



AUGUST 6-7, 2025
MANDALAY BAY / LAS VEGAS

From Packet to Process: Disrupting DNS C2 and Tunneling with AI and eBPF in the Linux Kernel at Cloud Scale

Speaker: Vedang Parasnis (Synarcs)

\$whoami



Vedang Parasnis

**Independent Graduate Researcher,
Master's Graduate @University Of Washington**

Email: vedang.parasnis@outlook.com

Research Interests:

**Kernel security, hardening, eBPF, cloud and
system security**

Agenda

- ☐ **DNS a critical backdoor for enterprise networks**
- ☐ **DNS Exfiltration Attack Vectors**
- ☐ **DNS C2 Attack Infrastructure**
- ☐ **Existing Approaches and Challenges**
- ☐ **AI-Driven Linux Kernel Enforced Endpoint Security**
- ☐ **Cloud Deployment Architecture at scale to combat DNS C2 infrastructures**
- ☐ **Demo (disrupt Sliver, DNSCat2)**
- ☐ **Key Takeaways & Future Directions**
- ☐ **Q&A**

They Breach Through DNS — Almost Every Time

Compromise Supply Chain:

- APT29 (Cozy Bear) — SolarWinds

Breach Cloud & Hyperscalers:

- UNC2452 (APT29)

Damage Critical Infrastructure:

- Volt Typhoon

Harvest Credentials at Scale:

- APT28 (GRU), Sea Turtle

Exploit Shared Offensive Tools:

- APT41, FIN7

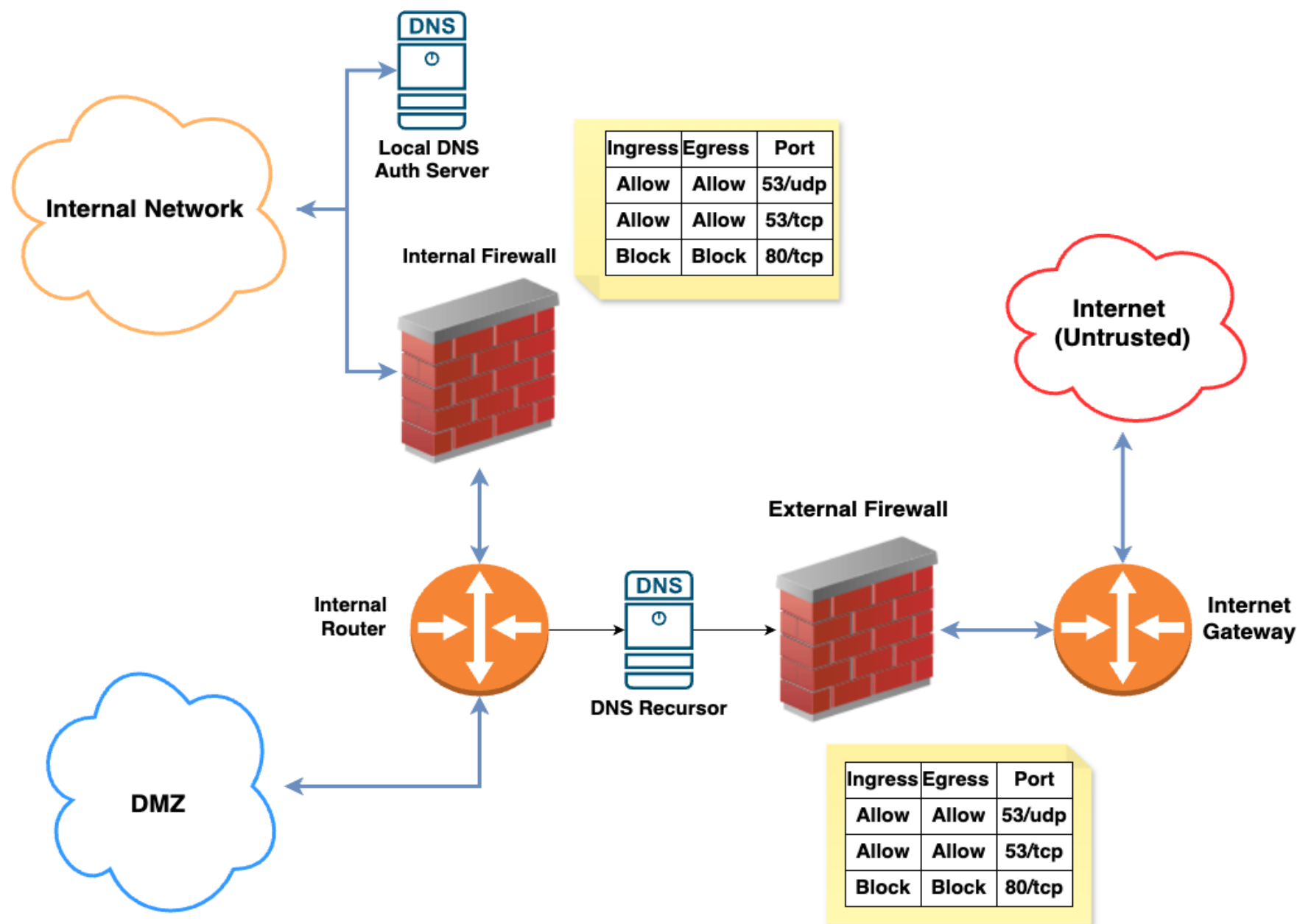
DNS-Based C2 and Tunneling Attacks Timeline



85%+ of APT's employ DNS for C2 and data breaches

DNS a Blind spot to compromise networks

- Unencrypted by Default
- Logs Rarely Monitored
- Firewall Blindspot
- Stateless Protocol



DNS: Not Just For Name Resolution Anymore. Next channel deliver zero-day attacks.

- ❑ **DNS C2** – Uses DNS to embed commands, data in queries and responses to maintain covert communication with remote C2 attacker infrastructure.
- ❑ **DNS Tunneling** – Encapsulates arbitrary data, other protocols within DNS packets to bypass network restrictions.
- ❑ **DNS Raw Exfiltration** – Leaks sensitive data files directly in DNS queries.

RCE & Shellcode – Exploiting memory bugs, dropping payloads

Script & File Attacks – Scripted execution, file corruption

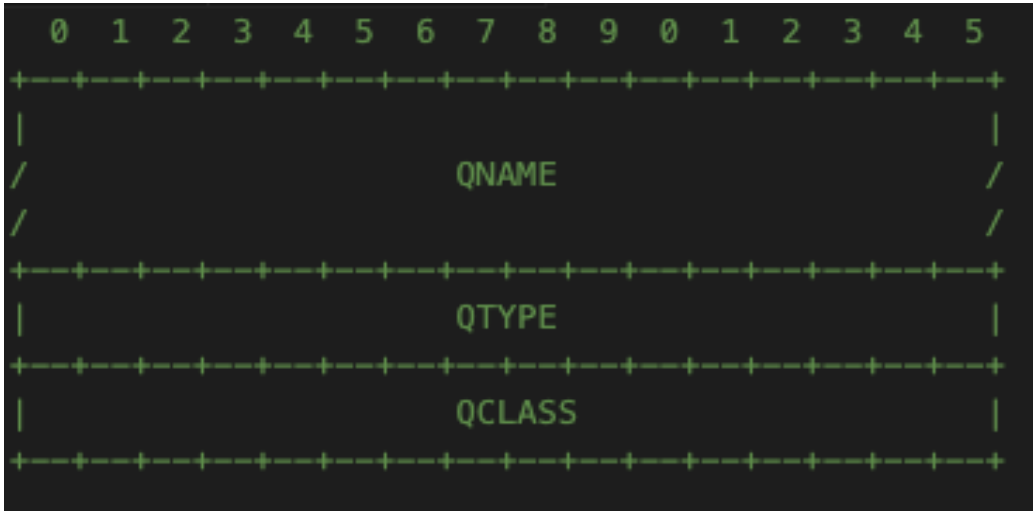
Side-Channel Process Abuse: Processing Injection Hollowing

Persistent Backdoors: Rootkits, ransomware stealth persistence.

Network Pivoting: Port Forwarding, reverse tunnels

DNS Protocol Specifications

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



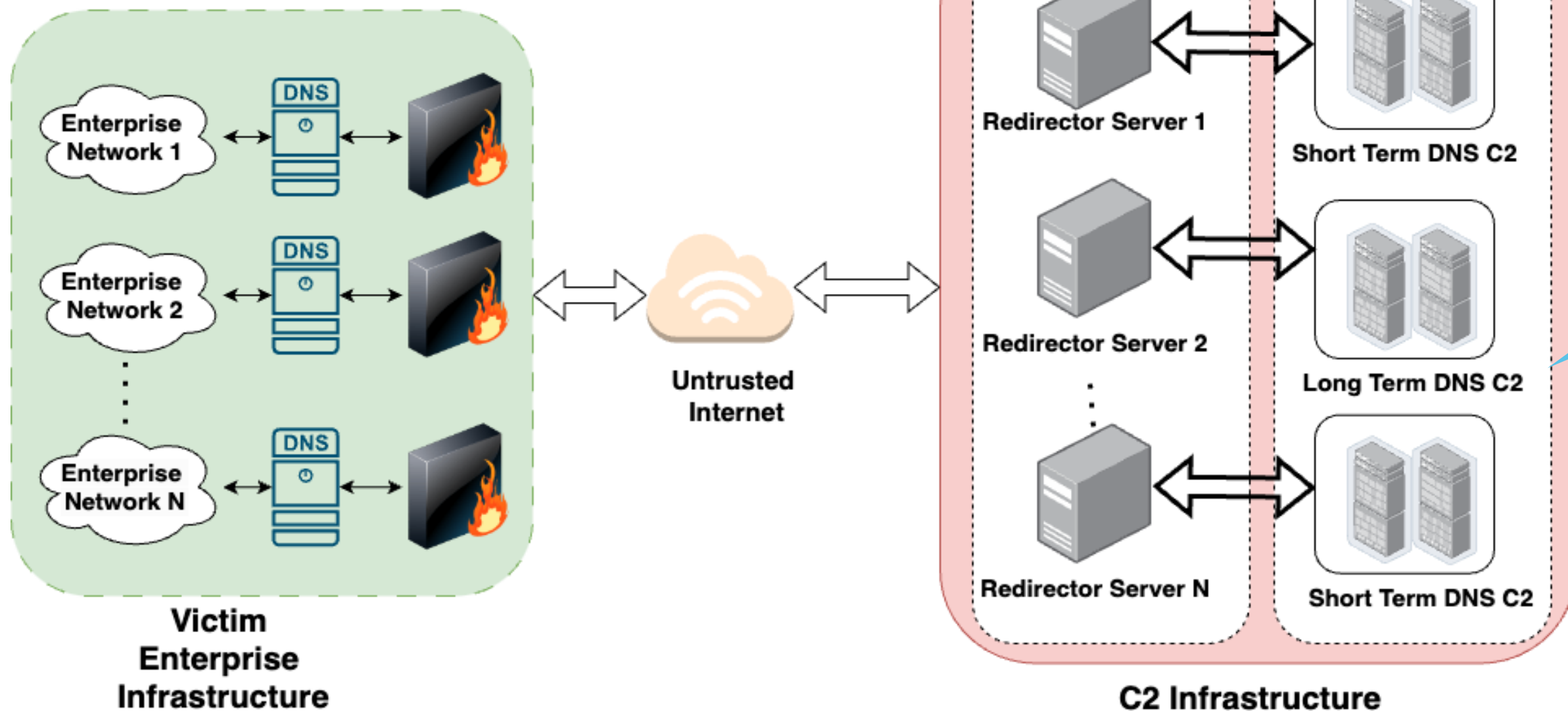
DNS Question Record

What Makes DNS Query contain C2 or exfiltrated data

- ☐ High Entropy QNAME
- ☐ Long or Excessive Labels
- ☐ No Dictionary Tokens
- ☐ DGA-style Patterns
- ☐ NXDOMAIN Abuse and Ghost domains flood

DNS C2 Attack Infrastructure

Redirector
Fleet for
L3 shield C2
Botnet Army



DGA {L7,L3}
Mutation
Powered
C2
Botnet Army

DGA (L7) and IP (L3) Mutation

- ❑ **Evade Detection** – Generates thousands of reflectors, IPS, domains to avoid static and policy blocklists.
- ❑ **Resilience** – If one domain is taken down, others remain reachable.
- ❑ **No Hardcoded domains** – Domains are algorithmically created on both attacker and implant sides.

Time-Based DGAs

Date +
SystemClock
fkeo12jdn7z.com
sk9qpdmx43a.com

Seed-Based DGAs

Seed + shared
math functions
bhack1.com
bhack2.com

Wordlist DGAs

Wordlist
dictionary
catsun.net
reddog.org

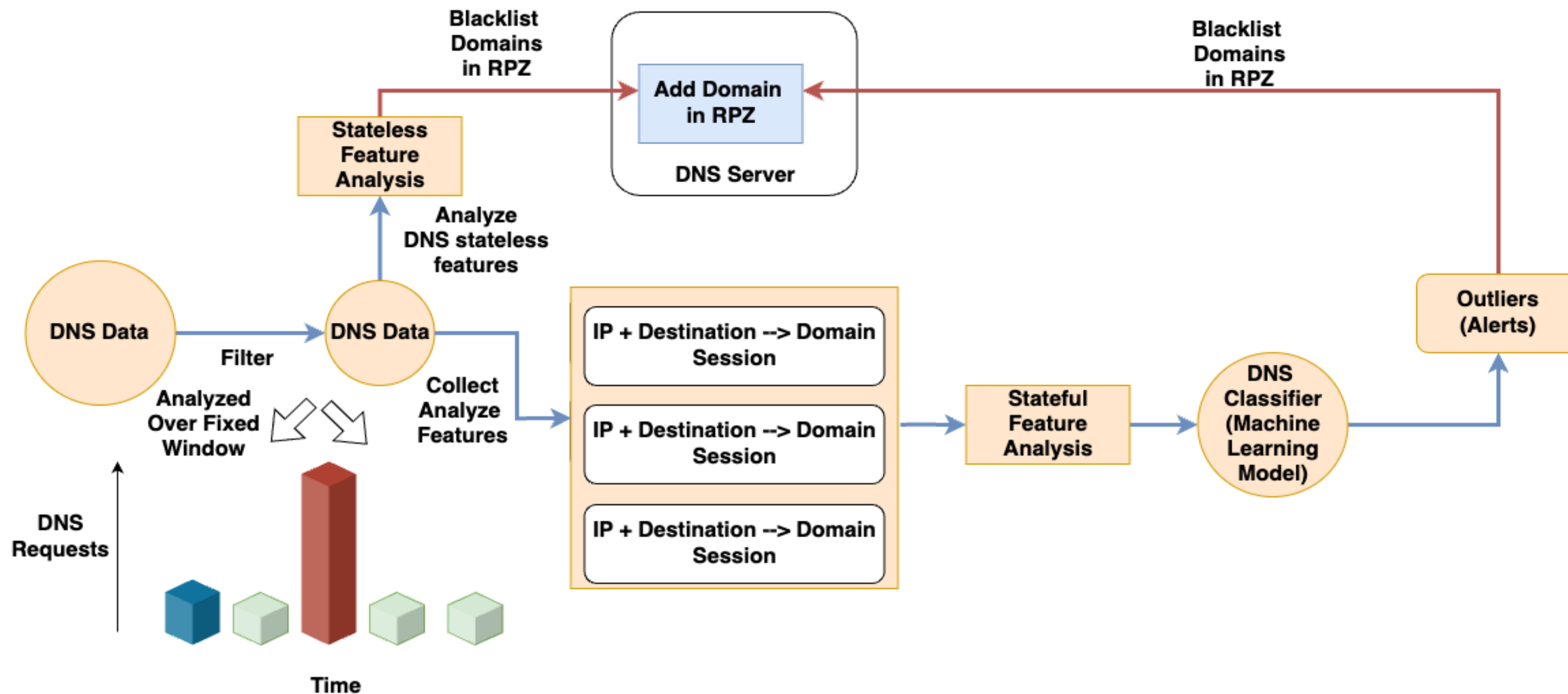
Character-Based or Randomized DGAs

Pseudo random
chars
sdas232.bleed.io

Existing Approaches

- **Semi-Passive Analysis**
 - DNS Exfiltration Security as Middleware (DPI as middleware)
- **Passive Analysis**
 - Anomaly Detection (Traffic Timing / Volume)
 - Threat Signatures, Domain Reputation scoring

DNS Traffic Anomaly Detection and Prevention Pipeline



Challenges with current approaches

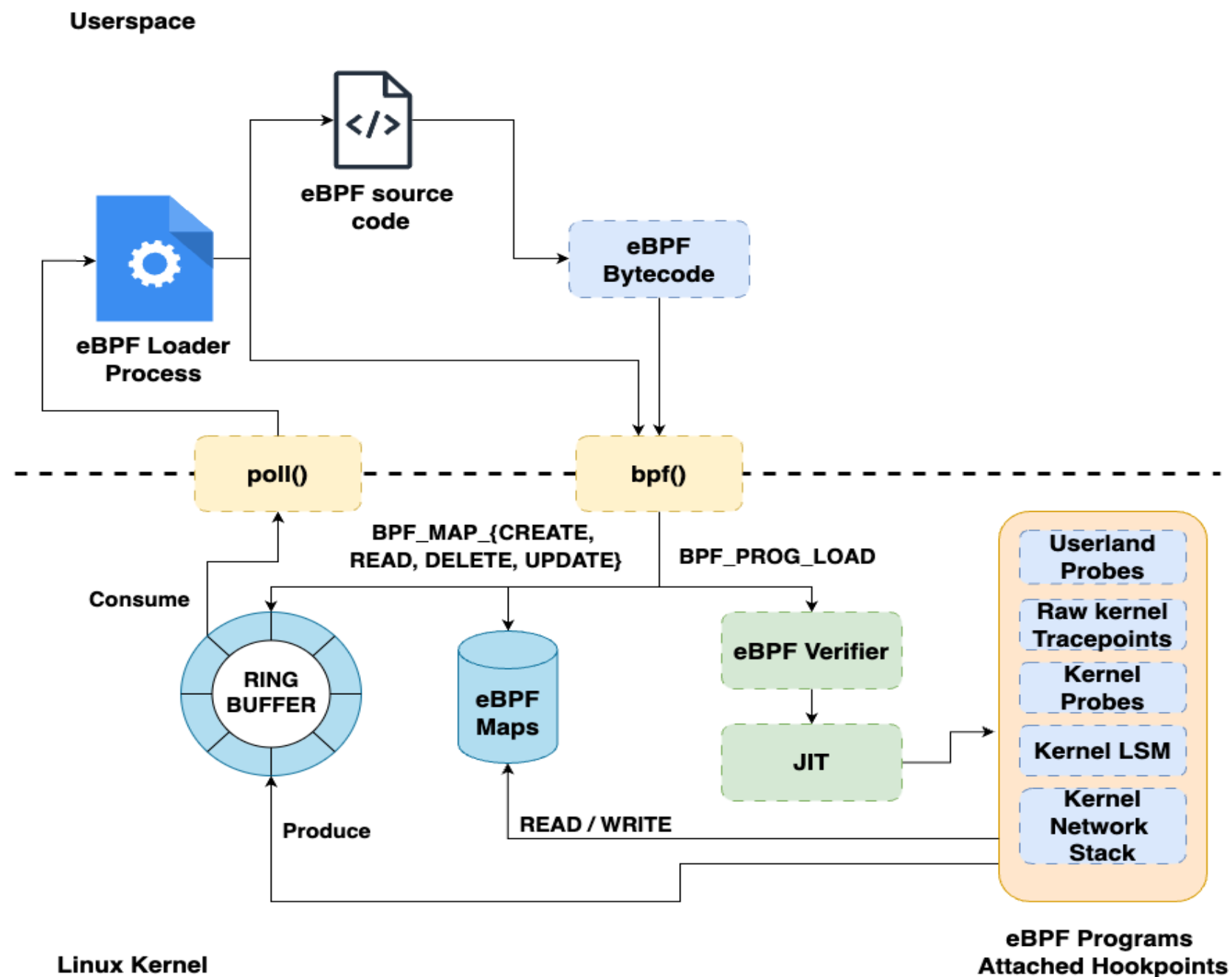
- ❑ **Slow Detection, Slower Response: Stealthy mutable Implants survive**
- ❑ **Slow and easy bypass to Advanced DNS C2 Attacks**
- ❑ **Lack robust protection over Domain Generation Algorithms, IP mutation**
- ❑ **Unwanted latency for proxy-based DPI on benign traffic**
- ❑ **Dynamic Threat Patterns**

Proposed Solution:

- ✓ **Reactive Kernel EDR at Ring 0 — closest to the wire, at the implant source, beyond reach of userland evasion .**

eBPF

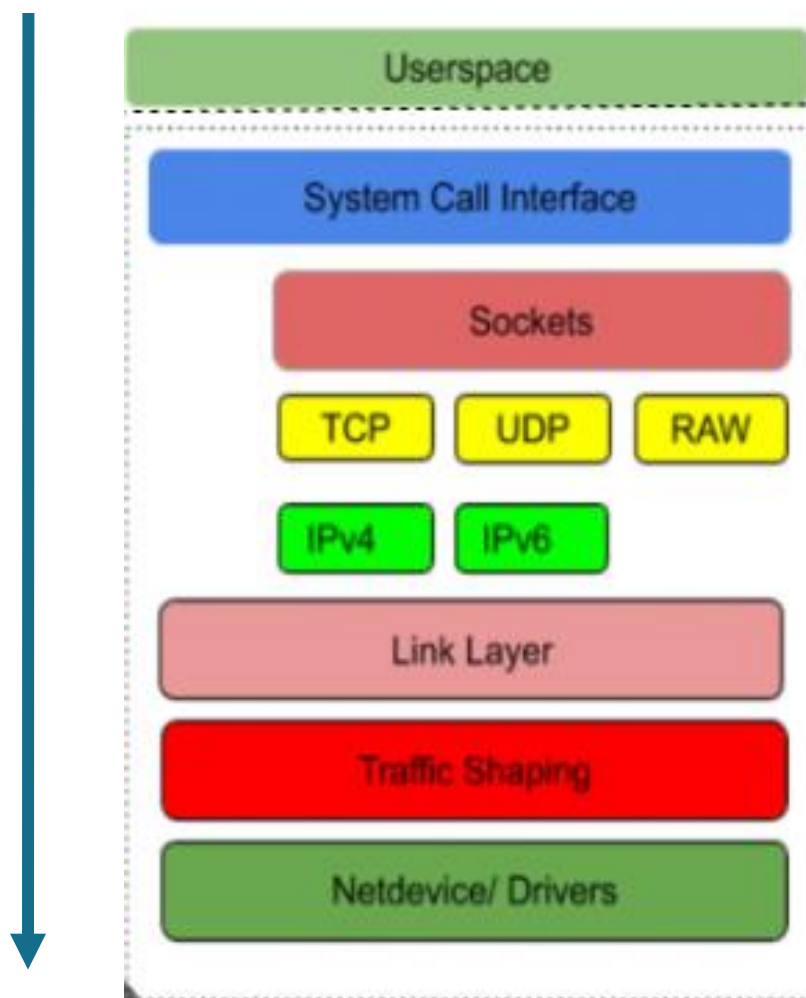
- Reprogram the Linux kernel in safe way.
- Runs BPF virtual machine inside kernel
- Custom BPF bytecode
- CPU architecture and Linux kernel version agnostic (BTF)



EDR Agent Linux Kernel eBPF Hooks

Kernel Network Stack Attachments

Kernel
Process
scheduler



BPF Kprobes/
Tracepoints

BPF Cgroups/
Sockops

DNS Sockets
Process

BPF Netfilter

BPF TC

BPF XDP

Egress DPI
of DNS from
SKB

Kernel MAC (Access Control) Attachments

Userspace

LSM (Linux Security Modules)

Core Kernel Subsystems

BPF LSM

Kernel
Keyring,
LSM
Strong eBPF
program
integrity

Egress

Kernel Enforced Endpoint Security for DNS

Agent based Endpoint Security

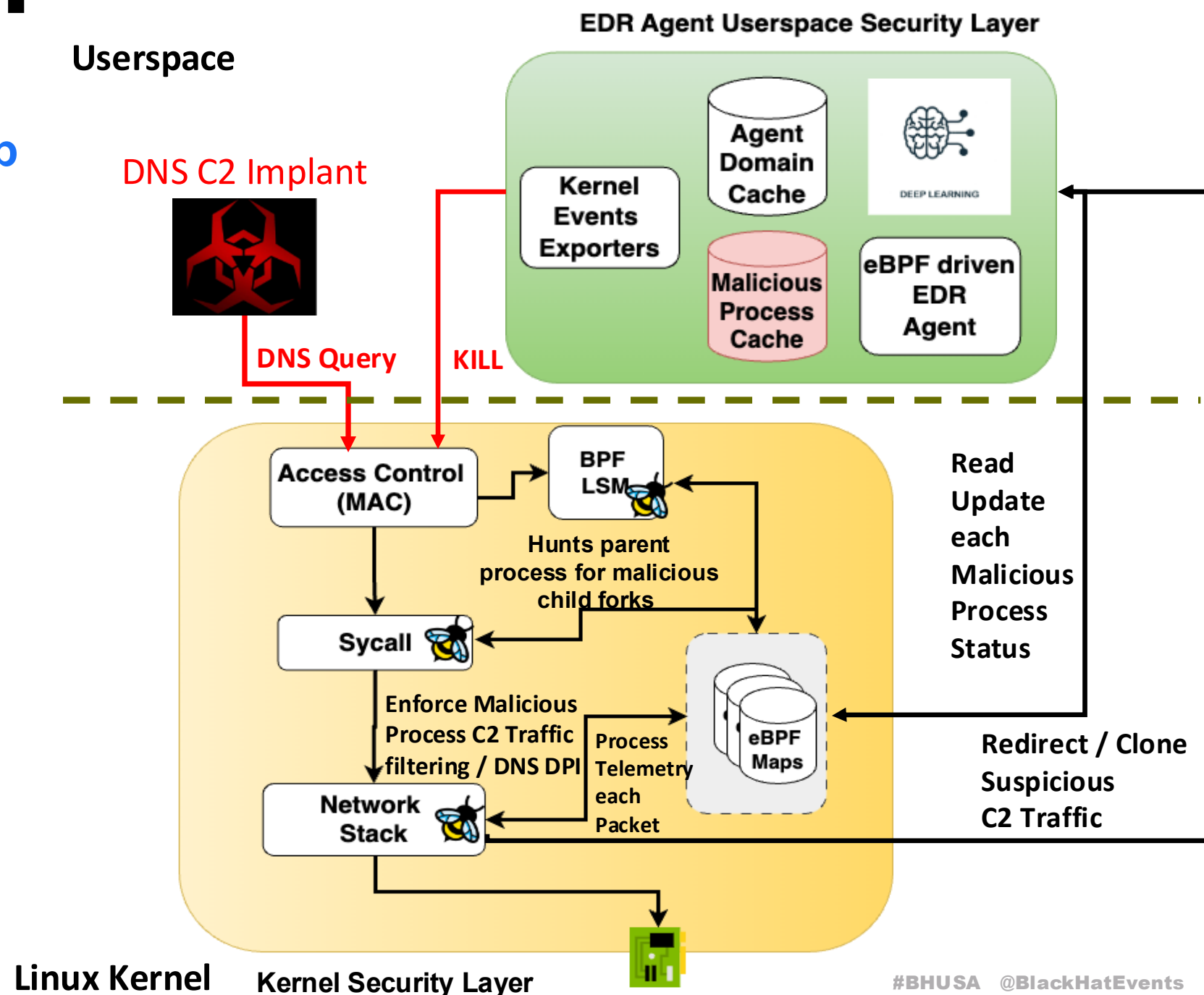
Continuous Security Enforcement Loop

Userspace

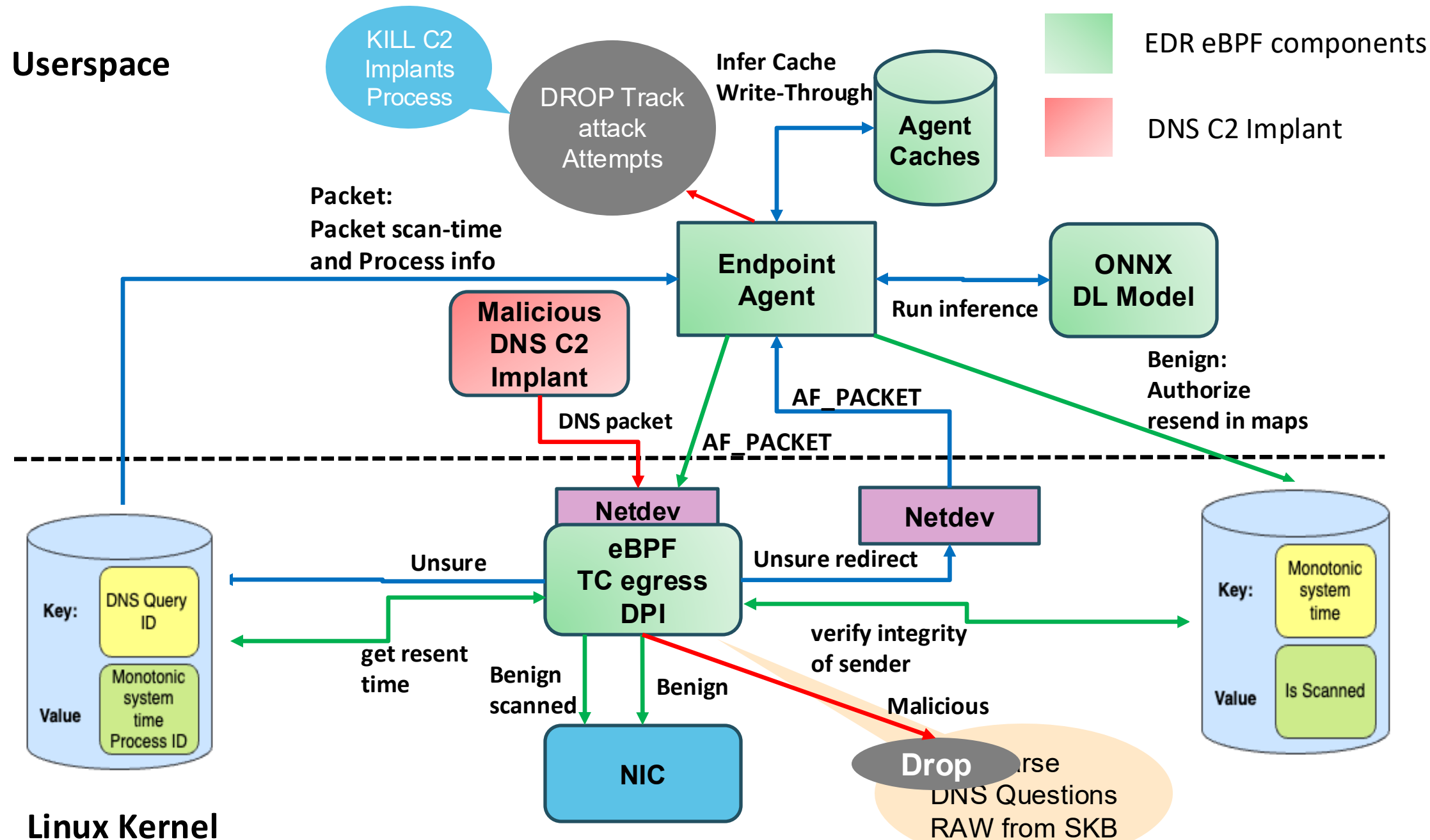
- eBPF Agent
- eBPF Agent Caches
- Quantized Deep Learning Model
- Events malicious metrics exporters

Linux Kernel

- eBPF Ring Buffers
- Access Control Layer (LSM)
- Syscall Layer (Tracepoints)
- Network Stack (TC, Sockets)



EDR Active Process Security Enforcement





DNN based DNS Data Obfuscation Detection (Features)

❑ Kernel Features

❑ Limits for DPI in Kernel

Feature	Description
subdomain_length_per_label	Length of the subdomain per DNS label.
number_of_periods	Number of dots (periods) in the hostname.
total_length	Total length of the domain, including periods/dots.
total_labels	Total number of labels in the domain.
query_class	DNS question class (e.g., IN).
query_type	DNS question type (e.g., A, AAAA, TXT).

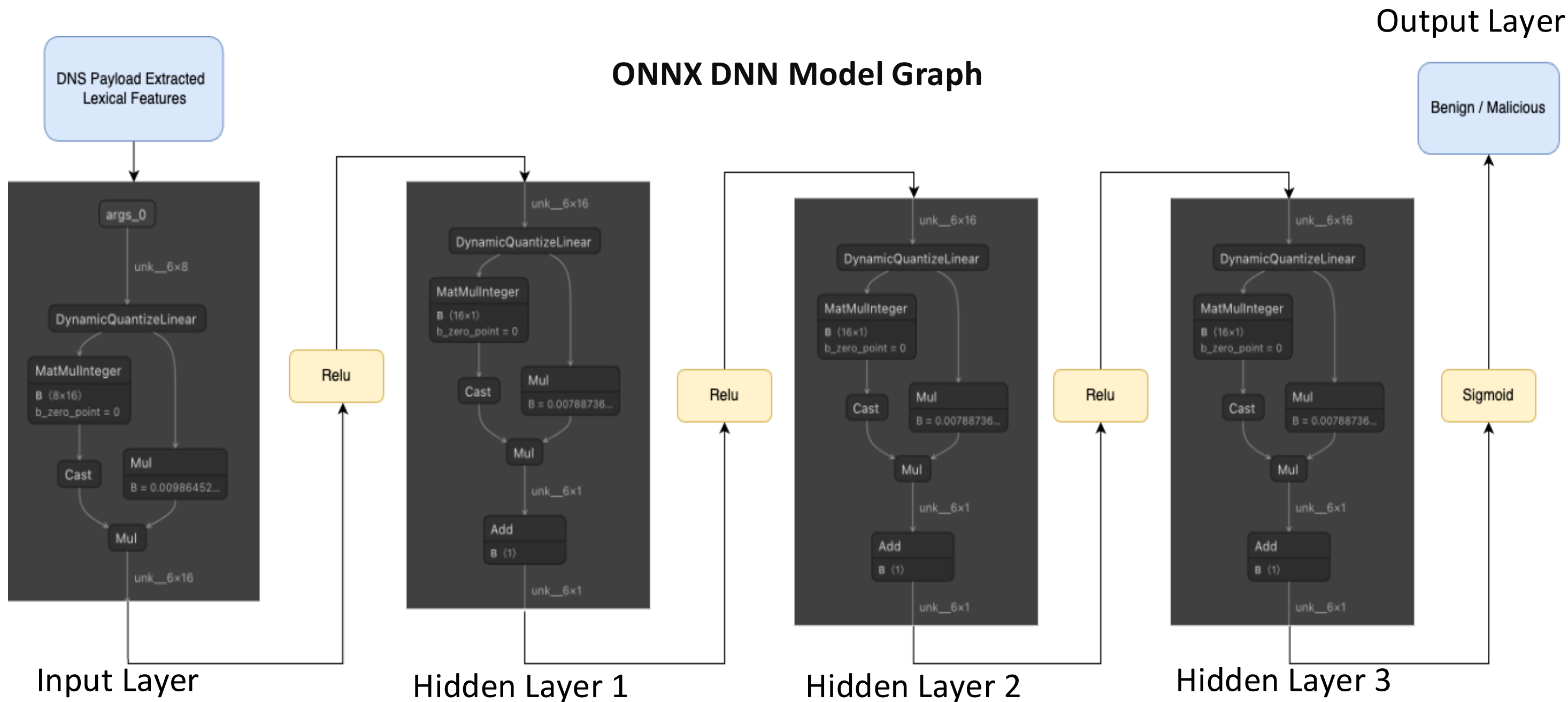
❑ Userspace Features

❑ Enhanced Lexical Features

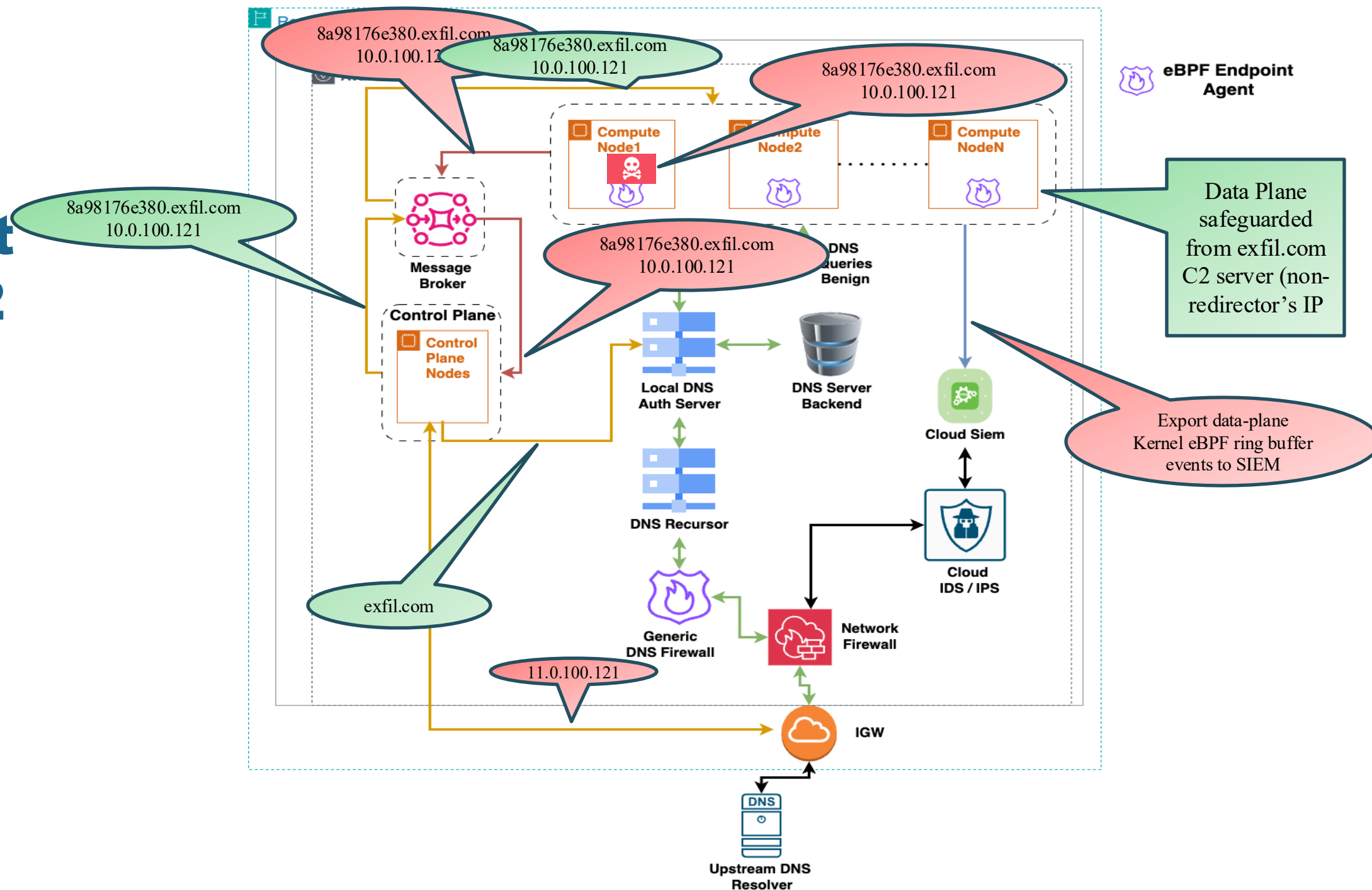
Feature	Description
total_dots	Total number of dots (periods) in DNS query.
total_chars	Total number of characters in DNS query, excluding periods.
total_chars_subdomain	Number of characters in the subdomain portion only.
number	Count of numeric digits in DNS query.
upper	Count of uppercase letters in DNS query.
max_label_length	Maximum label (segment) length in DNS query.
labels_average	Average label length across the request.
entropy	Shannon entropy of the DNS query, indicating randomness.

DNN fueled DNS Data Obfuscation Detection Model

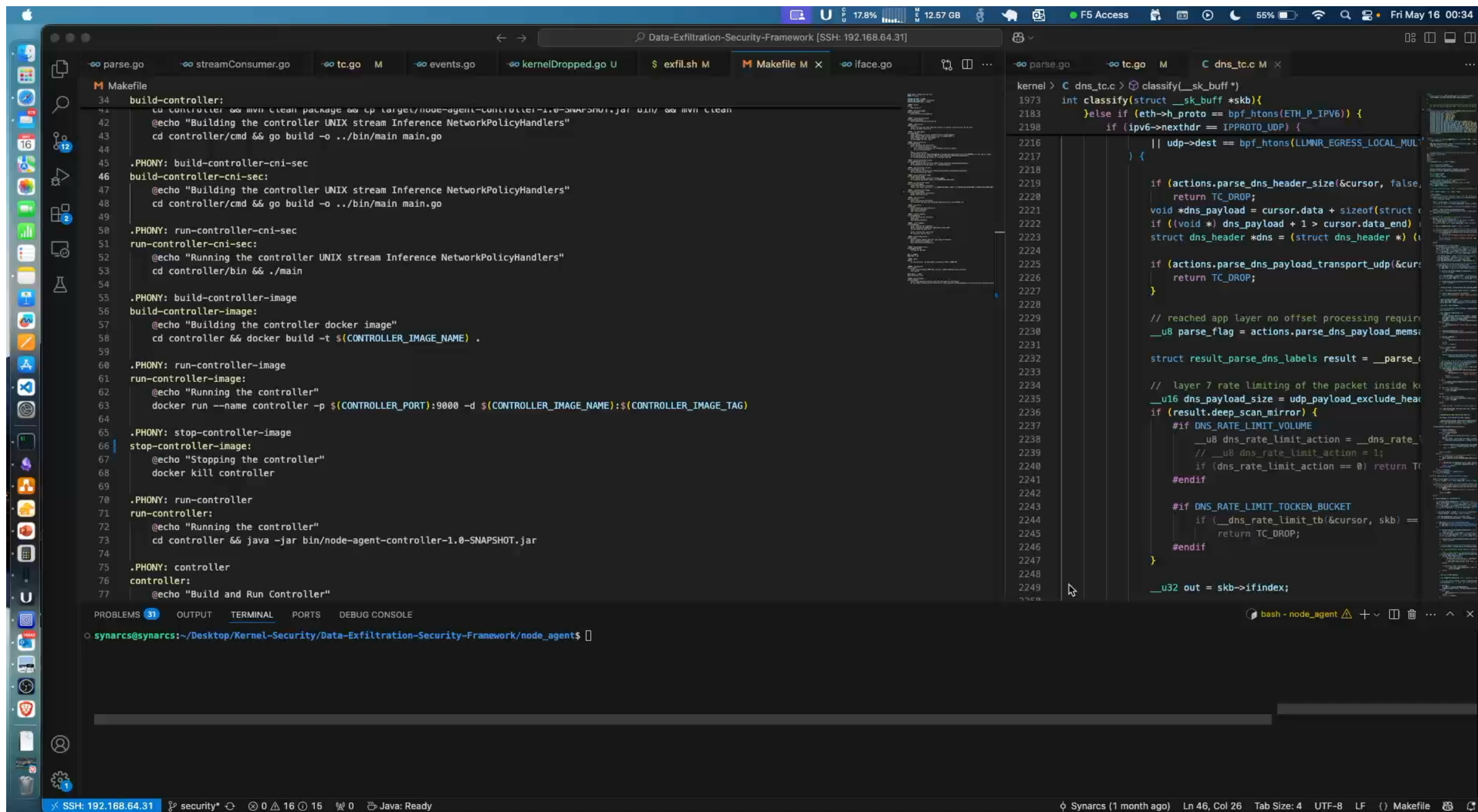
ONNX DNN Model Graph



Framework Deployment in Cloud to Disrupt Remote DNS C2 Infrastructure



Demo



The screenshot shows a macOS desktop environment. The top status bar indicates the date and time as "Fri May 16 00:34". The desktop has a dock on the left with various application icons. The main window is a code editor displaying a Makefile and C code. The Makefile is titled "Makefile" and contains instructions for building and running a controller. The C code is titled "dns_tc.c" and contains a function "classify" that processes network packets. The terminal window at the bottom shows the command prompt "synarcs@synarcs:~/Desktop/Kernel-Security/Data-Exfiltration-Security-Framework/node_agent\$".

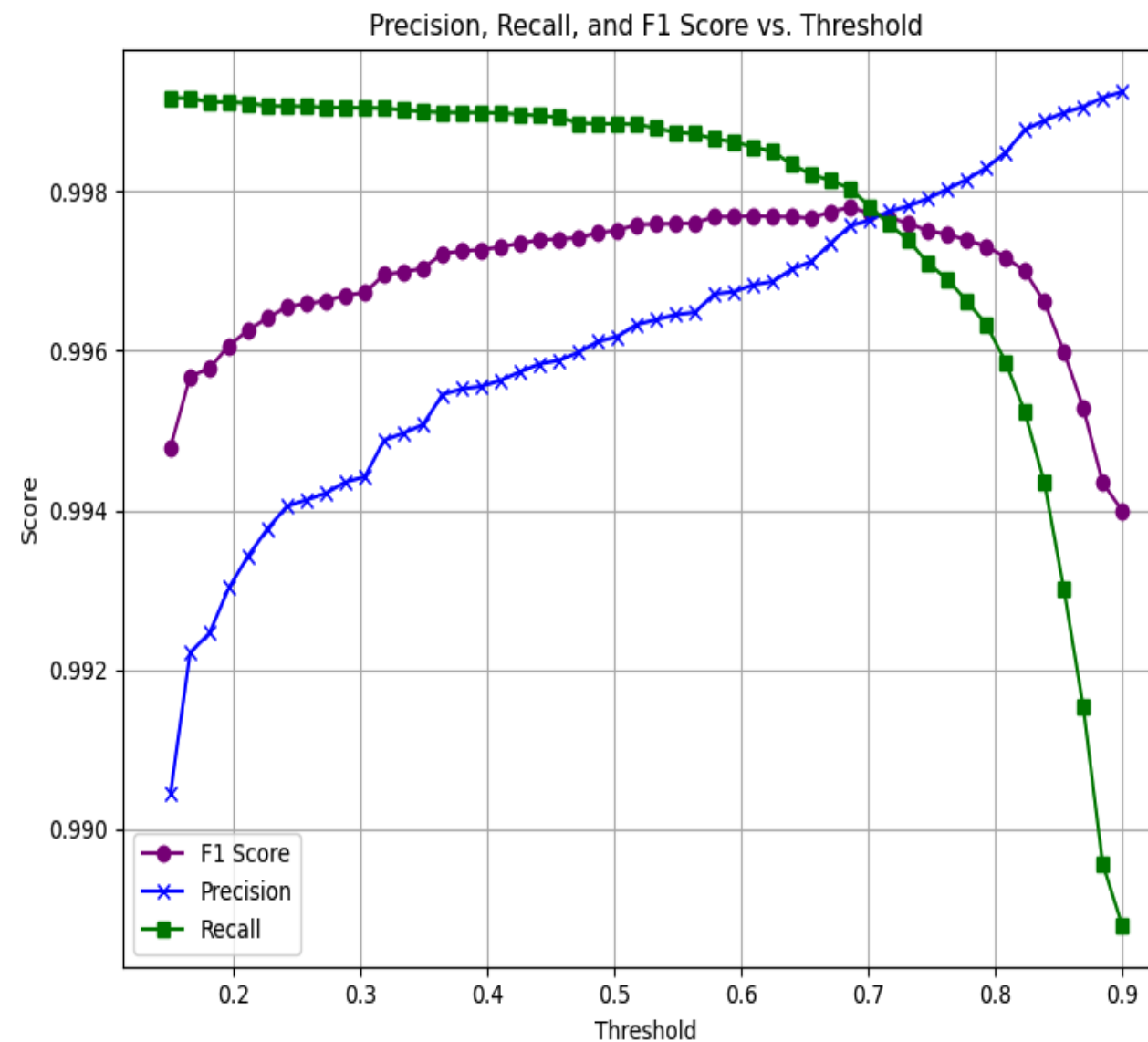
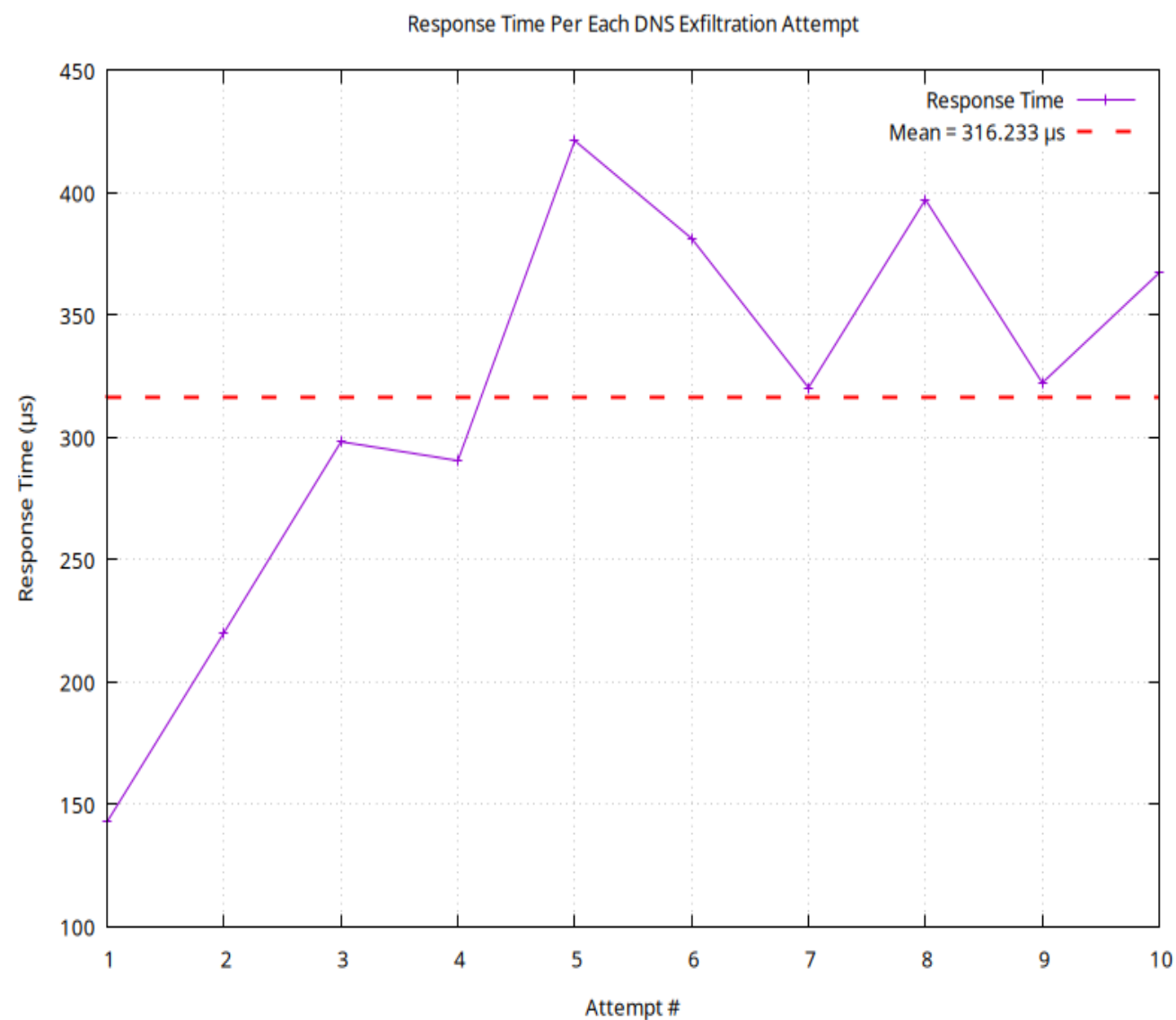
```
Makefile
34 build-controller:
35     cd controller && mvn clean package && cp target/node-agent-controller-1.0-SNAPSHOT.jar bin/ && mvn clean
42     @echo "Building the controller UNIX stream Inference NetworkPolicyHandlers"
43     cd controller/cmd && go build -o ../bin/main main.go
44
45 .PHONY: build-controller-cni-sec
46 build-controller-cni-sec:
47     @echo "Building the controller UNIX stream Inference NetworkPolicyHandlers"
48     cd controller/cmd && go build -o ../bin/main main.go
49
50 .PHONY: run-controller-cni-sec
51 run-controller-cni-sec:
52     @echo "Running the controller UNIX stream Inference NetworkPolicyHandlers"
53     cd controller/bin && ./main
54
55 .PHONY: build-controller-image
56 build-controller-image:
57     @echo "Building the controller docker image"
58     cd controller && docker build -t $(CONTROLLER_IMAGE_NAME) .
59
60 .PHONY: run-controller-image
61 run-controller-image:
62     @echo "Running the controller"
63     docker run --name controller -p $(CONTROLLER_PORT):9000 -d $(CONTROLLER_IMAGE_NAME):$(CONTROLLER_IMAGE_TAG)
64
65 .PHONY: stop-controller-image
66 stop-controller-image:
67     @echo "Stopping the controller"
68     docker kill controller
69
70 .PHONY: run-controller
71 run-controller:
72     @echo "Running the controller"
73     cd controller && java -jar bin/node-agent-controller-1.0-SNAPSHOT.jar
74
75 .PHONY: controller
76 controller:
77     @echo "Build and Run Controller"
```

```
kernel > C dns_tc.c > classify(__sk_buff *)
1973 int classify(struct __sk_buff *skb){
2183     }else if (eth->h_proto == bpf_htons(ETH_P_IPV6)) {
2198     if (ipv6->nexthdr == IPPROTO_UDP) {
2216         || udp->dest == bpf_htons(LLMNR_EGRESS_LOCAL_MULTICAST)
2217     ) {
2218
2219         if (actions.parse_dns_header_size(&cursor, false,
2220             return TC_DROP;
2221         void *dns_payload = cursor.data + sizeof(struct
2222         if ((void *) dns_payload + 1 > cursor.data_end)
2223         struct dns_header *dns = (struct dns_header *) (
2224
2225         if (actions.parse_dns_payload_transport_udp(&cursor,
2226             return TC_DROP;
2227         }
2228
2229         // reached app layer no offset processing required
2230         __u8 parse_flag = actions.parse_dns_payload_memory
2231
2232         struct result_parse_dns_labels result = __parse_c
2233
2234         // layer 7 rate limiting of the packet inside kernel
2235         __u16 dns_payload_size = udp_payload_exclude_header
2236         if (result.deep_scan_mirror) {
2237             #if DNS_RATE_LIMIT_VOLUME
2238             __u8 dns_rate_limit_action = __dns_rate_limit
2239             // __u8 dns_rate_limit_action = 1;
2240             if (dns_rate_limit_action == 0) return TC_DROP;
2241             #endif
2242
2243             #if DNS_RATE_LIMIT_TOKEN_BUCKET
2244             if (__dns_rate_limit_tb(&cursor, skb) ==
2245                 return TC_DROP;
2246             #endif
2247         }
2248
2249         __u32 out = skb->ifindex;
```

bash - node_agent

synarcs@synarcs:~/Desktop/Kernel-Security/Data-Exfiltration-Security-Framework/node_agent\$

Response Speed with Precision



Next Steps

- ❑ **TLS Fingerprinting & Tunnel Detection:** eBPF-based TLS fingerprinting to detect, hunt, and block exfiltration over encrypted channels (TLS, WireGuard).
- ❑ **Process Correlation:** Kernel eBPF programs and agents correlate cross-protocol exfil attempts to originating processes for precise blocking.
- ❑ **AI-Driven Detection:** Real-time drift detection, confidence-based updates, and GAN+LSTM models adapt to DNS obfuscation and kernel event patterns in eBPF maps.
- ❑ **DNS DDoS Guard:** eBPF-based endpoint defense against NXDOMAIN floods and DNS-C2 ghost domain flood.

Black Hat Sound Bytes

- **AI + eBPF for Real-Time Kernel Threat Hunting:** Dynamically detect and disrupt C2 implants in-kernel, boosting EDR with adaptive, AI-driven kernel enforcements.
- **Dynamic Kernel powered EDR fuels Cloud NACL's:** Enforce L3 filters at the endpoint and sync with cloud firewalls to disrupt DGA and evolving C2 infrastructure.
- **Deep OS Telemetry powers SIEM/SOAR:** Kernel-powered visibility via eBPF feeds rich behavioral signals into upstream SIEM and matures SOAR.

Thank You



Framework Codebase: <https://github.com/Synarcs/DNSObelisk>

Framework WhitePaper: <https://shorturl.at/42dVC>