IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 19 March 2023, accepted 9 April 2023, date of publication 3 May 2023, date of current version 17 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3272629

RESEARCH ARTICLE

Toward Effective Evaluation of Cyber Defense: Threat Based Adversary Emulation Approach

ABDUL BASIT AJMAL[1], SHAWAL KHAN[1], MASOOM ALAM[1], ABOLFAZL MEHBODNIYA[2], (Senior Member, IEEE), JULIAN WEBBER[2], (Senior Member, IEEE), AND ABDUL WAHEED[3]

[1]Department of Computer Science, COMSATS University Islamabad (CUI), Islamabad 45550, Pakistan
[2]Department of Electronics and Communication Engineering, Kuwait College of Science and Technology (KCST), Doha, Safat 35003, Kuwait
[3]Department of Computer Science, Women University Swabi, Swabi 23430, Pakistan

Corresponding authors: Shawal Khan (shawalsbbu@gmail.com) and Abdul Waheed (abdul@netlab.snu.ac.kr)

ABSTRACT Attackers compromise organizations with increasingly sophisticated ways, such as Advanced Persistent Threat (APT) attackers. Usually, such attacks have the intention to exploit endpoints to gain access to critical data. For security controls and defense evaluation, organizations may employ offensive security activities. The most important one is penetration testing and red teaming, but such operations are usually resource exhaustive and extend over a longer period of time. Furthermore, traditional Venerability Assessment and Penetration Testing (VAPT) works effectively in the mitigation of known attacks but did not prove to be effective against stealthy attacks. VAPT considers the whole offsec as an acting problem but in reality, an attacker has to deal with uncertainty while conducting real-world attacks. In this paper, we are presenting an adversary emulation approach based on MITRE ATT&CK adversary emulation plan with consideration of planning as a major part of each attack phase. The approach utilizes stealthy attack vectors and paths to emulate adversary for defense evaluation. For effective defense evaluation, we picked more than 40 techniques from ATT&CK, deployed their mitigation on target machines, and then launched attacks against all those techniques. We show that attack paths and payloads generated using our approach are strong enough to evade security controls at endpoints. This approach provides a special environment for cyber defenders to think like adversary, and create new attack vectors and paths to evaluate organizational security preparedness. This process constructs a special environment to expand the attack landscape view and defense evaluation with minimal resources for the organization.

INDEX TERMS ATT&CK predictions, endpoint security evaluation, cyber attack simulations, penetration testing, stealthy attacks, defense evaluation.

I. INTRODUCTION

Threat of cyber attacks continues to increase as cybercriminals become more sophisticated and organizations rely more heavily on technology. Recent stats show a drastic increase in cyber-attacks targeting endpoints. Such as servers, cell phones, and workstations. Endpoints are considered as the most valuable and vulnerable devices. One example is the use of "business email compromise" (BEC) [1] attacks, in which attackers impersonate executives or vendors to trick employees into providing sensitive information. Another example is the use of ransomware, in which attackers encrypt a company's data and demand a ransom payment to restore access. The threat of cyber attacks continues to increase as cybercriminals become more sophisticated and organizations rely more heavily on technology. Advancement of technology has posed increased threats as the number of endpoint nodes are increasing so endpoints security must be prioritized. Thus endpoint security is considered as the future of cybersecurity [2].

Many organizations conduct penetration testing periodically to determine the presence of potential vulnerabilities [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Young Jin Chun.

Such testing aims to evaluate the security controls adopted by the organization. Sample penetration tools and methods are discussed in [4]. Usually, organizations have adversary simulation teams on board to run these offensive activities as a ''cat and mouse'' game. One team is responsible for launching attacks and the other team is responsible for detecting them, that's how they evaluate security. This proved to be an effective approach, with one drawback: the red team's operations are resource exhaustive. In a changing threat landscape, where attackers are employing increasingly sophisticated attacks, organizations are more prone to cyber-attacks. Modern solutions, such as models for vulnerability scanning, vulnerability management, vulnerability mitigation and Vulnerability Assessment and Penetration Testing (VAPT), rely on ''known threats'', while we often see attackers exploiting unknown and zero-day vulnerabilities. Recent solutions tried to alleviate this situation by exploring control based evaluation [5], but this approach is still prone to zero days attacks.

Adversary emulation in cyber security is a technique used to simulate real-world cyber attacks in order to test an organization's security defenses. The process typically involves simulating the tactics, techniques, and procedures (TTPs) of known or hypothetical attackers in order to identify vulnerabilities and measure the effectiveness of security controls. This can include simulating phishing campaigns, malware attacks, and other types of cyber threats. The goal of adversary emulation is to improve an organization's security posture by identifying and mitigating potential vulnerabilities before they can be exploited by real attackers. It is also known as ''red teaming'' or ''threat emulation''. It can be done internally by the organization's security team or by hiring an external company to conduct the testing. In this paper, we show that employing threat-based emulation is an effective solution for evaluating security from an adversarial perspective, rather than performing full-scale red team operations.

In this research article, we introduce a threat-based adversary emulation approach that addresses the limitations of traditional penetration testing and red teaming techniques and provides a more effective and realistic simulation of real-world attacks. Our approach utilizes Mitre ATT&CK to learn TTPs to continuously adapt the attack simulation to the organization's evolving security posture. One of the key advantages of our approach is the ability to provide a more realistic simulation of real-world attacks, as it allows the red team to mimic the behavior of advanced persistent threats (APTs) and other sophisticated attackers. Additionally, by continuously adapting the attack simulation, our approach allows organizations to stay ahead of evolving threats and improve their overall security posture. For an operator who is conducting threat-based adversary emulation, it's necessary to know about malware classes. Malware classifications are given in Table 1. Most common among them are droppers which assist adversaries to download any type of malware. Furthermore, we propose a threat-based adversary emulation approach with payload generation algorithms, which consists of unique agnostic methods, with similar capabilities which are carried by real-world attack vectors. Moreover, it can be used to generate new attack paths. We provide four algorithms for generating stealthy attack vectors and use them to emulate adversarial TTPs. During experimentations, we generated different payloads and tested them against endpoint security solutions for the prevention and detection of threats such as EDR/AV (endpoint detection and response/ antivirus). That's how we evaluated the effectiveness of stealthy payloads. We evaluated our approach and mapped test case payloads according to ATT&CK framework, which is a worldwide free-to-use database of adversarial knowledge based on TTP. Moreover, this approach has the ability to recreate new methods for similar techniques on ATT&CK. We evaluated our approach and algorithms against endpoint security and concluded that this approach is effective for launching dynamic threats as well as for ''emerging threats'' assessment.

TABLE 1. Malware classification.

Class	Description
Virus	Propagates with user intervention
Dropper	Downloads malware
Worm	Self propagating
Rootkit	Masks its existence
Backdoor	Gives access to remote attacker
Ransomware	Encrypts data and hold it as hostage
PUP/PUA	Potentially unwanted program

TABLE 2. Payload attack vectors.

Type	Description
OLE files	VBA scripts in openXML file
PDF	Hidden malicious file at EOF
DLL	Malicious dynamic link libraries
PE	Executable .exe file
Bat file	Malicious .bat file
PS1	PowerShell script
Stegno Payload	Hidden payload

For this research paper, we have considered Windows system and Linux (privilege escalation only), in table 2 attack vector details are provided in form of payloads that are used in this research. More specifically we have used portable executable files, OpenXML files and malicious scripts (CMD/PS1/Bat) for the experiment. Open XML files are zip archive files containing different XML files such as styling and structure files. The customized raw malicious payload in .bin file is used to embed with macros. Mentioned attack vectors in table 2, can be generated by algorithms in section IV.

The rest of this paper is organized as follows: Immediate section introduces related work on adversary emulation and different penetration testing & vulnerability assessment approaches which is section II. In section III, we propose a formal model for threat-based adversary emulation. In section IV, we present an algorithmic implementation of a prototype of our model. In section V and VII, we present the

TABLE 3. Pentesting vs. adversary emulation.

Pentesting	Adversary Emulation
Identify Vulnerabilities	Assess how strong organizational defenses are
Limited scope with specific focus point	More Inclined on scenario execution
More focused on prevention	Typically tests both prevention and detection

results of the evaluation of our approach. Finally, we conclude this paper in section VIII with a few concluding remarks and future extensions.

II. RELATED WORK

The contribution of this paper is the introduction of an effective process for threat-based adversary emulation that provides output as quantifying the overall integrity, coverage, and rigor of security controls. Additionally, our approach allows for more efficient use of resources and focusing on manual efforts in areas that are most likely to be targeted by attackers. In this section, we review existing literature on VAPT and Mitre ATT&CK and controls-based security.

When organizations started realizing that attackers tend to exploit vulnerabilities present in our network, they came up with different methods for countering them. One of them was keeping systems patched. Adversaries were still unbeatable. ''Best Offense is the Best Defense'', Organizations started thinking from the perspective of attackers and started exploiting vulnerabilities before attackers do so and fix vulnerabilities. This approach is called ''Pen-testing''. Pen-testing has evolved over time and is now usually referred to as ''VAPT''. This approach is adequate for small and medium-size organizations for countering beginner and intermediate-level attackers. Network attacks are elaborated here [6], [7].

VAPT helps organizations to assess how effective their security solutions are. In [8], the authors provide an overview of various techniques used in vulnerability assessment and penetration testing. Past researches [9], [10] considered vulnerability management and continuous installing of patches as a solution for securing organization. Recent research, such as the penetration testing approach for exploiting mobile devices [11], considered testing of common security controls. Almost all these solutions are prone to different types of attacks such as zero days and social engineering.

Many organizations conduct Red Team exercises to evaluate their security. Such penetration testing and red team operations are thoroughly discussed in [12] and [13]. Detailed Discussion on APT is explained in [14]. Recent research has also considered re-developing existing popular pen-testing suites [15]. Such types of implant redevelopment techniques are widely used by ethical hackers to conduct testing of security mechanisms. We have extended this approach and integrated it into adversary emulation. A comparison of pen-testing and adversary emulation is given in table 3.

Such real-world attack emulation (adversary emulation) provides*live fire* training opportunities for analysts. Threat hunting vendor FireEye [16] has explained well how to extract techniques from IOCs and logs. We used this approach in adversary emulation and extended it with mappings on MITRE ATT&CK. Figure 1 explains, how techniques look like. We integrated this into our approach during the threat intelligence phase with the aid of open-source projects which make work easy by quickly analyzing threat reports. Automated penetration testing based on a threat model [17] is a recent work on penetration testing based on threats specific to an organization. However, this has limited scope and makes it prone to zero days and APT threats. Penetration testing with Metasploit is widely used by different organizations and is well explained in [18]. Moreover, it discusses different methods for evading security controls to make penetrating close to real-world attacks [19].

Adversary emulation assessments offer defenders the ability to view their networks from the point of view of an adversary. In [19] and [20], the authors have discussed the formal use of open-source tools such as ''Atomic Red Team'' to conduct adversary emulation and build test cases against TTPs from MITRE. We derived our planning approach from this paper. This research article [19], [21] is not agile in working with a lot of attack techniques and has a limited scope to emulate adversaries. For example, if we want to launch APT41 attack cases. Some questions arise, such as how to order the sequence of attack cases. How to build attack chains and use all collected information after emulation for security awareness of the organization. The proposed approach is a continuation of our previous research work [22], [23], [24], [25].

MITRE has provided some sample attack plans which depict the practical use of knowledge base [26]. We have used a modified form of these plans to incorporate the latest methods and techniques, in the ''organizing and analysis'' phase of our approach. ATT&CK knowledge base has diverse applications. One of them is the use of knowledge bases as threat intelligence. During this phase, noticeable action of the adversary can be mapped on different tactics on ATT&CK. There are many platforms, most of them are open source and support ATT&CK mappings. Such as MISP [27] and OpenCTI [28]. We have leveraged this and used it in a hybrid mode to learn more about specific threats and adversarial techniques (from organizations sharing threat Intelligence).

III. ADVERSARY EMULATION APPROACH

The agnostic threat-based adversary emulation approach involves simulating a wide range of potential attacks, including both known and unknown threats. This can include simulating attacks from multiple different types of attackers, such as nation-state actors, cybercriminals, and hacktivists. It also involves using a variety of tactics, techniques, and procedures (TTPs) to test an organization's defenses, such as phishing, social engineering, and malware.

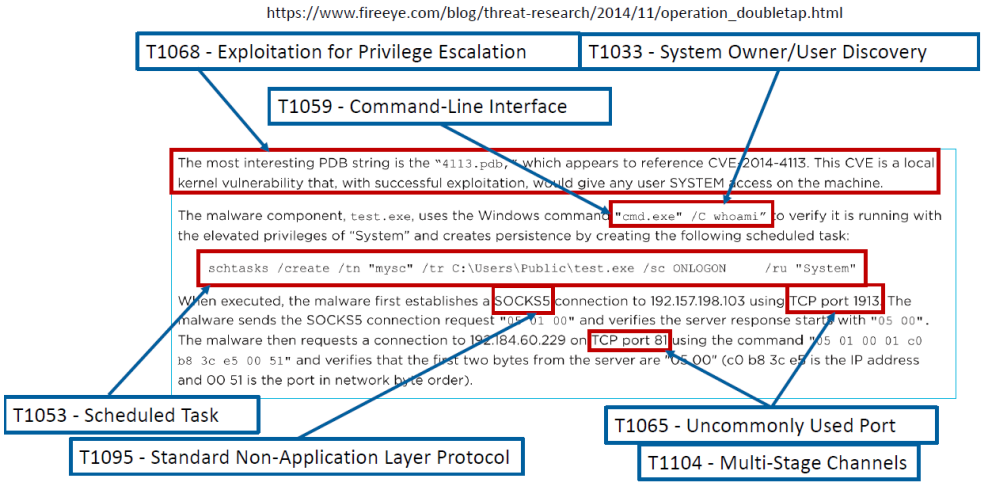

FIGURE 1. Threat research: Extracting techniques [16].

The goal of this approach is to identify vulnerabilities and measure the effectiveness of security controls in a more realistic and comprehensive way. It allows organizations to test their defenses against a wide range of potential threats and better understand their overall security posture. It also allows organizations to stay ahead of evolving threats by continuously testing and updating their defenses.

The first phase is to define the purpose and objective of the adversary emulation exercise. Moreover, at this stage, we also limit our scope and identify systems that are under test. This gives the whole activity a strong ground to carry out ''Threat Research'' which focuses on learning and exploring adversarial TTPs. Once a base for emulation is built, the tester can choose an adversary which matches with organization-specific threats and start mapping identified threats on the ATT&CK framework. At this stage, the tester has knowledge about threats. Testers can use publicly available forums/blogs like ''Palo Alto Unit 42'', ''Kaspersky Threat Research'', ''FireEye Threat Research'', and ''Shadowpad, or Malpedia'' for new attack techniques and vectors. Following the activity, extraction of techniques comes into play here, where testers try to find different attack paths with the aid of automated and manual tools. Once this process is done testers organize the techniques and order them in a sequence according to ATT&CK Tactics from ''initial access'' to ''ex-filtration''. Moreover, at this stage, the attack plan is prepared, and testers can build several plans to deal with uncertainty. Testers can now verify the scope and move on to searching for open-source tools that can be used to build tools for adversary emulation. Tools must be picked carefully as they must be able to emulate all extracted techniques and paths. Once tools are built, testers can execute plans. As shown in below Figure 2.

A. PURPOSE, OBJECTIVES & GOALS

The initial management phase in adversary emulation is crucial for the successful and effective testing of a system's security measures. It allows for clear objectives to be established, specific attack scenarios to be developed, and proper evaluation of the system's defenses. The first phase ''Purpose'', is a more managerial side task but it's important as it lays the foundation of the whole adversary emulation and answers critical questions such as Why do we need to conduct adversary emulation? Purpose of conducting adversary emulation? What are the outcomes expected? Next tasks come into play where we list down all ''Objectives'' and ''Goals'' of conducting adversary emulation in a form of a small report that will be aligned with the stakeholders of the target environment. The last step of this phase is to specify any requirement by the adversary emulation team from the organization. For example, if the team is required to do testing at a specific time or on specific systems only. For feasibility analysis, the team may visit an organization to check which scale of adversary emulation can be conducted. All achievable derivatives from the adversary emulation process must be defined here.

1) SCOPE

The operational flow for the whole process is defined here. Which defines how the whole flow will go. Division of duties and work is defined here. Moreover, Segregating roles are done here. For example, Team A, could be responsible for intelligence related to initial access. All teams shall document each phase and process for reporting. Systems under test are identified here this includes services or network-level services and devices.

B. THREAT RESEARCH

This phase is divided into ''Active & Passive Research''. Active scanning includes reverse engineering, static and dynamic analysis of malware samples. We learn actual techniques implementation details in **active research**, where we analyze different malware samples and map their detected techniques on ATT&CK. This process is done by using

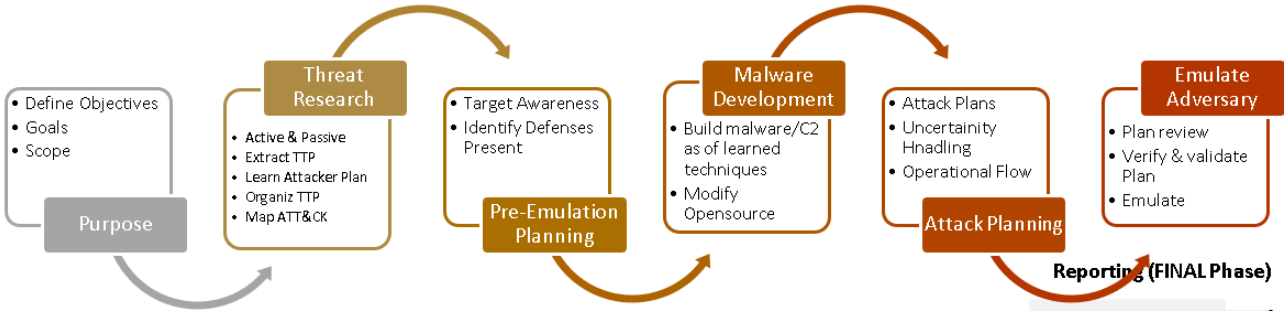

FIGURE 2. Adversary emulation process.

TABLE 4. Threat intelligence & TTP's.

Source	Type	Goal
Virusbay, ViruShare, Malshare, VX vault	Active	Extract TTP
MISP, OpenCTI, Malpipe, Unit 42, WeLive Security	Passive	extract TTP

forensic and malware analysis tools such as "FireEye Open-source CAPA framework" and YARA rules. **Passive threat** research includes gathering threat intelligence from threat research forums, blogs, threat reports, threat news, malware intelligence platform, threat intelligence platform such as MISP, AlienVault Threat, OpenCTI and CISCO Talos Intelligence platform. Such frameworks are updated on a daily basis with emerging, advancing threats from all over the world. After that, we have to categorize them in such order that curated techniques and methods answer, "who do they target?", "What are they behind for?".

This type of intelligence is vital in threat-based adversary emulation. This lets, emulate adversaries as it is with slightly new techniques to make sure security controls in place are enough for countering them. We are using rcATT, the original project is present on github.com/vlegoy/. rcATT works on ATT&CK predictions, the source of techniques predictions come from ATT&CK for example, technique IDs and their high-level objectives "tactics". This tool inputs text and on the basis of specific instincts of techniques present in the text and matches it from technique instincts from ATT&CK and predict the Technique ID. Threat intelligence gathering could be minimized if we define the specific scope for threat intelligence. For example, if the defined scope says that "We are gathering intelligence related to C2 evasion methods", then we would stick to the scope only.

C. PRE-EMULATION PLANNING

Optional phase used to gather information about security mechanisms deployed at the target and hence named "Pre adversary emulation plan activity". For example, during passive/active reconnaissance; it was identified that Symantec endpoint security is installed, and what utilities/tools/softwares are installed on the target machine. The result of this phase is utilized in building payloads and tools that can certainly evade security, for example, payload developers can test payloads in advance to make sure they can evade security. This phase is also known as situational awareness about the target.

D. LEARNING ADVERSARIAL TECHNIQUES

"Tactics" are the higher-level adversary objectives and these objectives are set in a form of steps. These series of high-level steps can be compared with ATT&CK to get an understanding of the attacker's goal. According to "Pyramid of Pain by David Bianco" [29], TTP is the toughest task and here we could utilize threat research and ATT&CK framework. Following steps are included in this process. Following is the approach to extract techniques from threat intelligence. At this stage, we can use tools like vlegoy/rcATT for predicting ATT&CK techniques from threat reports.

- *Step 1* Ordering learned techniques. For example: malware used dll unhooking technique to evade EDR. If we map this technique on the MITRE ATT&CK framework, it comes under defense evasion.
- *Step 2* Organize technique flow in process. For example, the first adversary used different techniques for hash stealing and then used it for password spraying to get access to the system.

At this stage, we have techniques, and their corresponding technique IDs. But still, this is not in a structured way. Now we aim to organize these techniques according to high-level attack objectives also known as tactics. In this way, we map techniques and order them.

E. MAPPING & ORGANIZING TECHNIQUES

In this process, we try to understand adversary goals and try to map techniques flow into adversary plans. Our approach is more concerned with the latest learned techniques from adversaries, in this we can add learned techniques in "sub-techniques in ATT&CK framework". If learned techniques are not present in ATT&CK then we can build a table and techniques name and their method. Basic working at this phase is to map techniques on ATT&CK and put them into the adversary plan, this will give a vivid view of the adversary plan. At this stage tools like rcATT

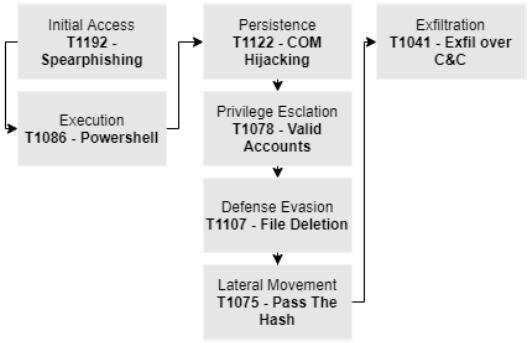

FIGURE 3. Emulation plan & operational flow.

(github.com/vlegoy/rcATT) can be used to extract techniques mapping. For example the following is the example of APT28 [30] details can get from MITRE 3 APT3 plan [31].

F. UNCERTAINTY HANDLING

In reality an attacker has to deal with uncertainty while launching an attack. For such cases we can expect that our attack plan may fail due to uncertainty, so we need to be prepared for such scenarios. In this phase we build different plans that can be used in case of uncertainty. For example, we failed in delivering payload. In such a case our counter plan can consist of out of the box techniques such as delivering payload in HEX over IPv6 requests and resembling at target and load it reflectively (file-less).

G. MALWARE AND TOOLS DEVELOPMENT

This phase can also be referred to as "**Arsenal Building**" as the arsenal is usually the place where we place or store weapons in a way that they can be used in an effective way when needed. In adversary emulation this concept is similar. The difference is "this arsenal is digital and in non-tangible form". Arsenal of adversary emulation exercise consists of offensive tools, scripts, exploits, domains, servers whatever we can use in adversary emulation to achieve our objectives. Our arsenal can consist of combined tools, in house build or open source.

Building new tools is a costly process, at this stage we aim to do changes in existing tools by mapping new techniques on tools. Writing malware from scratch is not a good idea. So, here we can use shellcode from MSF/CobaltStrike and using different injections techniques to execute shellcode in memory. Typical process injection. Following are the steps for building tools.

- *Step 1* Search tools on github and read recent Crowdstrike Falcon report to find mostly used open source tools by adversary
- *Step 2* Identify process specific requirements
- *Step 3* Identify tool capabilities and your requirements
- *Step 4* Build attack vectors or modify existing
- *Step 5* Defining tool-set

Usually, the adversary uses an initial custom-built dropper and later on they use Cobalt Strike beacon and directly inject it into memory using reflective injection. For example, we decided to use a meterpreter. In the second step, we identify the requirements for specific processes such as the .exe or .dll loading mechanism available with Metasploit, and how it can bypass EDR/AV. Here we try to identify if it meets the requirements as of the designed requirements. If it meets the requirement then we can use it otherwise we will modify it.

H. BUILDING ATTACK PLANS AND UNCERTAINTY HANDLING

This phase is the continuation of the previous step, where we design and build a strategy for launching an attack by defining a series of steps in a stealthy way to achieve our objectives. We build 3 to 4 attack plans depending on attacker's capability under consideration, for example, **Plan A** is a straightforward plan with tools developed tools at lab, **Plan B** is inspired for advanced attackers consisting of sophisticated evasion and delivery methods, **Plan C** is inspired by an apex attacker which relies on totally out-of-the-box methods and techniques for achieving goals. The rest of the plans are developed only to address uncertainty if it occurs, for example, if we have listed LLMNR for credential access; but this protocol is disabled by the system administrator, in this case, what should we do? All such answers are done in uncertainty plan building.

Plan is built by identifying a set of actions that can give a green pass toward achieving objectives. For example, if we exploit a vulnerability with CVE assigned to it then what can be the impact of doing this? Maybe the system is already patched then we can use a remote desktop or mount share or dump credentials or copy files. This phase is a conjunction of **planning and performing** tasks. As planning plays a crucial role in the success of a plan using early awareness of target and threat research.

I. PLAN REVIEW

Plan review is the last phase before emulating an adversary. At this stage we make sure that process-specific requirements meet with the tools and answer questions like; malware that we are using is it in accordance with goals that are set initially? If we are unable to validate any of the above activities, then we go back one step and repeat the process until all objectives of emulation are in accordance with adversary emulation.

J. LAUNCH

At this stage, we aim to emulate attack tactics by emulating techniques corresponding to each tactic.

K. REPORTING

This is the last phase of the adversary emulation approach where we document our activity throughout the approach. Reporting is further divided into two phases: technical and. . .non-technical. The **technical report** is targeted for

experts only and **non-technical report** is a a friendly report which explains, organizational ability to stand against real-world attacks with highlighted critical assets with high-risk factors.

L. SET REPRESENTATION OF ADVERSARY EMULATION APPROACH

We are defining our approach as a tuple (`R,P,O,I,T,A,E,L,F`) where set `R` represent prerequisites, `P` set consists of purpose, `O` set contain goals, objectives and scope `I` is intelligence set, `T` is an techniques set, `A` is an organized techniques set, `E` is an environment details set, `L` is set containing actual emulation details set. `F` is a feedback & reporting set containing feedback from each phase and builds a report at the end of emulation.

$$P = objective(P \times R) \tag{1}$$

Objective `O` defines goals of the whole operation formally. `objective-final()` utilizes different procedures including meeting between adversary emulation teams to formally define adversary emulation objectives and goals. P represents the set of information about the purpose of adversary emulation. R represents the set of prerequisites, of operation

$$O = objective - final(P) \tag{2}$$

Intelligence set I represents intelligence about adversaries `I=(openCTI, ATT&CK). Intel()` function contains procedures for sorting raw intelligence, analytical questions `(AQ)` are built and answered about adversaries that can be a hurdle in achieving organizational goals.

$$I = intel - gather(I \times AQ) \tag{3}$$

Technique set T represents adversary and a subset of sets. A represents organized adversary tactics and techniques. Set T includes two subsets which includes a set of structured techniques and another set is ordered techniques. Set A uses set T and maps those techniques on ATT&CK framework. Function `analyze_adv()` have procedures for understanding adversary goals and ordering techniques according to tactics.

$$A = analyze - organize(T \times A) \tag{4}$$

Environment building set E represents the information about requirements for the environment required and other available environments that can be utilized in the building requirement environment. Function `build_env()` passes the above data to procedures to get the environment.

$$E = build - env(A \times E) \tag{5}$$

Launch set L contains all vital information regarding adversary emulation such as target, which exploits we will use, how we will bypass security mechanisms. Evasion and exploitation is at the heart of this process. EV is a subset of the L set, this set represents evasion techniques that are developed in previous phases.

$$L = launch - plan(E \times EV) \tag{6}$$

Reporting set F is a set of feedback from each phase and passes this information to function `report()` to build a report at the end of adversary emulation.

$$F = reporting(A \times E \times L) \tag{7}$$

Reporting set contains each and every details from each phase and feed this data to reporting() to generate report, like how memory injections are done, IP of source and destination, beacon response time, HTTP profiles and traffic segmentation.

IV. ATTACK VECTORS ALGORITHM

Actual adversary emulation starts from the second last phase, where we launch our emulation plan. Some crucial tasks in the execution of a plan include getting initial access or harvesting passwords for getting access to the system. These types of techniques come in social engineering attacks.

The most crucial thing in a successful cyber attack is the ''attack vector''. In this section, we will discuss some complex tasks such as launching phishing attacks, defense evasion and creating an implant that will act as an agent to launch an attack.

A. PHISHING

1) PROCEDURE DETAIL

Business email compromise has increased its roots to carry out spear phishing attacks by adversaries [32]. Changing security paradigm BEC attacks are more reliable so, this algorithm is in the context of ordinary spear-phishing as well as BEC. Compromised email accounts can be found on hack-forums, so these accounts can act as initial input to this process. `email_validate()` is optional in it, but still, we can use it to verify if a company has not changed its domain by validating the email address. Next to this, the payload will be generated as specified in type such as a malicious document. This result will be given to the template generation function to finalize the email template. Once the template is ready, the algorithm will replace header details and send the final mail, initial access algorithm is from our previous work [22].

B. ALGORITHM 1: STATIC ANALYSIS BYPASS

1) PROCEDURE

Algorithm 1 utilizes metasploit implant ''meterpreter'' and uses ''nonstandard encoding'' schemes with different integrated methods to obfuscate function. For the sake of simplicity, the algorithm contains three functions; `SO(), MBX(), RCI(). SO()` is simple obfuscation which has some predefined procedures, those procedures use templates to inject junk code, instructions. This approach is good to bypass signature-based detection, sometimes it works for heuristic approaches as well. `MBX()` encodes the code with a key

Algorithm 1: static analysis bypass

Input: XE (meterpreter shell code in hex or in binary form). C2 (lhost and port details). obfuscation method can be: RCI (randomized code injection frequency). MBX (multibyte xor key). SO (Simple obfuscation)
Output: Compiled meterpreter EXE file
1: **Initialize:** OM = obfuscation method. LHOST , LPORT. n = number of iterations. PM = payload method. XOR_key = key . code_injection_random_key=2
2: msf_payload_generate (reverse_tcp)
3: **if** PM = RCI **then**
4: **result** ← **call** RCI (payload_hex)
5: **else**
6: **if** PM = MBX **then**
7: **result** ← **call** MBX (payload_hex)
8: **else**
9: **if** PT = SO **then**
10: **result** ← **call** SO (payload_bat_file)
11: RCI (payload_hex)
12: **Inject** exec method: Heap_RWX
13: **Inject** remote exec method: THE
14: **call** random_code(payload_hex)
15: **output:** c code
16: **result** ← (**call** comile (c_code))
17: SO (payload_bat)
18: **Inject junk:** SED to replace bytes and add junk
19: **result** ← (**call** compile ← (**call** base64 ()))
20: MBX (payload_hex)
21: **call** encrypt shellcode_hex(hex. MBX_key)
22: **output:** c code
23: **result** ← (**call** comile (c_code))
24: **end**

FIGURE 4. Algorithm 1: Static analysis bypass.

TABLE 5. Algorithm 1: Static analysis bypass.

Functions	Description
msf_ pay- load _gen()	This is the main function, which inputs `msfvenom` payload and then on the basis of "payload method", it calls appropriate function and passes `msfvenom` payload to it to returns the result.
MBX()	This function obfuscates the hex payload with multi-byte XOR against a given key. In this algorithm we have set a default key. This function traverses through each byte and performs encoding.
RCI()	This function injects random bytes at different places in the hex payload on the basis of specified frequency.
SO()	We call this function "standard obfuscation", it uses bat file and junk instructions in file. Before converting it into a .exe file, it adds escape sequence and strings obfuscations in code and then compiles it.
Compile	This function uses gcc, g++ and migw to compile code.

to obfuscate payload. Payload execution and remote code execution method can be specified for each payload or set by default.

Algorithm 2: dynamic analysis bypass

Input: meterpreter in c language
Output: Compiled obfuscated EXE file
1: **Initialize:** p = set of recorded patterns, ep = set of patterns wanted to achieve.
2: **call** debugger (shellcode) {
3: **call** modify_behavior ()
4: **Inject:** junk dynamic functions
5: **Inject:** Inject anti VM execution code
6: **Inject:** In-memory exec method. clean traces
7: **result** ← **call** compile_exe (**call** obfuscate ())
8: **update:** Obfuscation and execution method
9: \\verifying analyzing exe
10: **call** sandbox (compiled_exe)
11: **do**
12: **if** p not equal to ep **then**
13: **print** pattern not achieved
14: **start** from beginning
15: **else**
16: **if** p is equals to ep **then**
17: **print** new behavior achieved
18: **call** parse_headers (compiled_exe)
19: **call** modify ()
20: **Attach:** impersonate certificates
21: **Modify:** header, import table
22: **result** ← final_executable
23: **while** p =ep
24: **end**

FIGURE 5. Algorithm 2: Dynamic analysis bypass.

2) TIME COMPLEXITY

Algorithm 1 has three static inputs and the algorithm contains conditional statements. Algorithm contains no looping structure which makes it time complexity to O(1).

C. ALGORITHM 2: DYNAMIC ANALYSIS BYPASS

1) PROCEDURE

Logic behind this algorithm 2 is to break the sequence of existing malware at different points so that its traceability can be mitigated. Algorithms start with initializing two behaviors, first behavior is the existing malware behavior, other behavior is the one to achieve. The first function `debugger()`, `modify_behaviour()` uses reverse engineering techniques to modify behavior. Then injects different procedures to defeat dynamic algorithms such as anti-vm techniques, in-memory execution techniques. Anti-vm method exploits limitations of virtual machines to defeat dynamic analysis. Update method is a critical procedure, it updates methods at each execution to achieve FUD properties. After this process is done the algorithm verifies the payload for achieved behavior, if it is successful then `modify()` attaches certificates with PE/exe file and slightly modify import tables to achieve high possibility of dynamic evasion.

TABLE 6. Algorithm 2: Dynamic bypass using FUD strategy (Fear, uncertainty, and doubt).

Debugger()	Consists breakpoints at different places in raw c payload and executes them to learn behavior of different functions. This also includes dynamic analysis, which might be achieved by running it inside a sandbox.
modify_ be-haviour()	This function tries to modify the behaviour by modifying sequence of syscalls, kernel32dll and other dll imports. Also try to minimize the use of API's. This process is usually done on assembly code with different XOR action on registers and replacing them in different registers.
Inject junk dynamic functions	This subroutine injects different dynamic functions, such as get date function also Handler requests for sound or display drivers.
Inject anti-vm methods	Anti VM techniques, such as sleep(), pause(). delay() or trying to access or allocate a huge chunk of memory may stop execution of payload inside VM.
Inject in memory execution methods	This subroutine is not for generalized use. It uses custom loaders to load files from memory and executes payload in form of VB scripts "inside memory", we refer to this as in memory evasion.
Update method	Global function for updating techniques. This method might exploit different methods to keep techniques updated.
Obfuscate()	This is the same obfuscate function used in static analysis, which uses string obfuscation, function obfuscation using XOR and other techniques.
Compile()	This function compiles c code using, gcc, g++ and mingw. We considered c code in our algorithm.
Sandbox()	This function analyzes the newly built payload, and compares it; if the newly built payload is different in behavior from previous payload and analyzes the detection ratio.
Parse_ headers() modify()	This function parses the header of PE .exe file and modifies some parameters like compile date, time, platform, etc.
Impersonate certifi-cates	This function uses impersonate certificates of other legitimate applications and signs the executable file with that certificate.
modify() Import tables, PEB	Import tables and play an important role in detection of payload during runtime, when payload imports dlls. This function adds junk dll imports to modify the import table.

2) TIME COMPLEXITY

Algorithm 2 has one static input that shall remain the same for each execution. In actual implementation, input might be malware or different shell code. However, input shall remain 0<n, n=1. Algorithms consist of two phases, first one is modifying behavior and second phase is verifying and validation process. Second phase uses one loop to run the algorithm until it achieves a specific pattern. Second phase has linear complexity. If the number of iterations increases, it shall increase time complexity. It yields linear time complexity of $O(n)$. However, the phase has static input with no looping structure. Section one has constant time complexity of $O(1)$.

D. ALGORITHM 3: LINUX PRIVILEGE ESCALATION

1) PROCEDURE

Defense evasion in Linux is not an issue rather the actual problem is getting root access. Real problem starts after getting the payload executed, as Linux has a different directory structure and execution mechanism as compared to windows. For traversing directories or /root /bin attackers must have to "escalate privileges" to get root access. Aside from zero days or built-in system vulnerabilities in Linux, getting root access in latest build is a tough job. Most renowned method of privilege escalation is to take advantage of misconfigurations and improper management of access rights throughout the directories. Attackers traverse throughout the directories and find "dir" with execution rights or files with read/write access. This is a loophole which the attacker uses. Second renowned method is to use cron jobs or other scheduling jobs to run our payload. This approach also relies on improper access management of rights through the directories.

For effective adversary emulation we have formulated an algorithm for auto privilege escalation based on "sudo, environment variables misconfigurations and kernel exploitation". Algorithm takes local enumeration results as input and on the basis of results, the operator needs to identify which exploitation method is best. POC is available here.[1] This pseudocode is written for bash and works fine with zsh shell as well. Algorithm starts with a control sequence and matches input with available methods, if a method match is found then it tries to exploit it. Below is the algorithm and variable details are present in Table 7.

```
Algorithm 3: Linux Privilege Escalation
Input: EO (Enumeration Output)
Output: Method or exploit for privilege escalation
1: if (EO ==sudo)  {
2:     If (sudo ==EscapeSeq) {
3:             run shell script
echo "os.execute('/bin/sh')" > shell.nse && sudo nmap
script=shell.nse  }
5: else if (sudo ==ExtractHash)  {
6:             run shell
7:             sudo apache2 -f /etc/shadow  }
8: else if (sudo ==Obj_inject)  {
9:             Load exploit
11:            Modify Environment Path  }
12: else  {
13:        Print "no match"
14:    }
15: else if (EO== Cron) {
16:      if (Cron==Bin_Mod)  {
17:             run shell;
18:     echo 'cp /bin/bash /tmp/bash; chmod +s
19: /tmp/bash' >/home/user/overwrite.sh }
20:     else if (Cron==Path_Mod) {
21:             Replace cron path }
22: end
```

FIGURE 6. Algorithm 3: Linux privilege escalation.

[1] https://github.com/basit10/linux_privilege_escalation

TABLE 7. Algorithm 3: Privilege escalation.

Sudo / setuid	If we have found sudo related issues on the machine then, sudo related methods of privilege escalation are present in "else if block after sudo".
EscapeSec	This Variable decides if escape sequences need to be tried on the basis of sudo== EscapeSeqes.
ExtractHash	This comes in an exploitation method which can be used to extract hashes from shadow files on the basis of sudo ==ExtractHash and enumeration output.
Subroutines: Inject anti-vm methods	Anti VM techniques, such as sleep(), pause(). delay() or trying to access or allocate a huge chunk of memory may stop execution of payload inside VM.
Obj_inject	From enumeration result if we found the existence of obj injection then this variable will decide if we need to try this (sudo == obj_inject).
Cron	If enumeration results give way forward for cron based exploitation for privilege escalation then this variable will decide if we need to get in cron if else block.

2) TIME COMPLEXITY

Algorithm 4, has one input parameter which is always static in nature and the algorithm contains conditional statements to perform two-way decisioning on the basis of condition. Algorithm contains no looping structure which makes it time complexity to O(1).

V. EXPERIMENT

We evaluate our approach through experiment in the presence of "Bitdefender Total Security", "Trend Micro Deep Security", "Advanced Netsh Firewall", "Windows Defender", "SNORT", "Symantec Endpoint Security", "Fortinet EDR" and "Blue SPAWN (open source EDR)". Such similar security tools are installed at endpoints in organizations. We demonstrated how an actual adversary will behave in the presence of security mechanisms and evade them to achieve their malicious goals. This will evaluate real time security of an organization against adversary.

1) EXPERIMENTATION DETAILS ON GITHUB

We have repository on github[2] as well related to adversary emulation, containing all logs, endpoint data files, ATT&CK Evaluations, scripts etc.

Organizations need to conduct adversary emulation exercises periodically, to make their security measures more robust against emerging threats. We are assuming a test environment as a front desk system in a small size organization, usually front desk systems are prone to attacks. We conduct attacks against the system to evaluate their counter performance against actual adversaries.

[2]TBAE-Approach, https://github.com/basit10/TBAE-Threat-Based-Adversary-Emulation

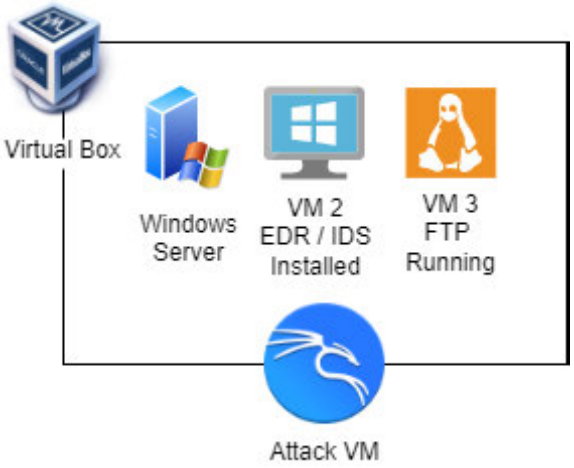

FIGURE 7. Target environment.

A. SCENARIO

In order to test the effectiveness, a test environment need to be set up to simulate the new team structure. Aim of this scenario is to follow the presented approach and algorithms to gain access to endpoint. This is just a test scenario so utilizing spear-phishing and BEC is not feasible so, we are considering an initial breach in this experiment.

1) TARGET ENVIRONMENT

Environment consists of two systems on virtualbox, which encapsulates them from the actual system. We are assuming that two systems are actual systems. System 1 is server. Which is a centralized system responsible for controlling systems on domain. Local exchange server is also running on it. System 2 is a front desk system with built-in security mechanism. Moreover, IDS is also installed on the system. Systems 3 is a linux machine running a FTP server.

2) ATTACK SCENARIO

We used an independent payload for different stages. First payload delivered via email which performs pre-adversary emulation reconnaissance tasks and tries to download actual payload on the system, Different strategies for such attacks are explained here [33], [34] [35]. Pre-emulation tasks include fetching information about security mechanisms installed at target.

3) OBJECTIVES

Objective for this attack scenario inside the target environment is to launch different attacks generated from our algorithms and emulate adversaries from the proposed approach. We are following APT28 techniques for this experiment, with modified procedures to evade emerging security mechanisms. Moreover, this will evaluate the security deployed at the target.

4) SCOPE

In this experiment, we have covered seven abstract categories of attacks which are usually known as "Tactics".

TABLE 8. Considered attack vectors for initial access.

Method	Purpose
Browser exploitation	foothold
Spear phishing	initial access
Password spraying	initial access
Hidden redirect Link	lure

Initial access, Execution, persistence, privilege escalation, defense evasion, credential access, command and control and ex-filtration. For each tactic, we used one technique with different methods to emulate emerging threats.

Systems under test are discussed earlier in this section. Initial access, and getting an initial foothold on the target system is the goal of this experiment. We will follow adversary techniques and use them to launch further attacks. After successful execution of the phishing payload, It will gather host details with security mechanisms installed and then invoke CMD to download the actual payload, Actual payload is python-based ''RAT''. Phishing in organizations is explained here [36]. **Initial and final payload**, Initially we used OLE files (docx/doc) with embedded VBA code to deliver a modified meterpreter staged payload. The process for generating this payload is pretty simple. We followed msfvenom to generate raw payload into bat file and used it to build a valid VBA macro using macroshop.py (github project). Execution starts from auto_exec() in VBA code which executes the code. The second method we utilized is password spraying on exchange accounts using different scripts. After successful execution, it will download the actual payload using scripted web delivery from hta and executes RAT. The third plan is to use online clipboards to download payload hex or use IPv6 DNS requests to fetch payload HEX.

TABLE 9. OS & tools used.

Defense	purpose
Kali Linux	Base OS
Custom C2	Give commands to RAT
Custom Build RAT	State of the art payload
Non-Standard Enc	Bypass static analysis
Metasploit	detect and prevent
Headless HTTP server	re-directors

5) ADVERSARY EMULATION PLAN

At this stage, we have defined our objective, goals, and scope. Gathering intelligence and new TTPs is a critical task. After researching different threats and mapping their techniques, and methods on MITRE ATT&CK. We have a clear view of adversary activities. In Figure 8, is our developed flow chart ''Experiment Flow Chart'', Which will test threats associated with adversary.

Flow starts from generating a phishing payload which is inspired by an adversary, which aims to help in getting initial access on the target system with pre-emulation tasks such as security tools installed at target. After successful execution it will invoke ''cmd'' to download the actual payload. Here we used different staged payload limits. Each payload is limited to its purpose. Once RAT [37] is downloaded, by using trusted binaries it will load itself in memory and remove all kernel callbacks associated with RAT to avoid EDR detection. Using system calls it will inject shellcode into the system. RAT has integrated SSL/TLS certificate which will initialize egress encrypted connection requests to C2, endpoint security will ignore this and consider it as a trusted connection. After establishing a successful connection with C2, RAT will put itself in non-interactive mode. Which is a very slow mode of communication with delayed call backs to C2. If C2 instructs to load a ransomware module, RAT will use a custom encrypt loader to encrypt a few specified files on target. that files can be specified on the basis of ''type''. We used ''jpg'' as the default type. Ransomware modules will be loaded from host to target and RAT will extend its ability with cryptography features.

Report will be generated for execution of the process and sent back to C2, and again RAT will put itself in non-interactive mode.

6) BUILDING TOOLS AND EXECUTION PHASE

We build a set of new methods for performing the same tasks used by adversaries. For Example, non standard encoding sachems and in-memory execution, executing shell code using trusted windows binaries and using system calls to inject payload. Building tools phase starts with documenting requirements for adversary emulation; in our experiment we need custom cryptography library, python wrappers, phishing server, Redirect servers, SSL certificates and a base tool which supports interaction with target. For the experiment, we have used our custom build tools.

Execution, Initially we generated phishing payload using algorithm 1, as discussed in earlier sections. Summary of phishing mail is below:

1) Wrapped malicious links to our planted websites using: short_url · PyPI
2) PDF document with segmented payload in body section and re-segment code in file header
3) Situational awareness module integrated in payload that will fetch details about host security measures

and share those details in the form of email or through https connection, and stay in non-interactive mode after specified time payload will download RAT and execute it using LOLB. RAT is python based and compiled with wine libraries and converted to windows exe using an algorithm discussed in earlier sections.

Credentials Access, Phishing mail sent to the target had an embedded link that generates a local DNS request to a nonreachable address (to attcker), our planted server on the network served that DNS request using LLMNR [38]. And forced the initiator to authenticate with a challenge and share a password hash. At this point, we use an open-source tool to crack the dumped hash.

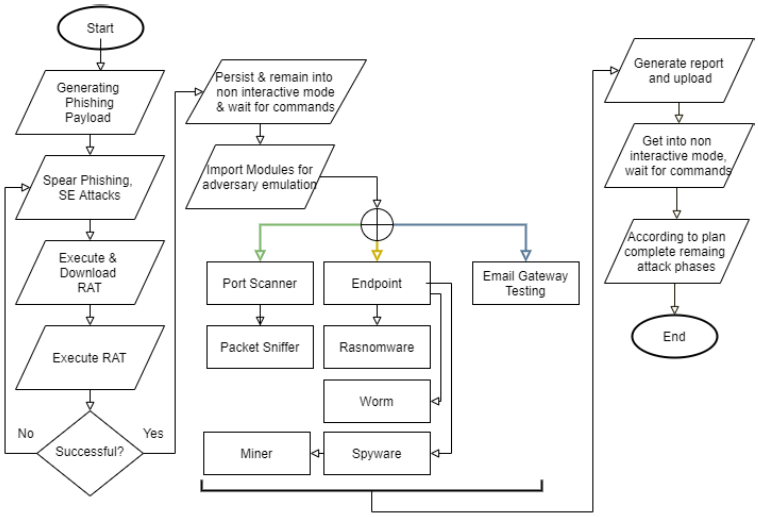

FIGURE 8. Experiment flowchart.

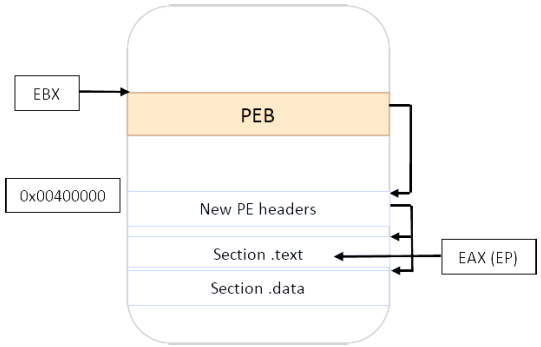

FIGURE 9. PE execution.

Use of living of the land binaries and non standard encoding schemes made it possible to bypass security mechanisms.

Technique used to evade at endpoint, we kept each module separate and independent from the actual RAT payload to avoid traceability and detection. execution of RAT is a critical process to whole adversary emulation. Right now we have employed T1055.012, T1055.08, T1055.04, T1055.09, T1055.014. In this experiment, we are focusing on **T1055.012**. We have modified the process hollowing technique with hybrid graded launch method to make sure endpoint security does not detect it. At the very first trusted binary call, RAT will list itself in PEB (Process Environment Block) and suspend itself, once a trusted binary is executed it replaces itself with the executed process by taking advantage of already stored data in PEB. The below image shows ''process hollowing'' in pictorial form. More details related to PE file use for adversarial activities are here [39], [40]. In portable Executable Relocations, we need to allocate a block of memory of size SizeOfImage in the destination process that will be our new ImageBaseAddress of the source image. Ideally, we would allocate new memory at ImageBaseAddress of the destination image.

Disabling tool as listed in ATT&CK T1562, sub-technique as T1562.001 is used in experiment to minimize kernel call backs to bypass EDR monitoring.

PE files contain a structure called Import Address Table. IAT contains pointers to information that is critical for an executable to do its job which contains a list of DLLs it depends on for providing the expected functionality. A list of function names and their addresses from those DLLs that may be called by the binary at some point. It is possible to hook function pointers specified in the IAT by overwriting the target function's address with a rogue function address and optionally to execute the originally intended function. This technique is known as ''Import Address Table Hooking''. We have integrated different techniques in RAT for successful execution. This process includes the use of living of the land binaries which are trusted windows binaries. **Ransomware Testing**: Used crypto.cipher import AES to encrypt files. Rather than performing complex methods to replace file addresses and encrypt them, then update file addresses. For this experiment, we just encrypted jpg files as ''file creation process'' to avoid any behavioral detection and then deleting the original files. Details of ransomware are here [41].

Email Gateway Testing: launched different phishing emails with hidden links, and malicious documents and was able to penetrate spam filters. **Report and feedback,** Report sent to C2 describing the execution of processes with successful and unsuccessful processes, summarizing the whole adversary emulation exercise. After this RAT puts itself in non-interactive mode to avoid any detection by smart firewalls. Results are discussed in the section below.

B. WINDOWS PRIVILEGE ESCALATION & LATERAL MOVEMENT

At this stage we must implement known approaches for privilege escalation and lateral movement with sys calls or ring0

calls, which will bypass user mode and mitigate user mode detection. For privilege escalation we are considering access token manipulation, using BOF (beacon object file) to execute our compiled object as a position independent code. Such files can execute in a running process and we can write ring0 calls as well in this. Header file is available here[3] writing a C code which declares to handle and write a prototype for LogonUserA() using current access token and manipulating it. Mingw is suitable for compiling for such CPP files into object files. For lateral movement we are considering DCOM exploitation, the process starts from declaring, and initializing COM. Following the unique CLSID for MMC20.Application class. Next, we need to find an interface that declares the specific method we want to invoke (for example ApplicationIfc & CoCreateInstance). Now we must create MMC20.App type object and getting a pointer to that function. Finally, we are ready to get an ID to invoke specified functions. Now we can invoke it and use it in our malware to move laterally.

C. EVADING FIREWALL FOR POSITIVE C2

For C2 and firewall evasion we are focussing on T1090 (sub-technique T1090.004), also known as ''Domain Fronting''. Mitigation technique for domain fronting is SSL/TLS Inspection [42] which is not widely deployed and applicable in different scenarios. Moreover, it's not very effective in mitigating it.

If it is possible to inspect HTTPS traffic, the captures can be analyzed for connections that appear to be domain-fronting. In a domain-fronted HTTPS request, one domain appears on the ''outside'' of an HTTPS request in plain text-in the DNS request and SNI extention-which will be what the client wants to pretend they are targeting in the connection establishment and is the one that is visible to censors, while a different domain appears on the ''inside''-in the HTTP Host header, invisible to the censor under HTTPS encryption-which would be the actual target of the connection.

Proxy methods [43] for C2 having sub techniques as T1090.001, T1090.002, T1090.003, T1090.004 are most reliable ways for positive C2. For approach details regarding domain fronting [22].

D. EVALUATION & RESULTS

ATT&Ck coverage of security mechanisms deployed at target which includes all known techniques covered by ''BlueSpawn EDR'', other EDR includes Fortinet EDR but they have not disclosed Mitre ATT&CK coverage. and we were able to bypass all of them. Defense Evasion ATT&CK Techniques include T1553, T1055.001.

Credential Access Man-in-the-Middle: LLMNR/NBT-NS Poisoning and SMB Relay, T1557.001. The above chart shows the results for our overall adversary emulation exercise with the division of attacks that are successful, reported, neutralized, and undetected. Of overall attacks at the endpoint only 78% of attacks were able to make it through, 10% were reported by EDR as warning or displaying suspicious activity detected but unable to identify it, this where EDR super-pass AV. 7% of attacks were detected in very initial phases. 5% of attacks were completely neutralized. Table 11 shows attack detection by security vendors. Mitre ATT&CK evaluations are available on github project.[4]

TABLE 10. Mitigation of ATT&CK on target.

ID	Description
T1055.002	Portable Executable Injection
T1055.003	Thread Execution Hijacking
T1055.004	Asynchronous Procedure Call
T1055.005	Thread Local Storage
T1055.008	Ptrace System Calls
T1055.009	Proc Memory
T1055.011	Extra Window Memory Injection
T1055.012	Process Hollowing
T1055.013	VDSO Hijacking
T1562.001	Disable or Modify Tools
T1562.002	Disable Windows Event Logging
T1562.003	Disable or Modify System Firewall
T1562.006	Indicator Blocking
T1562.007	Disable or Modify Cloud Firewall

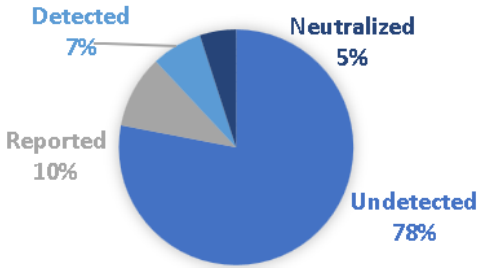

FIGURE 10. Adversary emulation outcomes.

TABLE 11. Attack detection by security vendors.

Security Vendor	Detection Status
Windows defender	No
Fortinet EDR	No
Symantec Endpoint Security	No
Windows Defender Cloud	No
Snort/Suricata	No
Trend Micro Deep Security	No
Windows ntsh Firewall	No
Kaspersky	Partially

E. IMPACT ANALYSIS

1) PHISHING

Mail headers possess a lot of information about their origin server. Analyzing email headers yields the use of blind mail with SMTP server IP address and location. Mail was encrypted with TLS during transit. The server we used was hosted on AWS, and initially mail landed in the inbox. This was the basic form of phishing mail we used. We advanced it and attached a payload (exe file) with custom UTF encoding attached during transit. This time mail landed in spam and

[3]https://www.cobaltstrike.com/downloads/beacon.h

[4]https://github.com/basit10/TBAE-Threat-Based-Adversary-Emulation

was detected as suspicious because of the presence of a suspicious file type. Next mail was launched from a mock company DNS and SMTP server with SPF, DKMI and DMARC enabled. This time the payload was a PDF file with embedded staged payload, which was carefully attached in fragments at different places. This time the mail landed in my gmail inbox.

2) PE PAYLOADS

We tested cobalt strike and meterpreter shell code in raw (with slight modifications such as multi byte XOR) on Windows 10 2004 and antiscan.me, after such modification file hash becomes different and this shell code was able to evade static analysis but got detected in heuristic. Advancing this approach, we tested shell code based on socket programming in python as a .py file and it was able to bypass all sorts of static analysis on Windows without any encoding or encryption. Moreover, we tested payloads with more complexities in them, which include heavy obfuscation, strings, and variable names encryption with AES. Using custom loaders (for in memory execution) embedded in payloads to load payload configuration and other modules for further testing, such payloads are FUD (fully undetectable). Furthermore, we used win32 API splitting method as ''divide & conquer approach'' to trick behavioral-based detection. Idea behind this approach is to divide the main process in sub-processes and call API's in different processes.

3) EXECUTION

Memory injections that use reflective loading mechanisms can be detected by analyzing entry points. Same way threat hunters can detect the presence of suspicious processes such as the T1553 technique. In such a scenario we can delay the detection and buy more time. The following are indicators that can indicate suspicious behavior.

- The name of PE file
- Access rights being used to access specific process

For specific techniques implementation with known methods plays an important role in detection such as strings inside PE files, and known hashes. Such types of characteristics are simple to modify with less effort. For the second type, Process execution hierarchy or which process initiated another process with what rights are the important indicators for detection. Techniques T1553 do not require full access rights, basic implementation of T1553 requires 2312 rights in base 10 value. Table 12 explains mitigation of malicious indicators for MSF shellcode.

For evasion, we used renaming of file with some trusted windows binary name like msmpeng.exe. This can mislead security mechanisms with cloned hashes, certificates for evasion. For the second type, as we mentioned earlier, rights required by T1553 are 2312, if we integrate different rights to achieve similar access. Dividing the whole process of T1553 in parts, one for execution, suspend, start and alteration of memory with different handles for each process for evasion.

TABLE 12. Malicious indicators and their mitigation.

Indicator	Remarks	Mitigation
MSF shellcode	Malicious opcodes sequence	Use Encoding
Virtuall-Alloc or EX	There are several rules related to VirtualAlloc() with custom given PID	allocate methods like HeapAlloc()
Win32 Imports	Kernel32/ntdll	Dynamic API resolution

TABLE 13. Linux privilege escalation outcome.

Distro	Daemons/Kernel	Sudo	cronjob & Misconfig
Red Hat (old)	yes	yes	yes
Red Hat (latest)	no	yes	yes
Debian Hardened	no	no	yes

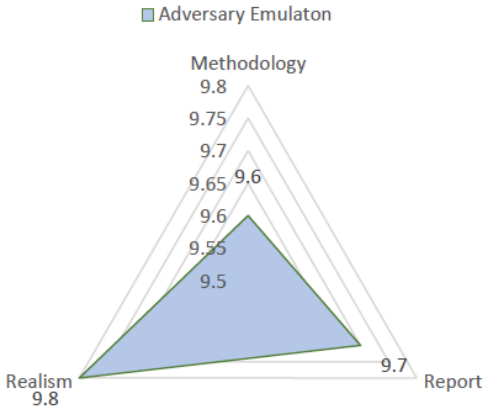

FIGURE 11. Approach evaluation metric.

TABLE 14. Metric evaluation table.

Question	Answer
Did you avoid detection?	yes
Performed latest TTP's?	Yes
Emulate inhouse build malware as well?	Yes
Extracted new TTP's and replicated them?	Yes
Used social engineering?	Yes
Performed exfiltration	Yes
Emulate malware?	Yes
Performed black box testing?	Yes

4) LINUX PRIVILEGE ESCALATION

For Linux privilege escalation, we evaluated it on different red hat and debian based machines (including stock and hardened system). Following is the Table 13 showing algorithm outcomes on Linux machines.

VI. APPROACH EVALUATION

The main problem that comes into play while doing adversary emulation is to qualify the quality of emulation. Here we are using an evaluation model based on three factors which are: realism, methodology, report. For each evaluation factor we have a set of questions that we need to answer to mark them on radar to get final results. The following is the model and we have achieved a high ''realism factor'' by considering all sort

of defensive mechanisms while emulating. Table 14, we have answered questions regarding ''Realism'' factor evaluation.

VII. CONCLUSION

For security infrastructure and controls evaluation, we need the same type of methodologies and strategies that are used by a real-world adversary to evaluate security controls with more emphasis on unknown threats. In this paper, we present an approach using a unique set of activities to build an adversary emulation plan, gather intelligence, and pre-adversary emulation activities and analyze techniques and executing plans. Our presented approach focuses on simulating real-world attacks with being limited by a specific threat or adversary. It allows organizations to test their defenses against a wide range of potential threats and better understand their overall security posture, as well as to stay ahead of evolving threats by continuously testing and updating their defenses. We induced some actions from the MITRE APT plan. We present three algorithms for generating payloads that are capable of bypassing security mechanisms usually installed by organizations. We tested our approach with a full-scale experiment and showed how efficient this method is to evaluate one's security against actual APT threats. In the future, we would like to build a modular automated system to rapidly switch defense evasion modules and other attack modules.

REFERENCES

[1] S. Herbert-Lowe, ''8 reasons why business email compromise is a risk for trustees,'' *Australas. Law Manag. J.*, pp. 1–3, Mar. 2022. [Online]. Available: https://search.informit.org/doi/10.3316/informit.20220602067914

[2] B. Canner. (2020). *Here is Why Endpoint Security is Important to Your Enterprise*. [Online]. Available: https://solutionsreview.com/endpointsecurityhereiswhyendpointsecurityisimportantforyourenterprise

[3] M. Denis, C. Zena, and T. Hayajneh, ''Penetration testing: Concepts, attack methods, and defense strategies,'' in *Proc. IEEE Long Island Syst., Appl. Technol. Conf. (LISAT)*, Apr. 2016, pp. 1–6.

[4] H. M. Z. A. Shebli and B. D. Beheshti, ''A study on penetration testing process and tools,'' in *Proc. IEEE Long Island Syst., Appl. Technol. Conf. (LISAT)*, May 2018, pp. 1–7.

[5] D. Olifer, N. Goranin, A. Kaceniauskas, and A. Cenys, ''Controls-based approach for evaluation of information security standards implementation costs,'' *Technol. Econ. Develop. Economy*, vol. 23, no. 1, pp. 196–219, 2017.

[6] X. Cai, K. Shi, K. She, S. Zhong, Y. C. Soh, and Y. Yu, ''Performance error estimation and elastic integral event triggering mechanism design for T–S fuzzy networked control system under DoS attacks,'' *IEEE Trans. Fuzzy Syst.*, vol. 31, no. 4, pp. 1327–1339, Apr. 2023.

[7] X. Cai, K. Shi, K. She, S. Zhong, and Y. Tang, ''Quantized sampled-data control tactic for T–S fuzzy NCS under stochastic cyber-attacks and its application to truck-trailer system,'' *IEEE Trans. Veh. Technol.*, vol. 71, no. 7, pp. 7023–7032, Jul. 2022.

[8] P. S. Shinde and S. B. Ardhapurkar, ''Cyber security analysis using vulnerability assessment and penetration testing,'' in *Proc. World Conf. Futuristic Trends Res. Innov. Social Welfare (Startup Conclave)*, Feb. 2016, pp. 1–5.

[9] S. Shah and B. M. Mehtre, ''A modern approach to cyber security analysis using vulnerability assessment and penetration testing,'' *Int. J. Electron. Commun. Comput. Eng.*, vol. 4, no. 6, pp. 47–52, 2013.

[10] S. Shah and B. M. Mehtre, ''A reliable strategy for proactive self-defence in cyber space using VAPT tools and techniques,'' in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res.*, Dec. 2013, pp. 1–6.

[11] I. L. Aller, J. M. R. Lopez, and L. A. V. Martinez, ''Towards lightweight mobile pentesting tools to quickly assess machine security levels,'' *IEEE Latin Amer. Trans.*, vol. 17, no. 7, pp. 1116–1123, Jul. 2019.

[12] G. Weidman, *Penetration Testing: A Hands-On Introduction to Hacking*. San Francisco, CA, USA: No Starch Press, 2014.

[13] B. Clark, *RTFM: Red Team Field Manual*, J. Vest, Ed. Createspace Independent Publishing Platform, Feb. 2014.

[14] T. Wrightson, *Advanced Persistent Threat Hacking: The Art and Science of Hacking Any Organization*. New York, NY, USA: McGraw-Hill, 2014.

[15] N. T. Pages, ''Module development in metasploit for pentesting,'' M.S. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 2019. [Online]. Available: https://upcommons.upc.edu/bitstream/handle/2117/171278/Module%20development%20in%20Metasploit%20for%20pentesting.pdf

[16] N. Moran. (Nov. 2014). *Operation Double Tap*. [Online]. Available: https://www.fireeye.com/blog/threatresearch/2014/11/operation_doubletap.html

[17] N. A. Almubairik and G. Wills, ''Automated penetration testing based on a threat model,'' in *Proc. 11th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2016, pp. 413–414.

[18] A. Singh, N. Jaswal, M. Agarwal, and D. Teixeira, *Metasploit Penetration Testing Cookbook: Evade Antiviruses, Bypass Firewalls, and Exploit Complex Environments With the Most Widely Used Penetration Testing Framework*. Birmingham, U.K.: Packt, 2018.

[19] D. Miller, R. Alford, A. Applebaum, H. Foster, C. Little, and B. Strom, ''Automated adversary emulation: A case for planning and acting with unknowns,'' MITRE, McLean, VA, USA, Tech. Rep. 18-0944-1, 2018. [Online]. Available: https://www.mitre.org/sites/default/files/2021-11/prs-18-0944-1-automated-adversary-emulation-planning-acting.pdf

[20] A. Belfadel, M. Boyer, J. Letailleur, Y. Petiot, and R. Yaich, ''Towards a security impact analysis framework: A risk-based and MITRE attack approach,'' in *Computer Security. ESORICS 2022 International Workshops: CyberICPS 2022, SECPRE 2022, SPOSE 2022, CPS4CIP 2022, CDT&SECOMANE 2022, EIS 2022, and SecAssure 2022, Copenhagen, Denmark, September 26–30, 2022, Revised Selected Papers*. Copenhagen, Denmark: Springer, 2023, pp. 212–227. [Online]. Available: https://www.springerprofessional.de/en/towards-a-security-impact-analysis-framework-a-risk-based-and-mi/24040982

[21] K. Kotis, S. Stavrinos, and C. Kalloniatis, ''Review on semantic modeling and simulation of cybersecurity and interoperability on the Internet of Underwater Things,'' *Future Internet*, vol. 15, no. 1, p. 11, Dec. 2022.

[22] A. B. Ajmal, M. A. Shah, C. Maple, M. N. Asghar, and S. U. Islam, ''Offensive security: Towards proactive threat hunting via adversary emulation,'' *IEEE Access*, vol. 9, pp. 126023–126033, 2021.

[23] M. Mbow, K. Sakurai, and H. Koide, ''Advances in adversarial attacks and defenses in intrusion detection system: A survey,'' in *Science of Cyber Security-SciSec 2022 Workshops: AI-CryptoSec, TA-BC-NFT, and MathSci-Qsafe 2022, Matsue, Japan, August 10–12, 2022, Revised Selected Papers*. Matsue, Japan: Springer, 2023, pp. 196–212. [Online]. Available: https://kyushu-u.elsevierpure.com/en/publications/advances-inadversarial-attacks-anddefenses-inintrusion-detection-

[24] M. Alowaidi, S. K. Sharma, A. Al Enizi, and S. Bhardwaj, ''Integrating artificial intelligence in cyber security for cyber-physical systems,'' *Electron. Res. Arch.*, vol. 31, no. 4, pp. 1876–1896, 2023.

[25] A. B. Ajmal, A. Anjum, A. Anjum, and M. A. Khan, ''Novel approach for concealing penetration testing payloads using data privacy obfuscation techniques,'' in *Proc. IEEE 18th Int. Conf. Smart Communities, Improving Quality Life Using ICT, IoT AI (HONET)*, Oct. 2021, pp. 44–49.

[26] *Adversary Emulation Plans*. Accessed: Feb. 2022. [Online]. Available: https://attack.mitre.org/resources/adversaryemulationplans

[27] C. Nguyen, C. Morgan, and S. Mittal, ''Poster CTI4AI: Threat intelligence generation and sharing after red teaming AI models,'' in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2022, pp. 3431–3433.

[28] C. Leite, J. den Hartog, D. R. dos Santos, and E. Costante, ''Actionable cyber threat intelligence for automated incident response,'' in *Proc. 27th Nordic Conf. Secure IT Syst. (NordSec)*, Reykjavic, Iceland. Cham, Switzerland: Springer, 2023, pp. 368–385.

[29] C. Yucel, I. Chalkias, D. Mallis, E. Karagiannis, D. Cetinkaya, and V. Katos, ''On the assessment of completeness and timeliness of actionable cyber threat intelligence artefacts,'' in *Proc. 10th Int. Conf. Multimedia Commun., Services Secur. (MCSS)*, Kraków, Poland. Cham, Switzerland: Springer, Oct. 2020, pp. 51–66.

[30] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and B. C. Thomas, ''Mitre att&ck: Design and philosophy,'' MITRE Corp., Tech. Rep. 19-01075-28, 2018, pp. 1–46. [Online]. Available: https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf

[31] C. A. Korban, D. P. Miller, A. Pennington, and C. B. Thomas, ''APT3 adversary emulation plan,'' MITRE Corp., Tech. Rep. 17-3569, 2017. [Online]. Available: https://attack.mitre.org/docs/APT3_Adversary_Emulation_Plan.pdf

[32] N. S. Al-Musib, F. M. Al-Serhani, M. Humayun, and N. Z. Jhanjhi, "Business email compromise (BEC) attacks," *Proc. Mater. Today*, vol. 81, pp. 497–503, 2023, doi: 10.1016/j.matpr.2021.03.647.
[33] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, "Advanced social engineering attacks," *J. Inf. Secur. Appl.*, vol. 22, pp. 113–122, Jun. 2015.
[34] F. Salahdine and N. Kaabouch, "Social engineering attacks: A survey," *Future Internet*, vol. 11, no. 4, p. 89, Apr. 2019.
[35] S. Gupta, A. Singhal, and A. Kapoor, "A literature survey on social engineering attacks: Phishing attack," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, Apr. 2016, pp. 537–540.
[36] J.-W. Bullee, L. Montoya, M. Junger, and P. Hartel, "Spear phishing in organisations explained," *Inf. Comput. Secur.*, vol. 25, no. 5, pp. 593–613, Nov. 2017.
[37] I. Kara and M. Aydos, "The ghost in the system: Technical analysis of remote access trojan," *Int. J. Inf. Technol. Secur.*, vol. 11, no. 1, pp. 73–84, 2019.
[38] H. Raval, "Getting credentials without exploiting the target," *Digit. Forensics (4N6)*, vol. 5, no. 1, pp. 47–54, 2020.
[39] Y. Kucuk and G. Yan, "Deceiving portable executable malware classifiers into targeted misclassification with practical adversarial examples," in *Proc. 10th ACM Conf. Data Appl. Secur. Privacy*, Mar. 2020, pp. 341–352.
[40] E. Arul, "Hypervisor injection attack using X-cross API calls (HI-API attack)," *J. Ambient Intell. Humanized Comput.*, vol. 12, pp. 1–7, May 2020.
[41] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, "Classification of ransomware families with machine learning based on N-gram of opcodes," *Future Gener. Comput. Syst.*, vol. 90, pp. 211–221, Jan. 2019.
[42] P. Isaev, I. Sayag, A. Volodin, and T. Zegman, "Method for performing TLS/SSL inspection based on verified subject name," U.S. Patent 16/226 661, Jun. 25, 2020.
[43] H. Majed, H. N. Noura, and A. Chehab, "Overview of digital forensics and anti-forensics techniques," in *Proc. 8th Int. Symp. Digit. Forensics Secur. (ISDFS)*, Jun. 2020, pp. 1–5.

ABDUL BASIT AJMAL received the master's degree in information security from COMSATS University Islamabad, Pakistan, in 2021. His current research interests include cyber defense in private and public clouds, application security, and risk assessment.

SHAWAL KHAN received the bachelor's degree in computer science from Shaheed Benazir Bhutto University, Upper Dir, Khyber Pakhtunkhwa, Pakistan. He is currently pursuing the master's degree in information security with COMSATS University Islamabad, Islamabad, Pakistan. His research interests include access control, cryptography, and network security.

MASOOM ALAM received the Ph.D. degree in computer sciences from the University of Innsbruck, Austria. He is currently an Associate Professor with the Department of Computer Science, COMSATS University Islamabad, Islamabad, Pakistan. His research interests include access control systems, model-driven architecture, and workflow management systems.

ABOLFAZL MEHBODNIYA (Senior Member, IEEE) received the Ph.D. degree from the INRS-EMT, University of Quebec, Montreal, Canada, in 2010. He is currently an Associate Professor and the Head of the Department of ECE, Kuwait College of Science and Technology (KCST). Before coming to KCST, he was a Marie-Curie Senior Research Fellow with University College Dublin, Ireland, and before that, he was an Assistant Professor with Tohoku University, Japan, and as a Research Scientist with the Advanced Telecommunication Research (ATR) Institute International, Kyoto, Japan. His research interests include communications engineering, the IoT, and artificial intelligence in wireless networks and real-world applications. He is a Senior Member of IEICE. He was a recipient of numerous awards, including the JSPS Young Faculty Startup Grant, the KDDI Foundation Grant, the Japan Radio Communications Society (RCS) Active Researcher Award, the European Commission Marie Skodowska-Curie Fellowship, and the NSERC Visiting Fellowships in Canadian Government Laboratories.

JULIAN WEBBER (Senior Member, IEEE) received the M.Eng. and Ph.D. degrees from the University of Bristol, U.K., in 1996 and 2004, respectively. He was with Texas Instruments Europe, from September 1996 to October 1998. He was a Research Fellow with the University of Bristol, from November 2001 to August 2007, and with Hokkaido University, Japan, from September 2007 to March 2012. He was a Research Scientist with Wave Engineering Laboratories, ATR Institute International, Japan, from April 2012 to March 2018. He has been an Assistant Professor with Osaka University, since April 2018, and a Guest Research Scientist with ATR Institute International. He is currently an Associate Professor with the Kuwait College of Science and Technology (KCST).

ABDUL WAHEED received the master's and Ph.D. degrees in computer science from Hazara University Mansehra, in 2014 and 2021, respectively. During his doctoral studies, he conducted his research with NetLab-INMC under the School of Electrical and Computer Engineering (ECE), Seoul National University (SNU), South Korea. He completed his Ph.D. research with SNU, under the HEC Research Program, in 2019. He is currently a member of the Crypto-Net Research Group, Hazara University. He is also an Assistant Professor and the Head of the Department of Computer Science, Women University Swabi, Khyber Pakhtunkhwa. Additionally, he performs the duties of being in charge of the IT Section, Women University Swabi. Before this, he held the position of Assistant Professor with the Department of Computer Science and the Dean of the Faculty of Engineering and Information Technology (FEIT), Northern University, Nowshera, Khyber Pakhtunkhwa. He is a highly accomplished Computer Scientist. His research interests include information security, secure and smart cryptography, heterogeneous communications within the IoT, mobile adhoc networks (MANETs), wireless sensor networks (WSNs) security, and fuzzy logic-based decision-making theory. He has authored numerous publications in journals and international conferences, establishing himself as a respected and accomplished researcher in his field.