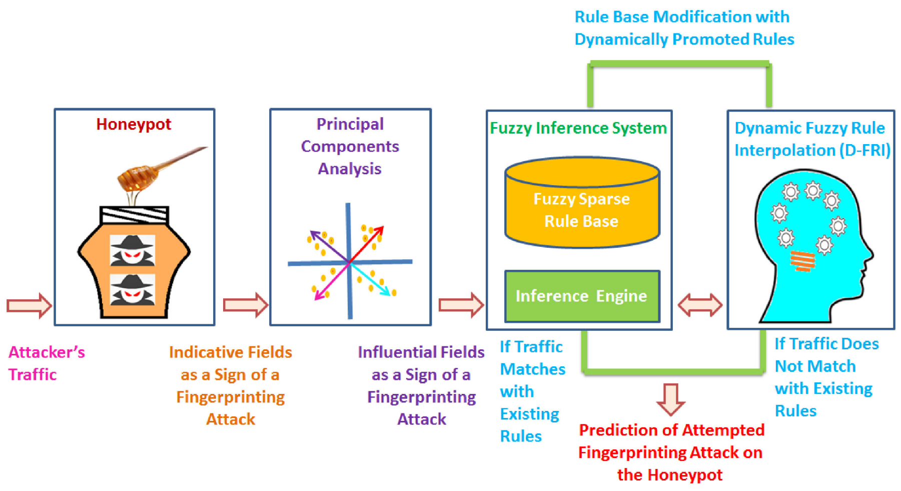

Rule Base Modification with Dynamically Promoted Rules
Honeypot
Principal Components Analysis
Fuzzy Inference System
Fuzzy Sparse Rule Base
Inference Engine
Dynamic Fuzzy Rule Interpolation (D-FRI)
Attacker's Traffic
Indicative Fields as a Sign of a Fingerprinting Attack
Influential Fields as a Sign of a Fingerprinting Attack
If Traffic Matches with Existing Rules
If Traffic Does Not Match with Existing Rules
Prediction of Attempted Fingerprinting Attack on the Honeypot

Fig. 7. D-FRI-Honeypot for predicting OS fingerprinting attacks on honeypots.

being open or closed. The target replies with an ICMP error message *Destination Unreachable (ICMP Type 3)* if the port is closed; otherwise, it receives no reply for an open or filtered port. Generally, an UDP packet used in OS fingerprinting is either empty or set to a fixed payload. An attacker can also set an IP DF flag in the UDP packet that can prompt the target 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. Also, this can be combined with other indicators as a sign of such an attack.

I. Abnormalities/Patterns in ICMP Requests

ICMP is an error announcing protocol that is used for troubleshooting, and for providing control and error message services. It may be particularly employed by network devices (e.g., routers, gateways, hosts) to announce error messages when there is an issue in delivering packets. As a result, an attacker can exploit legitimate ICMP request packets to collect significant information about an OS of the target. Such packets include: 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).

Most OS fingerprinting tools utilise abnormal ICMP requests by changing certain parameters of the aforementioned ICMP requests. For example, an abnormal ICMP Echo request (Type 8) can be easily determined by examining its *Code* value. This is because such a value should always be *Code* 0, but certain OS fingerprinting tools use an 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. They can also be combined with other indicators as a sign of an OS fingerprinting attack.

J. Abnormalities/Patterns in ICMP Packet Size

ICMP packets are normally used to report errors in a standard format and therefore, their sizes are relatively stable with respect to the OS, within a predictable range. 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 detail. For example, in Nmap-based experimental simulation, the baseline size was 74 bytes (as with the most common ICMP packet size in Windows), and the sizes of two collected ICMP packets by KFSensor Honeypot are 149 to 179. The recorded sizes of the ICMP packets for all the Nmap experimental iterations are the same. This is a clear indication of abnormality/pattern found in the ICMP request packets received from a likely OS fingerprinting attacker. This can be combined with other indicators as a sign of an OS fingerprinting attack, of course.

V. D-FRI-Honeypot for Predicting Fingerprinting Attacks on Honeypots

The development of D-FRI-Honeypot is herein based on D-FRI, supported by Principal Components Analysis (PCA). In particular, PCA is used to determine the most influential TCP/IP fields from the earlier analysed TCP/IP fields, which are subsequently employed by D-FRI to predict or detect an OS fingerprinting attack. The working procedure of this D-FRI-Honeypot is shown in Fig. 7.

A. Determining the Most Influential TCP/IP Fields

Principal Components Analysis is one of the most effective computational techniques for data dimensionality reduction (DR) while retaining most of the information contained with the original data. In this work, the primary reasons for the preferred choice of PCA over alternative DR techniques are:

- that reduced number of TCP/IP fields means decreased requirements for capacity and memory, enabling a lightweight system;
- that it is a very efficient technique for data involving not very high dimensionality, which is the case here;

TABLE II
PRINCIPAL COMPONENTS ANALYSIS AND LOADING/ROTATION MATRIX

(a) 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

(b) Loading/Rotation Matrix of 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

- that it is of low noise sensitivity, facilitating the handling of volatile network traffic which typically incurs in the problem concerned; and
- that it uses simple and readily accessible statistical calculations, avoiding the need of complex programming tasks.

As outlined previously, based on empirical analysis, there are ten most crucial fields that may be exploited as indicators for signs of fingerprinting attacks. To aid in improving the efficacy of prediction of a 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 be accomplished using PCA, where principal components with higher variances reveal the most significant attributes, showing additional information about the underlying 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 or Loading (contribution of each original field to the principal component). In this experimental investigation, traditionally accepted thresholds are considered to determine the number of principal components to use, including: $Cumulative\, Proportion\, of\, Variance$ >85%, $Eigenvalue$ >1 (from Kaiser's rule [37]) and $Loading^2 > 1/Total\ Number\, of\, Variables$ (which indicates that $Loading$ for any selected component should be relatively higher than others) [38]–[40].

Table II(a) illustrates the standard deviation, variances and cumulative proportion of variances for the ten principal components. The cumulative proportion of variance for the first five components is 0.8821793 (≈88%), which is higher than the pre-decided threshold value of 85% (commonly adopted in the literature [40]). From this, it is known that the first five components are the most significant for the collected fingerprinting data. The cumulative value of the remaining five components is approximately 12%, indicating that their contribution to the data is rather low. Further evaluation of the selected first five components is useful, checking whether their eigenvalues >1 (see Fig. 8). It is true for the first four components, however,

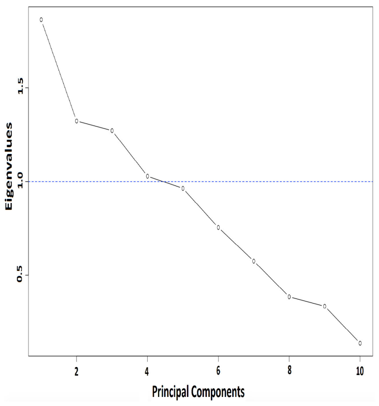


Fig. 8. PCA Graph demonstrating eigenvalues for principal components.

the fifth component is slightly smaller than 1 (≈1), but it is essential, in conjunction with the other four, to constitute the 85% cumulative proportion of the variance. Thus, unlike those five components whose joint contribution is less than 15%, this fifth one is retained as one of the most influential.

The selection of influential variables are additionally further evaluated by examining the $Loading$ values, showing 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 shown in Table II(b), wherein, the five original fields of TCP Flags, TCP Options, ICMP Requests, ICMP Packet Size and UDP Requests have greater $Loadings$ than the other five (TCP Window Size, IP Time-To-Live, IPID Value, IP Type Of Services and TCP Urgent Pointer). This indicates that the first five fields have greater correlation with the five most significant principal components.

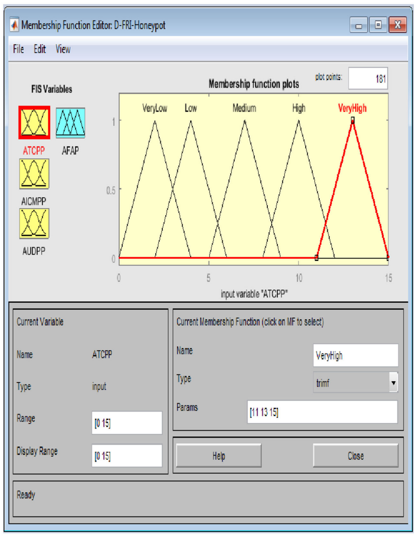
An in-depth analysis of the $Loadings$ of the first five fields in relation to the top five principal components leads to interesting observations and inductions. In particular, the $Loadings$ of the first principal component with respect to the first five original fields highlight its higher weighting and hence, greater importance in the data. PC2 and PC5 are mainly represented by the two fields of TCP Flags and TCP Options due to their greater *Loadings*. This means that the two original fields can be grouped as a new TCP attribute (TCP Flags + TCP Options). Similarly, PC3 and PC4 are mainly represented by the two fields of ICMP Requests and ICMP Packet Size and thus, these two original variables can be grouped as a new ICMP attribute (ICMP Requests + ICMP Packet Size). The fifth field, UDP Requests, has consistently greater *Loadings* in relation to all the five principal components. This highlights its higher weighting and importance in the data, but as a separate networking protocol. From this observation, it may be considered as a separate UDP attribute to be combined with TCP and ICMP attributes in representing any principal components from PC1 to PC5. Together, these three derived (i.e., combined) attributes from PCA collectively represent the original data, thereby significantly increasing computational efficiency.

B. Predicting OS Fingerprinting Attacks With Severity Levels

In the previous analysis, the three new attributes (related to TCP, ICMP and UDP) are derived from the five most significant principal components which are useful to act as a sign of OS fingerprinting attacks. Additionally, it is very useful to determine the value range of these attributes for developing a generic attack prediction mechanism, due to their interrelationships with several techniques underlying the data representation mechanisms within commonly adopted OS fingerprinting tools. Yet, it is equally important to be able to exploit these attributes in a way that a proposed method can efficiently predict most OS fingerprinting attacks accurately irrespective of the underlying techniques/tools. Fuzzy set-based representation can help address both issues effectively, by offering a value range for each attribute to deal with the problem of imprecise attribute representation in most OS fingerprinting techniques, accurately [6].

1) Fuzzy Input and Output Variables: In designing the fuzzy inference system for predicting OS fingerprinting attacks, those three influential attributes as identified previously are employed. In particular, TCP flags and TCP options are merged as a single attribute named Abnormal TCP Packet (ATCPP); ICMP requests and ICMP packet size are merged as another single attribute named Abnormal ICMP Packet (AICMPP); and the variable UDP requests is kept unchanged but renamed as Abnormal UDP Packet (AUDPP) to better match the eyes (see Figs. 9 to 11). The value ranges for these fuzzy variables are all set to 1-15 packets empirically, based on the analysis of thousands of TCP/IP packets collected from Nmap and Xprobe2 simulations and the underlying principles of fingerprinting tools. From this, five fuzzy sets are defined on this common range: Very Low, Low, Medium, High and Very High, respectively representing five severity levels of an OS fingerprinting attack in the prediction. In terms of the support of each of these fuzzy values, the overall range is split such that Very Low is of the support of 0-4 packets, Low is of 2-6 packets, Medium is of 5-9 packets, High is of 8-12 packets, and Very High is of 11-15 packets.

The fuzzy output variable from the system is named Attempted Fingerprinting Attack Possibility (AFAP), which signifies the predicted possibility of whether there may exist an OS fingerprinting attack, based on computed correlation of the

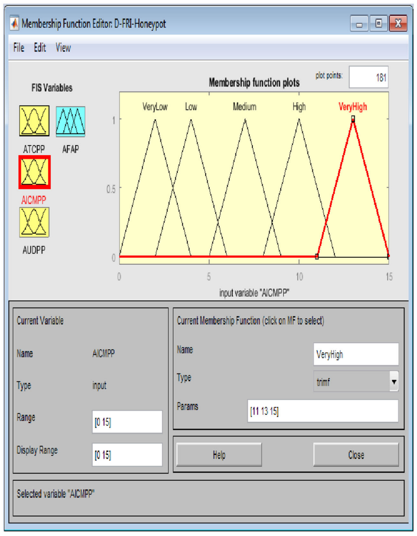


Current Variable
Name ATCPP
Type input
Range [0 15]
Display Range [0 15]

Current Membership Function (click on MF to select)
Name VeryHigh
Type trimf
Params [11 13 15]

Ready

Fig. 9. Fuzzy input variable ATCPP.

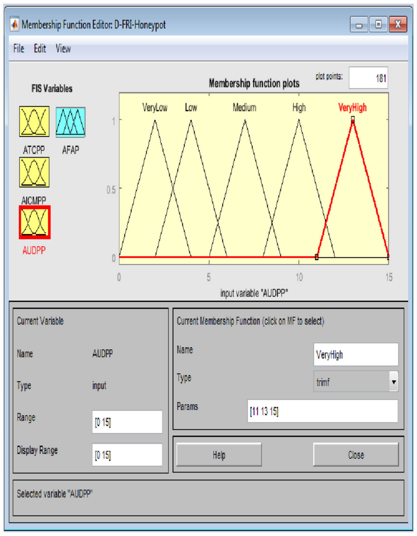


Current Variable
Name AICMPP
Type input
Range [0 15]
Display Range [0 15]

Current Membership Function (click on MF to select)
Name VeryHigh
Type trimf
Params [11 13 15]

Selected variable "AICMPP"

Fig. 10. Fuzzy input variable AICMPP.

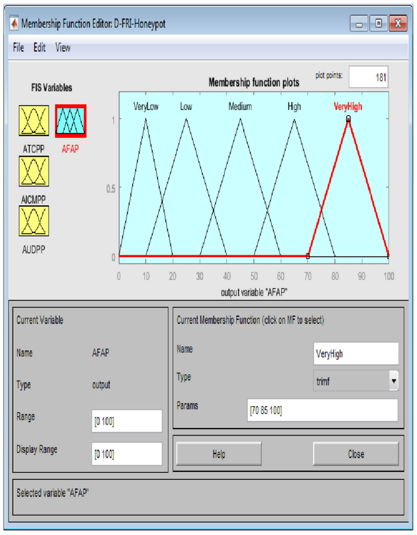


Current Variable
Name AUDPP
Type input
Range [0 15]
Display Range [0 15]

Current Membership Function (click on MF to select)
Name VeryHigh
Type trimf
Params [11 13 15]

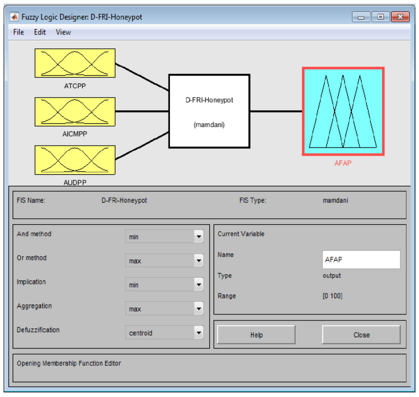
Selected variable "AUDPP"

Fig. 11. Fuzzy input variable AUDPP.



Current Variable
Name AFAP
Type output
Range [0 100]
Display Range [0 100]
Current Membership Function (click on MF to select)
Name VeryHigh
Type trimf
Params [70 85 100]
Help
Close
Selected variable "AFAP"

Fig. 12. Fuzzy output variable AFAP.



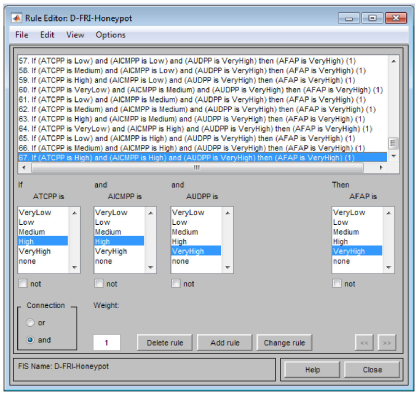
FIS Name: D-FRI-Honeypot
FIS Type: mamdani
And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid
Current Variable
Name AFAP
Type output
Range [0 100]
Help
Close
Opening Membership Function Editor

Fig. 13. Fuzzy prediction system.

above three fuzzy input variables. Its range is represented within the range of 0-100%, split also into similar five fuzzy terms: Very Low, Low, Medium, High and Very High (see Fig. 12). Particularly, the supports of these fuzzy values are empirically defined as follows: Very Low is of the support 0-20%, Low is of 10-40%, Medium is of 30-60%, High is of 50-80% and Very High is of 70-100%.

2) *Fuzzy Rule Base and Fuzzy Inference:* The rules are created on the basis of computed correlations between the three fuzzy input variables and the output variable. The constraint over this rule generation process is imposed such that the resulting fuzzy rule base should consist of generic rules applicable to several commonly adopted OS fingerprinting techniques or tools. Samples of the generated rules are shown in Fig. 14 and the fuzzy rule base is presented in Fig. 15. Note that due to the nature of the application problem, the rule base is rather sparse. Any prediction performed through a direct use of these rules is implemented by the FIS (see Fig. 13), which is based on the popular Mamdani's inference method [19].

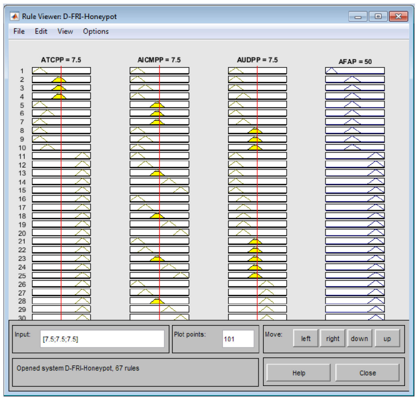
3) *D-FRI Sub-system:* Low-interaction honeypots are typically restricted in their resources and capabilities. This natural fact is in addition to the limitation of running sparse rule-based



```
57. If (ATCPP is Low) and (AICMPP is Low) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
58. If (ATCPP is Medium) and (AICMPP is Low) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
59. If (ATCPP is High) and (AICMPP is Low) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
60. If (ATCPP is VeryLow) and (AICMPP is Medium) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
61. If (ATCPP is Low) and (AICMPP is Medium) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
62. If (ATCPP is Medium) and (AICMPP is Medium) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
63. If (ATCPP is High) and (AICMPP is Medium) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
64. If (ATCPP is VeryLow) and (AICMPP is High) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
65. If (ATCPP is Low) and (AICMPP is High) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
66. If (ATCPP is Medium) and (AICMPP is High) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
67. If (ATCPP is High) and (AICMPP is High) and (AUDPP is VeryHigh) then (AFAP is VeryHigh) (1)
```

If ATCPP is: VeryLow, Low, Medium, [✓ High], VeryHigh, none; not
and AICMPP is: VeryLow, Low, Medium, [✓ High], VeryHigh, none; not
and AUDPP is: VeryLow, Low, Medium, High, [✓ VeryHigh], none; not
Then AFAP is: VeryLow, Low, Medium, High, [✓ VeryHigh], none; not
Connection: or [✓ and]
Weight: 1
Delete rule
Add rule
Change rule
FIS Name: D-FRI-Honeypot
Help
Close

Fig. 14. Sample fuzzy rules.



Input: [7.5;7.5;7.5]
Plot points: 101
Opened system D-FRI-Honeypot, 67 rules
Help
Close

Fig. 15. Fuzzy rule base.

FIS and therefore, may adversely affect the effectiveness of the detection. Furthermore, in practice, the system may not collect all traffic whilst its associated operation is subject to perpetual changes in network conditions. As Mamdani's inference is only effective when the input conditions are at least met partially with any existing rule (otherwise, no detection can be made without being able to fire any rule), D-FRI sub-system is employed here. It helps to significantly reduce the occurrence of situations where no detection is made when there is no match available given the sparse rule base.

Initially, it performs a certain fuzzy rule interpolation when no match is available from the given rules, generating an outcome. The sub-system stores every such interpolated result (interchangeably termed interpolated rule when used together with the otherwise unmatched values for the input variables), to enable dynamic learning subsequently. This learning process is done after reaching a pre-defined threshold of the interpolated rules. Dynamic learning is only applied once a while, on such a group of interpolated rules, in order to obtain and promote the most concurrent rules to the sparse rule base.

Importantly, the dynamic rule promotion process enhances the overall system's efficacy in two ways: 1) increasing the possibility of inference (thus reducing interpolation overheads) in future, and 2) increasing the accuracy of the outcomes by generating and making use of more accurate and concurrent rules. This is evident in the experimental results to be presented

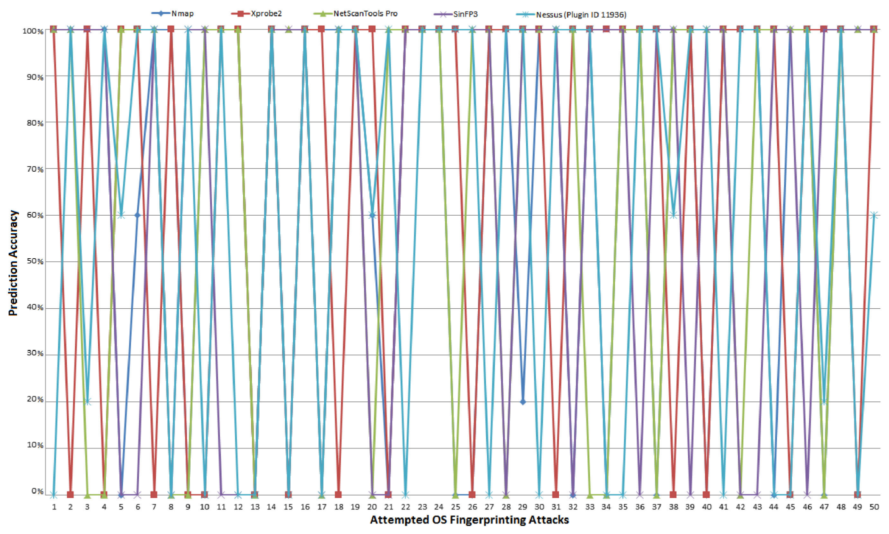


Fig. 16. Accuracy per each prediction with D-FRI-Honeypot running fuzzy inference only.

next. Overall, the D-FRI sub-system improves the detection rate of an active fingerprinting attack, by minimising false positives and false negatives thanks to the exploitation of any recent traffic and hence, update rules.

VI. Experimental Results

D-FRI-Honeypot is tested against five different OS fingerprinting tools Nmap, Xprobe2, NetScanTools Pro, SinFP3 and Nessus, with its prediction result for an attempted OS fingerprinting attack recorded. These tools are selected to cover a diverse range of the underlying fingerprinting approaches, including the TCP-based, ICMP-Based and combinational tools. This is important to ensure rigorous testing of the D-FRI-Honeypot against a variety of different fingerprinting attacks. In this experimentation, each fingerprinting tool is utilised to simulate 50 different attacks. Thus, a total of 250 attacks are carried out on the honeypot. The performance of D-FRI-Honeypot is recorded under two different conditions: 1) D-FRI-Honeypot using just fuzzy inference and 2) D-FRI-Honeypot using D-FRI sub-system.

Throughout the experimental investigations, the following two performance indices are used for system evaluation: *prediction accuracy* and *detection sensitivity*. *Prediction accuracy* shows how accurately D-FRI-Honeypot has predicted each attack. Note that the fuzzy linguistic outcomes (namely, Very High, High, Medium, Low and Very Low) are translated into numerical values in terms of weightage in percentages, such as 100%, 80%, 60%, 40% and 20% for evaluation purpose, with a failure to detect an attempted attack considered as 0%. In addition to the prediction accuracy, it is also important to check how many attacks are not detected or alerted by D-FRI-Honeypot. Therefore, *detection sensitivity* is calculated as the ratio of the number of *True Positives* over the total of *True Positives* and *False Negatives*. The selection of theses two performance indices *prediction accuracy* and *detection sensitivity* offered a balanced approach to measuring the success (or otherwise) of

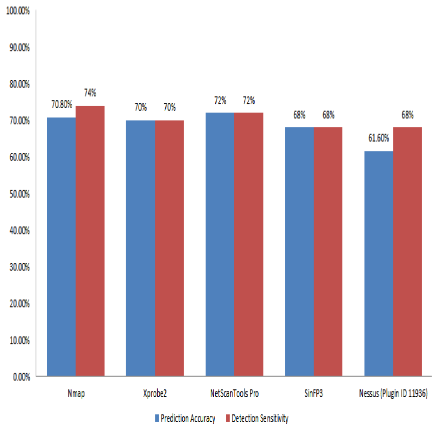


Fig. 17. Prediction Accuracy and Detection Sensitivity of D-FRI-Honeypot running fuzzy inference only.

D-FRI-Honeypot, in terms of how many and how accurate it may predict attempted fingerprinting attacks.

A. *D-FRI-Honeypot Performance Using Fuzzy Inference*

With D-FRI-Honeypot running fuzzy inference only, the prediction accuracy for all the five tools investigated is illustrated in Fig. 16. The overall *prediction accuracy* for each tool and the associated 50 attempted attacks is calculated as follows: 70.8% for Nmap, 70% for Xprobe2, 72% for NetScanTools Pro, 68% for SinFP3 and 61.6% for Nessus. These results are depicted in Fig. 17. The overall prediction accuracy of D-FRI-Honeypot running just fuzzy inference is therefore, 68.48%.

The *detection sensitivity* of D-FRI-Honeypot without the use of D-FRI for each tool is calculated over 50 attempted attack, resulting in: 74% for Nmap, 70% for Xprobe2, 72% for NetScanTools Pro, 68% for SinFP3 and 68% for Nessus. These are also shown in Fig. 17. The overall detection sensitivity of D-FRI-Honeypot using fuzzy inference is therefore, 70.4%. This is itself a decent performance considering the sparse rule

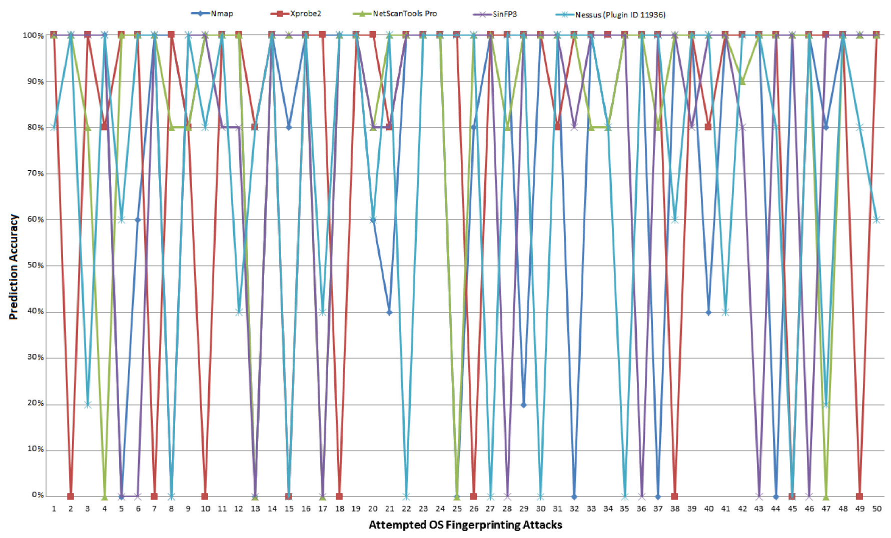


Fig. 18. Accuracy per each prediction of D-FRI-Honeypot with D-FRI sub-system.

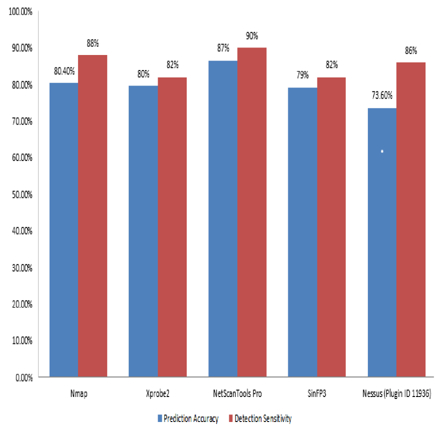


Fig. 19. Prediction Accuracy and Detection Sensitivity of D-FRI-Honeypot with D-FRI sub-system.

base given since the rule base covers approximately 56% of the problem space (which equivalently speaking, would require 44% more rules to form a dense rule base). It is nevertheless crucial to improve this performance in order to detect more attempted attacks.

B. *D-FRI-Honeypot Performance Using D-FRI Sub-system*

Use of D-FRI-Honeypot running with the D-FRI sub-system helps in improving the system's prediction performance. This is reflected in the evaluation with both metrics *prediction accuracy* and *detection sensitivity* (for the same attacks considered).

The prediction accuracy in response to each attack for all the five fingerprinting tools is illustrated in Fig. 18. The prediction accuracy per tool regarding the 50 attempted attacks is calculated as: 80.4% for Nmap, 79.6% for Xprobe2, 86.6% for NetScanTools Pro, 79.2% for SinFP3 and 73.6% for Nessus. These are summarised in Fig. 19. The prediction accuracy of D-FRI-Honeypot running the D-FRI sub-system is therefore,

TABLE III
SUMMARY OF PREDICTION ACCURACY AND DETECTION SENSITIVITY OF D-FRI-HONEYPOT

D-FRI-Honeypot	Prediction Accuracy	Detection Sensitivity
Without D-FRI	68.48%	70.4%
With D-FRI	**79.88%**	**85.6%**

79.88%. This demonstrates a considerable improvement over the system without the use of D-FRI.

The detection sensitivity of D-FRI-Honeypot with D-FRI sub-system for each tool is also calculated over the 50 attempted attacks, resulting in 88% for Nmap, 82% for Xprobe2, 90% for NetScanTools Pro, 82% for SinFP3 and 86% for Nessus. These are shown in Fig. 19. The overall detection sensitivity of D-FRI-Honeypot using D-FRI is 85.6%, a significant improvement over the case where no D-FRI is applied. More detailed comparisons are given below.

C. *Comparing D-FRI-Honeypot With or Without D-FRI*

The above two sets of experimentations, running D-FRI-Honeypot with just Mamdani's fuzzy inference and with D-FRI sub-system respectively, have to a certain extent shown the strengthened performance of the latter, given the same sparse fuzzy rule base. As a matter of fact, the employment of D-FRI has improved the *prediction accuracy* of D-FRI-Honeypot against attempted attacks by a rate of approximately 10% for each attack tool and a rate of 11.4% overall (see Fig. 20). Similarly, the *detection sensitivity* of D-FRI-Honeypot against the attempted attack for each fingerprinting tool by approximately 12% and by 15.2% overall (see Fig. 21).

To reinforce the performance gains through the use of D-FRI, both *prediction accuracy* and *detection sensitivity* of D-FRI-Honeypot with or without D-FRI are summarised in Table III. Clearly, D-FRI sub-system helps D-FRI-Honeypot performing

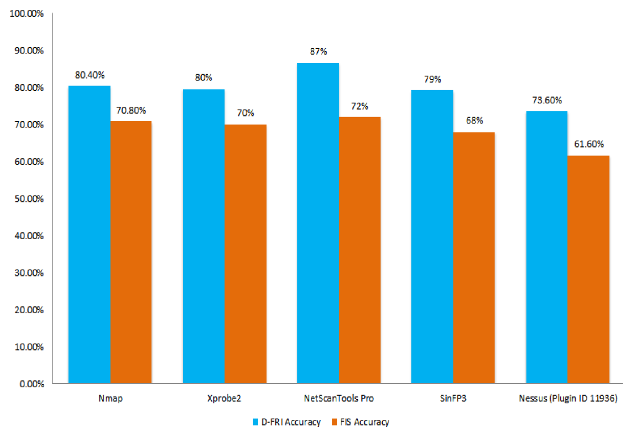


Fig. 20. Prediction Accuracy of D-FRI-Honeypot with D-FRI or without D-FRI (i.e., only FIS).

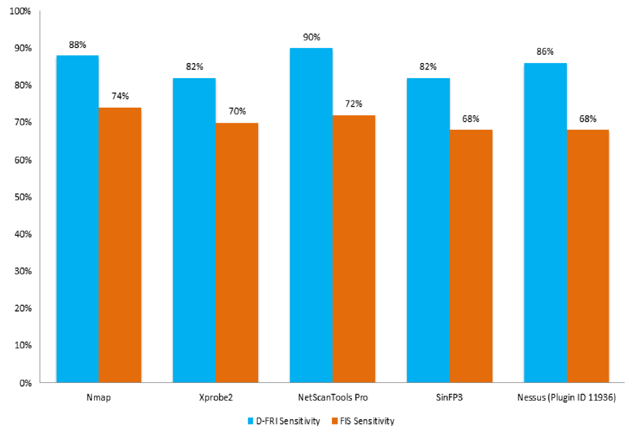


Fig. 21. Detection Sensitivity of D-FRI-Honeypot with D-FRI or without D-FRI (i.e., only FIS).

by utilising fuzzy inference and rule interpolation with the sparse rule base, reducing the failure rate of inference. Note that owing to the dynamic promotion of interpolated rules, computational overheads of interpolation are reduced as the detection process progresses. This is because the dynamic enrichment of the rule base will facilitate increasingly more direct rule firing without the need to carry out interpolation.

D. Analysis of D-FRI-Honeypot Results Running D-FRI

Examining the results of D-FRI-Honeypot with the use of D-FRI for the different OS fingerprinting tools reveals more interesting observations. In general, D-FRI-Honeypot has proven to be very effective in the prediction of all detected fingerprinting attacks, with an average rate of 93.55% predicted attacks being regarded at the *Very High/High* levels. In response to individual type of attacking tools, the following results are attained: 88.63% for Nmap, 100% for Xprobe2, 100% for NetScanTools Pro, 100% for SinFP3 and 79.1% for Nessus. This means that D-FRI-Honeypot can predict a majority of attempted OS fingerprinting attacks as *Very High/High* for all the five fingerprinting tools, as summarised in Table IV.

TABLE IV
PREDICTION BY D-FRI-HONEYPOT FOR ATTEMPTED ATTACKS USING DIFFERENT OS FINGERPRINTING TOOLS

OS Fingerprinting Tool	**Main Protocol for Reconnaissance**	**Predicted Severity Level of Attempted Attacks**	**Exceptions in Prediction of Attempted OS Fingerprinting Attacks**
Nmap	TCP-Based	$VeryHigh^*/High^*$ $Severity\ Level$	Protocols affected where HTTP-based attack scripts (e.g., *nmap -sV*) are used.
Xprobe2	ICMP-Based	$VeryHigh/High\ Severity\ Level$	
NetScanTools Pro	ICMP-Based	$VeryHigh/High\ Severity\ Level$	
SinFP3	TCP-Based	$VeryHigh/High\ Severity\ Level$	
Nessus	Both TCP and ICMP-Based	$VeryHigh^*/High^*$ $Severity\ Level$	Protocols affected where SMB, NTP, SNMP, SSH and HTTP-based attack scripts are used.

Where * means that D-FRI-Honeypot can predict *Very High/High* severity for majority of OS Fingerprinting Attacks, with *Exception* of certain application layer protocol-based attacks (see rightmost column).

Table IV also shows the exceptions where the predictions are made at a severity level different from *Very High/High*. These are mostly 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, whilst the present investigation is concentrated on the core protocols of the network and transport layer (TCP, ICMP, UDP and IP). Using HTTP and other application layer protocols, a reduced reliance on the core TCP/IP protocols results, in the process of obtaining OS fingerprinting information that may lead to the generation of lower TCP/IP traffic and fewer abnormalities/patterns for the system to predict. Whilst HTTP and other application layer protocols can be included in the detection process, with each protocol targeting a very specific attack, this will significantly increase the complexity and overheads of the proposed approach. With just core TCP/IP protocols included in all tools/attacks based on TCP/IP stack fingerprinting, a lightweight generic approach is made feasible, being capable of predicting all TCP/IP based fingerprinting attacks.

E. Accuracy of D-FRI Generated Dynamic Rules

In order to test the accuracy of the dynamically promoted rules for the D-FRI-Honeypot, a total of 20 new rules are generated from the collection of 400 interpolated rules. These new rules are added to the original sparse rule base to enhance the honeypot's ability to generate correct results and minimise future computational effort. Significantly, reduction of future interpolation also leads to a decrease of any errors caused by the interpolative process.

The accuracy of the dynamic rules is compared to: 1) that of directly using the interpolated rules ($\epsilon_{\%dvi}$), and 2) that of the *ground-truth* rules ($\epsilon_{\%dvt}$) which are generated on the basis of translating the underlying defining fuzzy grids, in exactly the

TABLE V
ACCURACY OF D-FRI GENERATED DYNAMIC RULES

Metric	$\epsilon_{\%dvi}$	$\epsilon_{\%dvt}$	$\epsilon_{\%ivt}$
AVG	2.48	**1.39**	2.64
SD	2.77	**1.40**	2.74

same way as utilised to create those original rules in the sparse rule base. Note that it is purely for experimental comparison purpose that such 'ground-truth' rules are provided assuming that there have been sufficient training data available for their creation. In reality, there do not exist such rules, otherwise there is no need for rule interpolation given a dense rule base.

The differences between the accuracies attainable by the use of just static rules for interpolation and that of the *ground-truth* rules ($e_{\%ivt}$) are also provided. In all aforementioned comparisons, the percentage error $\epsilon_{\%} = \epsilon / range_y$ is computed corresponding to the range of the consequent variable. Table V exhibits the average values and standard deviations with respect to the three measurements. The result clearly supports the present work, showing the benefits of promoting accurate rules through the application of D-FRI. Indeed, the results of the intelligent dynamic honeypot are closer to the use of the *ground-truth* rules.

VII. LIMITATIONS OF D-FRI-HONEYPOT

The D-FRI-Honeypot has been successfully tested against several fingerprinting tools in identifying a fingerprinting attack in-situ. However, it has a number of limitations:

- D-FRI-Honeypot may not always generate accurate results for untested fingerprinting tools, despite it has been developed according to the underlying attack principles of popular fingerprinting tools (Nmap and Xprobe2), and further tested for three other tools NetScanTools Pro, SinFP3 and Nessus.
- D-FRI-Honeypot may not generate accurate results for fingerprinting attacks that utilise unknown mechanisms other than those described previously.
- D-FRI-Honeypot is mainly focused on the core protocols TCP, UDP and ICMP, without considering the application layer protocols (e.g., HTTP, FTP, SMTP, SNMP) which may be exploited in a fingerprinting attack and thereby, may affect its prediction accuracy.
- D-FRI-Honeypot is developed for low-interaction honeypots where fingerprinting is a serious threat to revealing its identity; consequently, it is neither focused on nor recommended for high-interaction honeypots.
- D-FRI-Honeypot may generate false positives for those attacks which exhibit similar abnormalities or patterns to those identified in the empirical simulations.
- D-FRI-Honeypot produces prediction results that are indicative as they are in fuzzy linguistic terms; thus, further investigation to prove or disprove such a result is required if a boolean result is sought.
- D-FRI-Honeypot works depending upon the volume of traffic sampled as input to perform accurate detection of a fingerprinting attack; therefore, any obstacle in traffic may affect the entire identification and prediction process.

VIII. CONCLUSION

This paper has presented a dynamic fuzzy rule interpolation based honeypot called D-FRI-Honeypot for detecting or predicting active fingerprinting attacks, including their severity levels. Earlier, there was little work available to detect and predict an attempted fingerprinting attack on honeypots. This observation has led to the investigation of various fingerprinting attacks and the development of an intelligent and adaptive D-FRI-Honeypot in the research reported herein.

The design of D-FRI-Honeypot has been focused on the most common OS fingerprinting attacks. The simulation of fingerprinting attacks and data (TCP/IP packets) collection have both been accomplished by employing *KFSensor* honeypot tool and *Nmap* and *Xprobe2* fingerprinting tools. Based on preliminary observations and empirical evidence, important fields of collected TCP/IP packets have been analysed in an effort to establish abnormalities or patterns as a sign of an attempted fingerprinting attack. Subsequently, PCA has been utilised to determine the most influential TCP/IP fields, which are then used to develop the honeypot based on D-FRI. D-FRI-Honeypot has been successfully tested against five popular fingerprinting tools. The analysis of experimental results achieved has demonstrated that D-FRI-Honeypot can significantly improve the *prediction accuracy* and *detection sensitivity*, covering a majority of attempted OS fingerprinting attacks, while possessing the ability of accurately and dynamically enriching the system's own knowledge base online.

Whilst D-FRI-Honeypot is promising, encompassing several types of TCP/IP based fingerprinting attacks, it may omit certain fingerprinting attacks that take advantage of application layer protocols such as HTTP, SMTP, FTP, SMB, NTP, SNMP and SSH. Thus, as an important future work, it is essential to enhance D-FRI-Honeypot so that it could incorporate these fingerprinting attacks. Of course, work also remains to be done in dealing with those shortcomings identified in the preceding section.

REFERENCES

[1] S. Balaganur, "Top AI-based attacks that shocked the world in," 2019. [Online]. Available: https://analyticsindiamag.com/top-ai-based-attacks-that-shocked-the-world-in-2019/

[2] K. OFĺaherty, "AI: A new route for cyber-attacks or a way to prevent them?," 2019. [Online]. Available: https://www.information-age.com/ai-a-new-route-for-cyber-attacks-or-a-way-to-prevent-them-123481083/

[3] M. Vizard, "Cisco report confirms cyber attacks more sophisticated," 2018. [Online]. Available: https://securityboulevard.com/2018/02/cisco-report-confirms-cyber-attacks-more-sophisticated/

[4] C. Cantrell, M. Willebeek-LeMair, D. Cox, J. McHale, B. Smith, and D. Kolbly, "Active network defense system and method," US Patent 7,454,499, Nov. 18 2008.

[5] R. Joshi and A. Sardana, *Honeypots: A New Paradigm Inf. Secur.*, CRC Press, 2011.

[6] N. Naik, C. Shang, P. Jenkins, and Q. Shen, "Building a cognizant honeypot for detecting active fingerprinting attacks using dynamic fuzzy rule interpolation," *Expert Syst.*, 2020.

[7] R. A. Grimes, *Honeypots, Hacking Hacker: Learn Experts Who Take Down Hackers*. John Wiley & Sons, Inc., Indianapolis, Indiana, 2017, doi: 10.1002/9781119396260.ch19.
[8] L. Spitzner, *Honeypots: Tracking Hackers*. Addison-Wesley Reading, 2003, vol. 1.
[9] N. Naik, P. Jenkins, N. Savage, and L. Yang, "A computational intelligence enabled honeypot for chasing ghosts in the wires," *Complex Intell. Syst.*, 2020.
[10] D. Dubois and H. Prade, "On fuzzy interpolation," *Int. J. Gen. Syst.*, vol. 28, no. 2-3, pp. 103–114, 1999.
[11] L. Koczy and K. Hirota, "Approximate reasoning by linear rule interpolation and general approximation," *Int. J. Approx. Reasoning*, vol. 9, no. 3, pp. 197–225, 1993.
[12] S. Saga, H. Makino, and J. I. Sasaki, "A method for modelling freehand curves - The fuzzy spline interpolation," *Syst. Comput. Jpn.*, vol. 26, pp. 77–87, 1995.
[13] N. Naik, R. Diao, and Q. Shen, "Dynamic fuzzy rule interpolation and its application to intrusion detection," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 4, pp. 1878–1892, Aug. 2018.
[14] N. Naik, R. Diao, and Q. Shen, "Application of dynamic fuzzy rule interpolation for intrusion detection: D-FRI-Snort," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2016, pp. 78–85.
[15] N. Naik, R. Diao, C. Shang, Q. Shen, and P. Jenkins, "D-FRI-WinFirewall: Dynamic fuzzy rule interpolation for Windows firewall," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2017, pp. 1-6.
[16] N. Naik, C. Shang, Q. Shen, and P. Jenkins, "D-FRI-CiscoFirewall: Dynamic fuzzy rule interpolation for Cisco ASA Firewall," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2019, pp. 1-6.
[17] N. Naik, C. Shang, Q. Shen, and P. Jenkins, "Intelligent dynamic honeypot enabled by dynamic fuzzy rule interpolation," in *Proc. 4th IEEE Int. Conf. Data Sci. Syst.*, 2018, pp. 1520–1527.
[18] N. Naik, C. Shang, Q. Shen, and P. Jenkins, "Vigilant dynamic honeypot assisted by dynamic fuzzy rule interpolation," in *Proc. IEEE Symp. Series Comput. Intell.*, 2018, pp. 1731-1738.
[19] E. H. Mamdani and S. Assilina, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Mach. Stud.*, vol. 7, no. 1, pp. 1–13, 1975.
[20] Z. Huang and Q. Shen, "Fuzzy interpolative reasoning via scale and move transformations," *Fuzzy Syst., IEEE Trans.*, vol. 14, no. 2, pp. 340–359, Apr. 2006.
[21] N. Naik, P. Jenkins, R. Cooke, and L. Yang, "Honeypots that bite back: A fuzzy technique for identifying and inhibiting fingerprinting attacks on low interaction honeypots," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2018, pp. 1-8.
[22] N. C. Rowe, "Measuring the effectiveness of honeypot counter-counterdeception," in *Proc. System Sci.,. HICSS'06. Proc. 39th Annu. Hawaii Int. Conf.*, vol. 6., 2006, pp. 129c–129c.
[23] N. Naik and P. Jenkins, "A fuzzy approach for detecting and defending against spoofing attacks on low interaction honeypots," in *Proc. 21st Int. Conf. Inf. Fusion.*, 2018, pp. 904–910.
[24] N. Naik and P. Jenkins, "Discovering hackers by stealth: Predicting fingerprinting attacks on honeypot systems," in *Proc. IEEE Int. Symp. Syst. Eng.*, 2018, pp. 1-8.
[25] J. M. Allen, "OS and Application Fingerprinting Techniques," 2008. [Online]. Available: https://www.sans.org/reading-room/whitepapers/authentication/os-application-fingerprinting-techniques-32923
[26] N. Naik, P. Jenkins, and N. Savage, "Threat-aware honeypot for discovering and predicting fingerprinting attacks using principal components analysis," in *Proc. IEEE Symp. Series Comput. Intell.*, 2018, pp. 623-630.
[27] G. F. Lyon, "Chapter 9. Nmap Scripting Engine," 2009. [Online]. Available: https://nmap.org/book/nse-usage.html
[28] L. G. Greenwald and T. J. Thomas, "Toward undetected operating system fingerprinting," *WOOT*, vol. 7, pp. 1–10, 2007.
[29] G. F. Lyon, "Chapter 15. Nmap Reference Guide," 2009. [Online]. Available: https://nmap.org/book/man-os-detection.html
[30] G. F. Lyon, "Chapter 7. Service and Application Version Detection," 2009. [Online]. Available: https://nmap.org/book/vscan.html
[31] G. F. Lyon, "Nmap OS Fingerprinting 2nd Generation DB," 2017. [Online]. Available: https://svn.nmap.org/nmap/nmap-os-db
[32] G. F. Lyon, "Nmap Service DB," 2011. [Online]. Available: https://svn.nmap.org/nmap/nmap-services
[33] O. Arkin and F. Yarochkin, "A fuzzy approach to remote active operating system fingerprinting,"2003. [Online]. Available: http://www.sys-security.com/archive/papers/Xprobe2.pdf
[34] F. V. Yarochkin, O. Arkin, M. Kydyraliev, S.-Y. Dai, Y. Huang, and S.-Y. Kuo, "Xprobe2++: Low volume remote network information gathering tool," in *Proc. Dependable Syst. Netw., DSN'09. IEEE/IFIP Int. Conf.*, 2009, pp. 205–210.
[35] O. Arkin and F. Yarochkin, "Xprobe2(1) - Linux man page," 2018. [Online]. Available: https://linux.die.net/man/1/xprobe2
[36] J. Touch, "Updated specification of the IPv4 ID Field," 2014. [Online]. Available: https://tools.ietf.org/html/rfc6864
[37] H. F. Kaiser, "The application of electronic computers to factor analysis," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 141–151, 1960.
[38] A. B. Costello and J. W. Osborne, "Best practices in exploratory factor analysis: Four recommendations for getting the most from your analysis," *Practical Assessment, Res. Eval.*, vol. 10, no. 7, pp. 1–9, 2005.
[39] W. R. Zwick and W. F. Velicer, "Comparison of five rules for determining the number of components to retain," *Psychol. Bulletin*, vol. 99, no. 3, p. 432, 1986.
[40] J. L. Horn, "A rationale and test for the number of factors in factor analysis," *Psychometrika*, vol. 30, no. 2, pp. 179–185, 1965.

Nitin Naik received the Ph.D. degree in computer science from Aberystwyth University, Aberystwyth, U.K. He additionally holds several academic qualifications: M.Tech., M.Sc., MBA, MSW, B.Sc., and Polytechnic (Electrical Engineering). He has authored or coauthored more than 100 peer-reviewed papers in the areas of artificial intelligence, cybersecurity, big data, cloud computing, Internet of Things, and game based learning. He is currently a Senior Lecturer with the School of Informatics and Digital Engineering, Aston University, Birmingham, U.K.

Changjing Shang received the Ph.D. degree in computing and electrical engineering from Heriot-Watt University, Edinburgh, U.K., in 1995. She is currently a University Research Fellow with the Department of Computer Science, Aberystwyth University, Aberystwyth, U.K. She has authored or coauthored more than 150 peer-reviewed papers and supervised 15 Ph.Ds/PDRAs, in the areas of pattern recognition, data mining and analysis, and reasoning with uncertainty.

Paul Jenkins received the Ph.D. degree in applied mathematics and computing from Cardiff University, Cardiff, U.K. He has authored or coauthored more than 50 peer-reviewed papers in the areas of artificial intelligence, cybersecurity, big data, cloud computing, Internet of Things, and game based learning. He is currently a Senior Lecturer with the Cardiff School of Technologies, Cardiff Metropolitan University, Cardiff, U.K.

Qiang Shen received the Ph.D. degree in computing and electrical engineering from Heriot-Watt University, Edinburgh, U.K., in 1990, and the D.Sc. degree in computational intelligence from Aberystwyth University, Aberystwyth, U.K., in 2013. He holds the Established Chair of Computer Science and is Pro Vice-Chancellor for Faculty of Business and Physical Sciences, Aberystwyth University. He has authored two research monographs and more than 400 peer-reviewed papers.