black hat®
BRIEFINGS

AUGUST 6-7, 2025
MANDALAY BAY / LAS VEGAS

From Packet to Process: Hunting DNS C2 Implants in the Linux Kernel with eBPF for Cloud Environments

Speaker: Vedang Parasnis (Synarcs)

$whoami

Vedang Parasnis

Independent Researcher,

Master's Graduate @University Of Washington

Email: vedang.parasnis@outlook.com

Research Interests:

Kernel security hardening, eBPF, cloud, platform 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 — Every Time

Compromise Supply Chain:

- APT29 (Cozy Bear) — SolarWinds

Breach Cloud & Hyperscalers:

- UNC2452 (APT29)

Damage Critical Infrastructure:

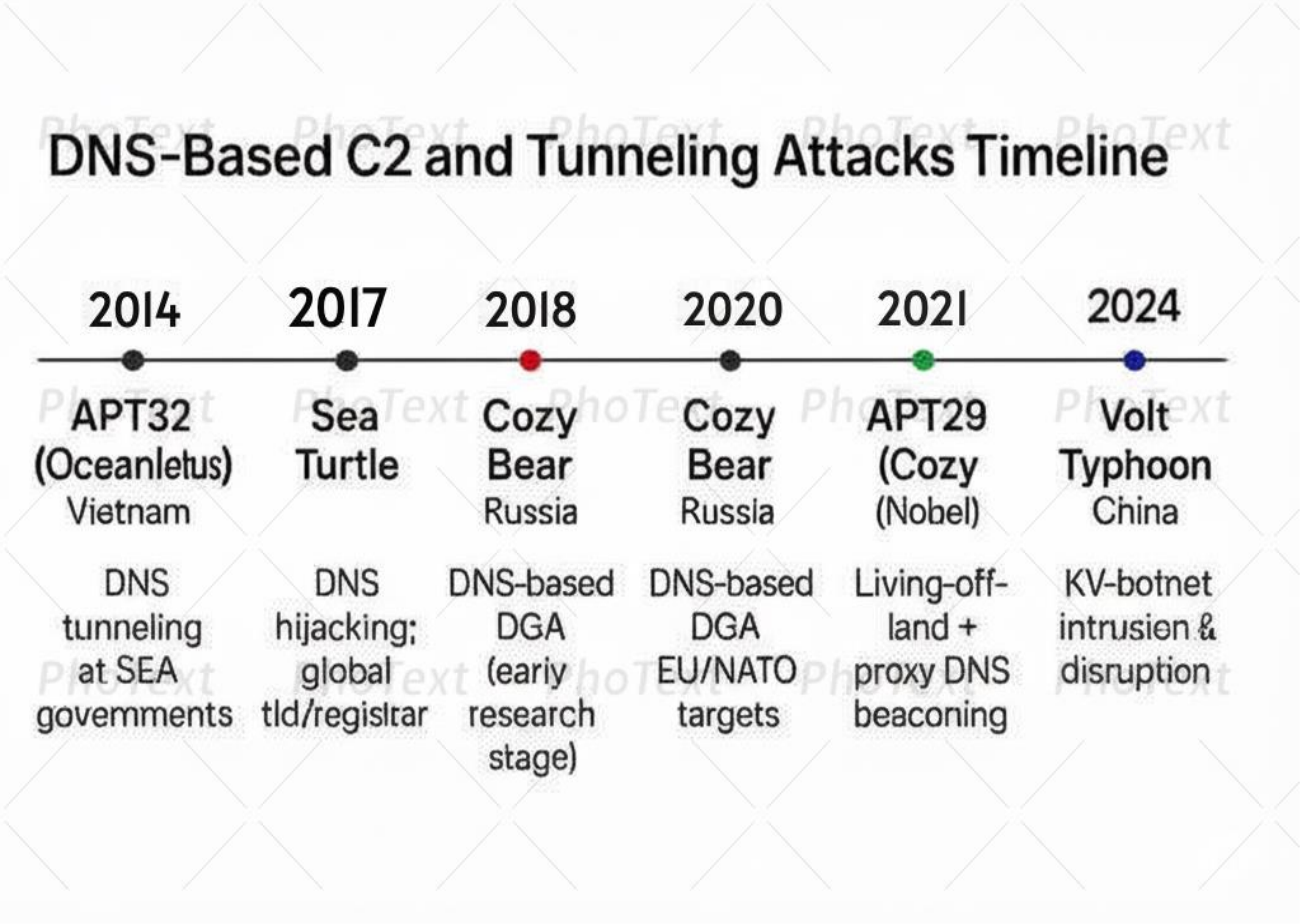
- Volt Typhoon

Harvest Credentials at Scale:

- APT28 (GRU), Sea Turtle

Exploit Shared Offensive Tools:

- APT41, FIN7



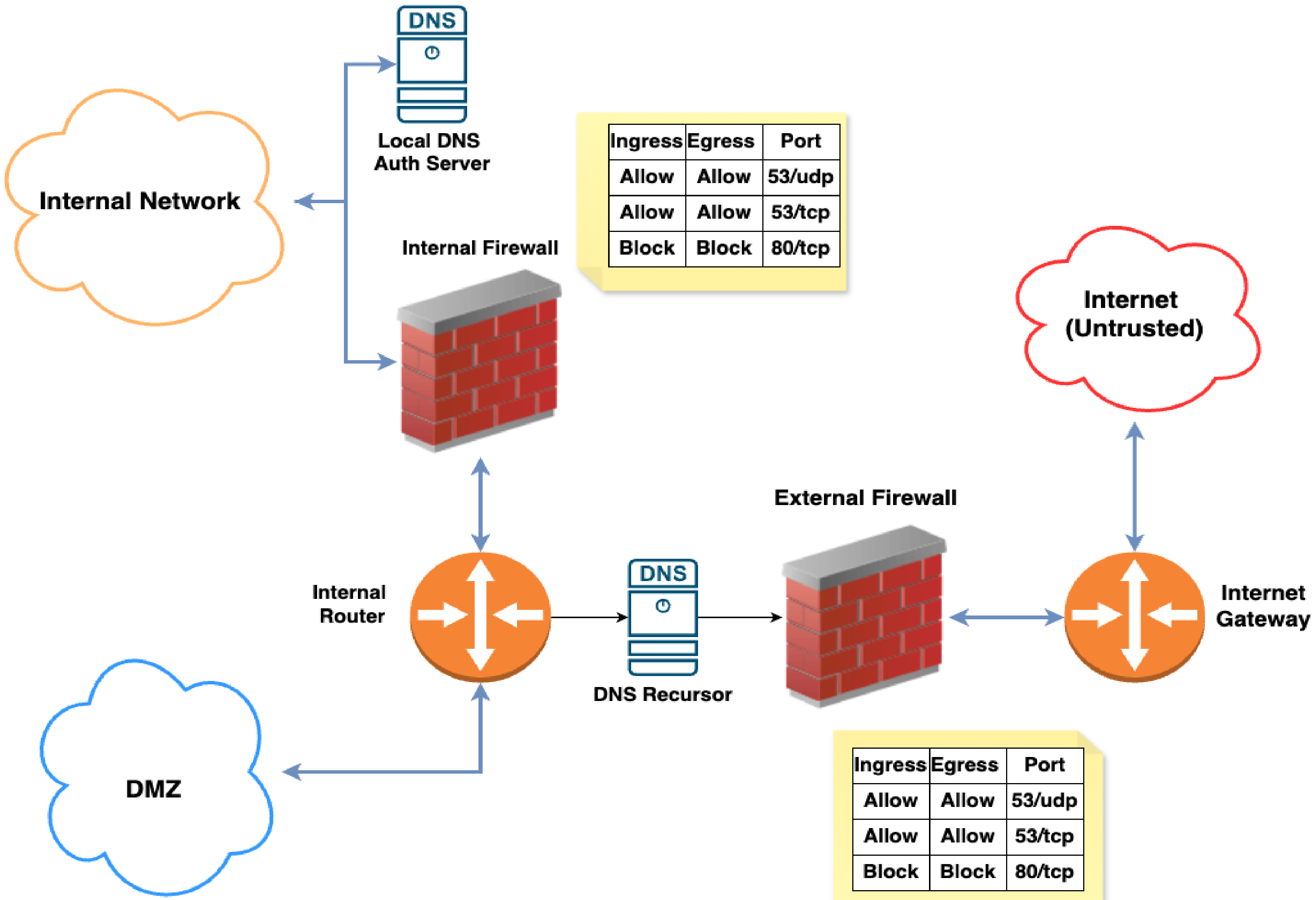
DNS-Based C2 and Tunneling Attacks Timeline

2014	2017	2018	2020	2021	2024
APT32 (Oceanletus) Vietnam	**Sea Turtle**	**Cozy Bear** Russia	**Cozy Bear** Russia	**APT29 (Cozy** (Nobel)	**Volt Typhoon** China
DNS tunneling at SEA govemments	DNS hijacking; global tld/regisltar	DNS-based DGA (eariy research stage)	DNS-based DGA EU/NATO targets	Living-off-land + proxy DNS beaconing	KV-botnet intrusien & disruption

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



Ingress	Egress	Port
Allow	Allow	53/udp
Allow	Allow	53/tcp
Block	Block	80/tcp

Ingress	Egress	Port
Allow	Allow	53/udp
Allow	Allow	53/tcp
Block	Block	80/tcp

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.

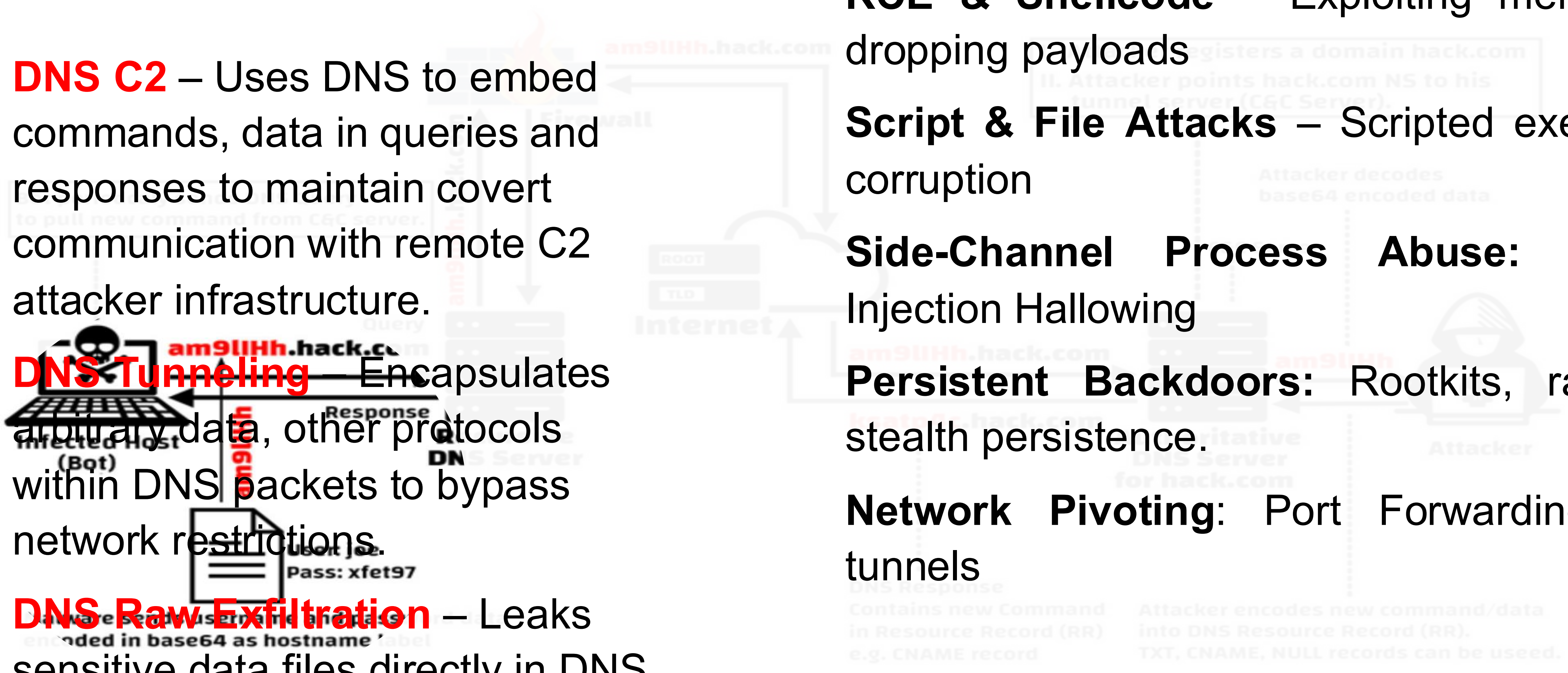
RCE & Shellcode – Exploiting memory bugs, dropping payloads

Script & File Attacks – Scripted execution, file corruption

Side-Channel Process Abuse: Processing Injection Hallowing

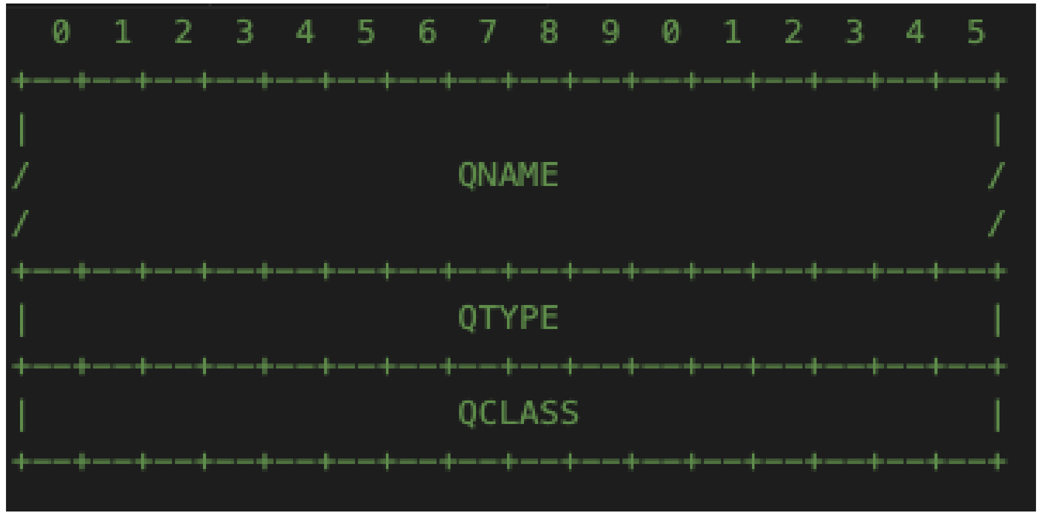
Persistent Backdoors: Rootkits, ransomware stealth persistence.

Network Pivoting: Port Forwarding, reverse tunnels


am9lIHh.hack.com
Firewall
to pull new command from C&C server.
Query
Response
Infected Host (Bot)
DNS Server
User: joe.
Pass: xfet97
Malware sends username and password encoded in base64 as hostname label
ROOT
TLD
Internet
I. Attacker registers a domain hack.com
II. Attacker points hack.com NS to his tunnel server (C&C Server).
Attacker decodes base64 encoded data
Authoritative DNS Server for hack.com
Attacker
DNS Response Contains new Command in Resource Record (RR) e.g. CNAME record
Attacker encodes new command/data into DNS Resource Record (RR). TXT, CNAME, NULL records can be useed.

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



```
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                     QNAME                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QTYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QCLASS                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

DNS Question Record

What Makes DNS Query contain C2 or exfiltrated data

❑ **High Entropy QNAME** – Encrypted or binary-encoded payloads

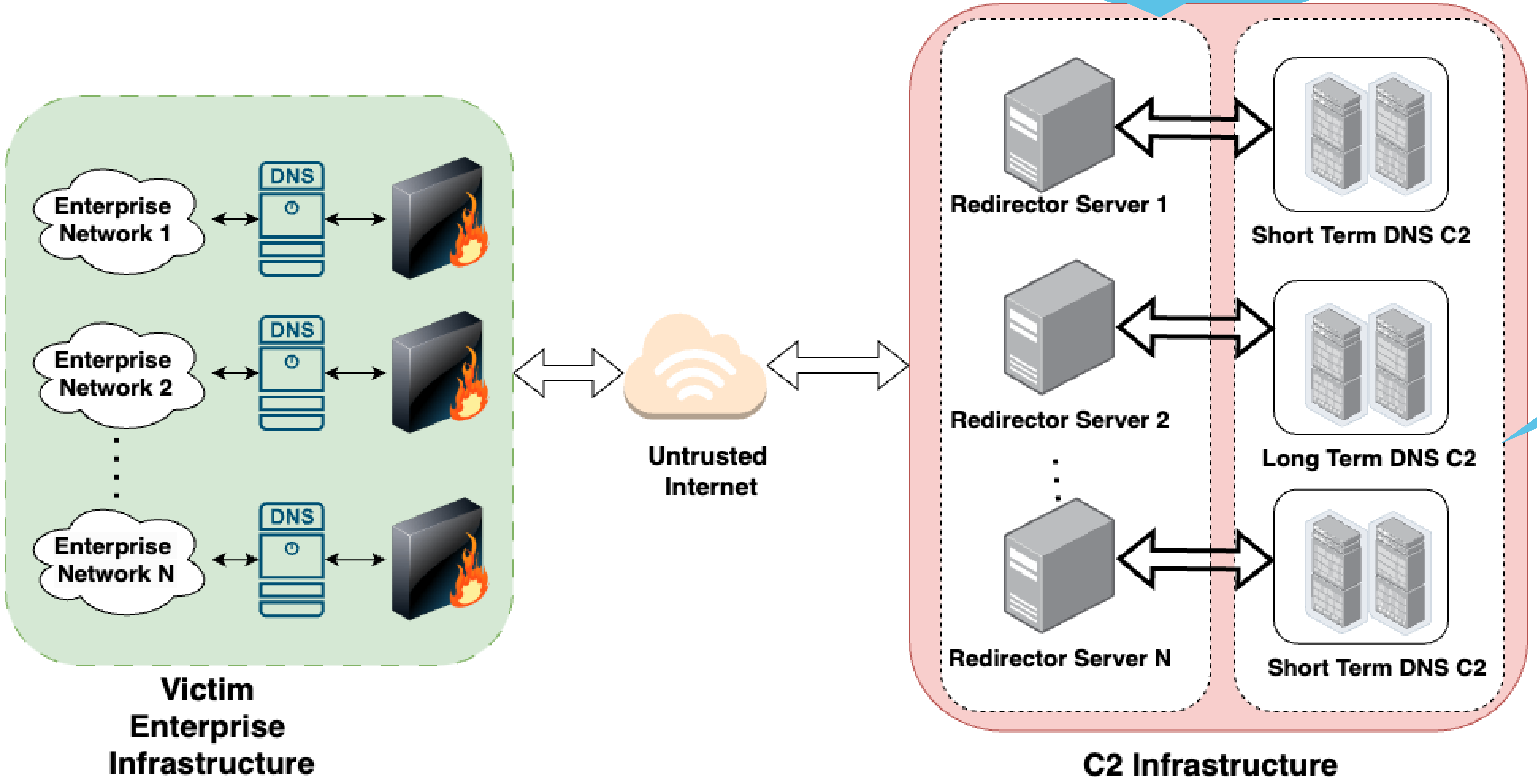
❑ **Long or Excessive Labels** – Chained subdomains to chunk and smuggle data

❑ **No Dictionary Tokens** – Encoded strings, no legit words — signals data, not domains

❑ **DGA-style Patterns** – Time/seed-based domains — predictable but meaningless

❑ **NXDOMAIN Abuse** – Ghost domains used for covert signaling, no resolution needed

DNS C2 Attack Infrastruct



Redirector Fleet to L3 mask 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. **(Evades automated static playbooks)**

□ **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

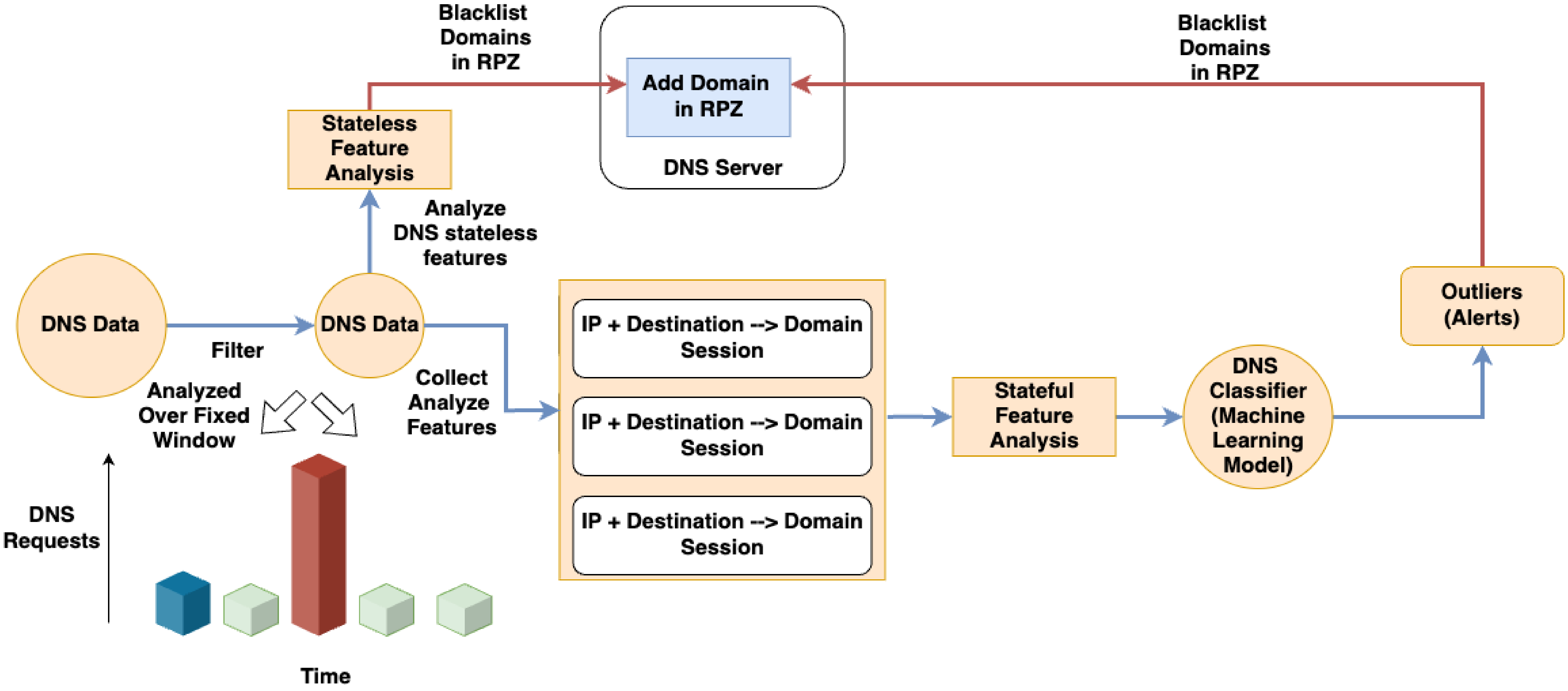
Character-Based or Randomized DGAs

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


Blacklist Domains in RPZ
Add Domain in RPZ
DNS Server
Blacklist Domains in RPZ
Stateless Feature Analysis
Analyze DNS stateless features
DNS Data
Filter
DNS Data
Analyzed Over Fixed Window
Collect Analyze Features
IP + Destination --> Domain Session
IP + Destination --> Domain Session
IP + Destination --> Domain Session
Stateful Feature Analysis
DNS Classifier (Machine Learning Model)
Outliers (Alerts)
DNS Requests
Time

Challenges with current approaches

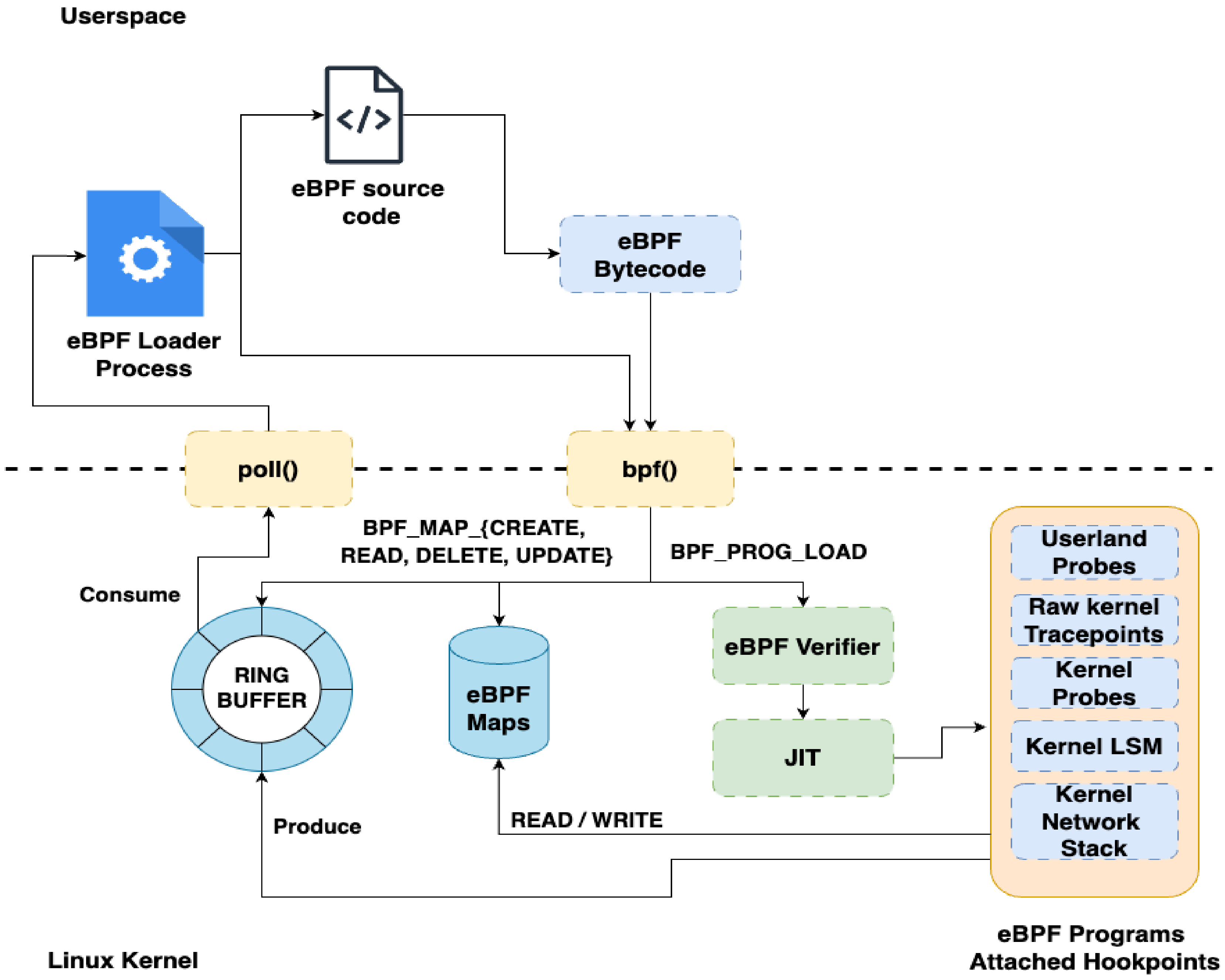
- ❑ Slow Detection, Slower Response: Stealthy mutable Implants survive
- ❑ Slow and easy bypass to Advanced DNS C2 Attacks
- ❑ Lack robust protection over Domain Generation Algorithms, IP mutation
- ❑ Unwanted latency for proxy-based DPI on legit traffic
- ❑ Dynamic Threat Patterns

Proposed Solution:

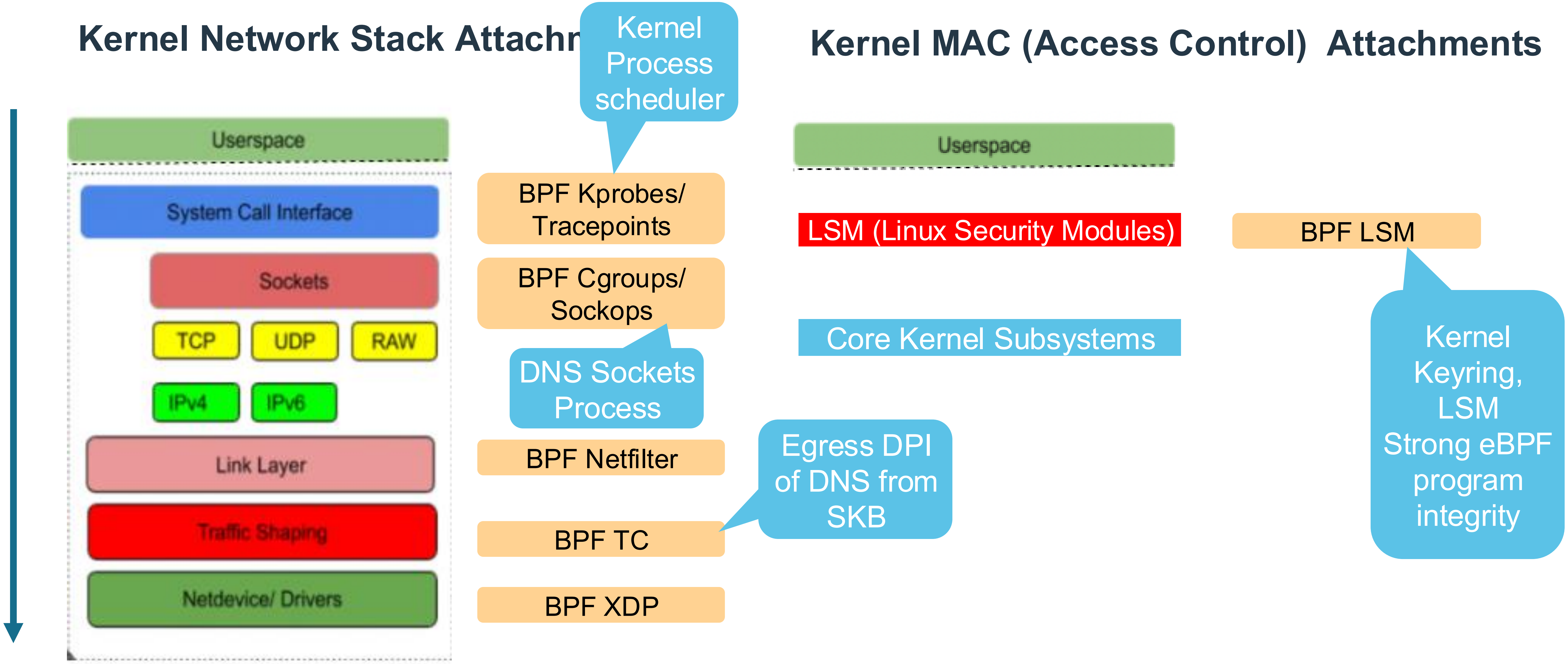
✓ **Reactive Kernel EDR at Ring 0 — closest to the wire, beyond reach of userland evasion.**

eBPF

- Reprogram the Linux kernel in safe way.
- Runs BPF virtual machine inside kernel
- Custom BPF bytecode
- Uses 512 bytes of stack
- eBPF Maps as heap
- CPU architecture and Linux kernel version agnostic (BTF)



EDR Agent Linux Kernel eBPF Hooks



Kernel Network Stack Attachr

Kernel MAC (Access Control) Attachments

Egress

Kernel Enforced Endpoint Security for DNS

Agent based Endpoint Security

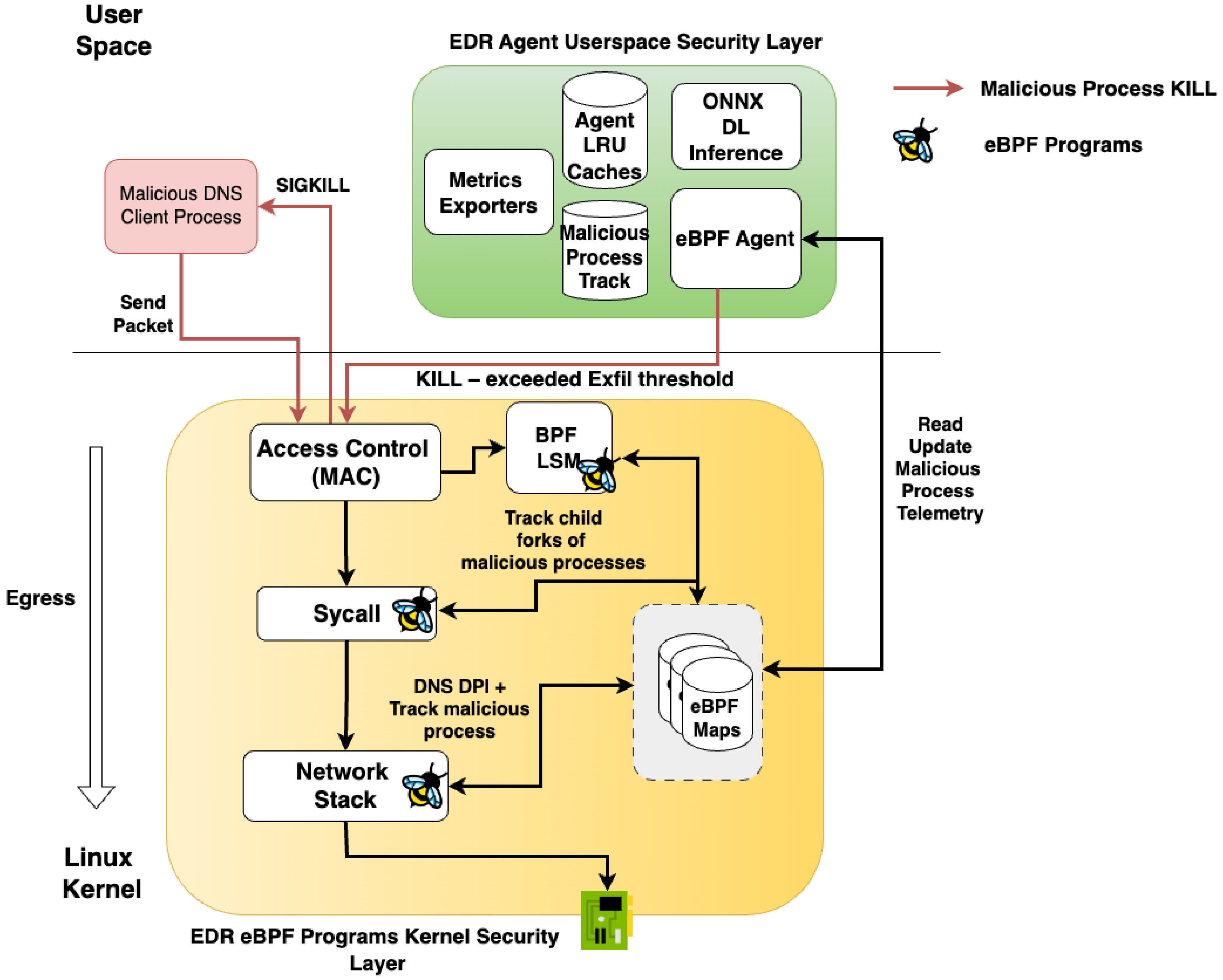
Continuous Security Enforcement Loop

Userspace

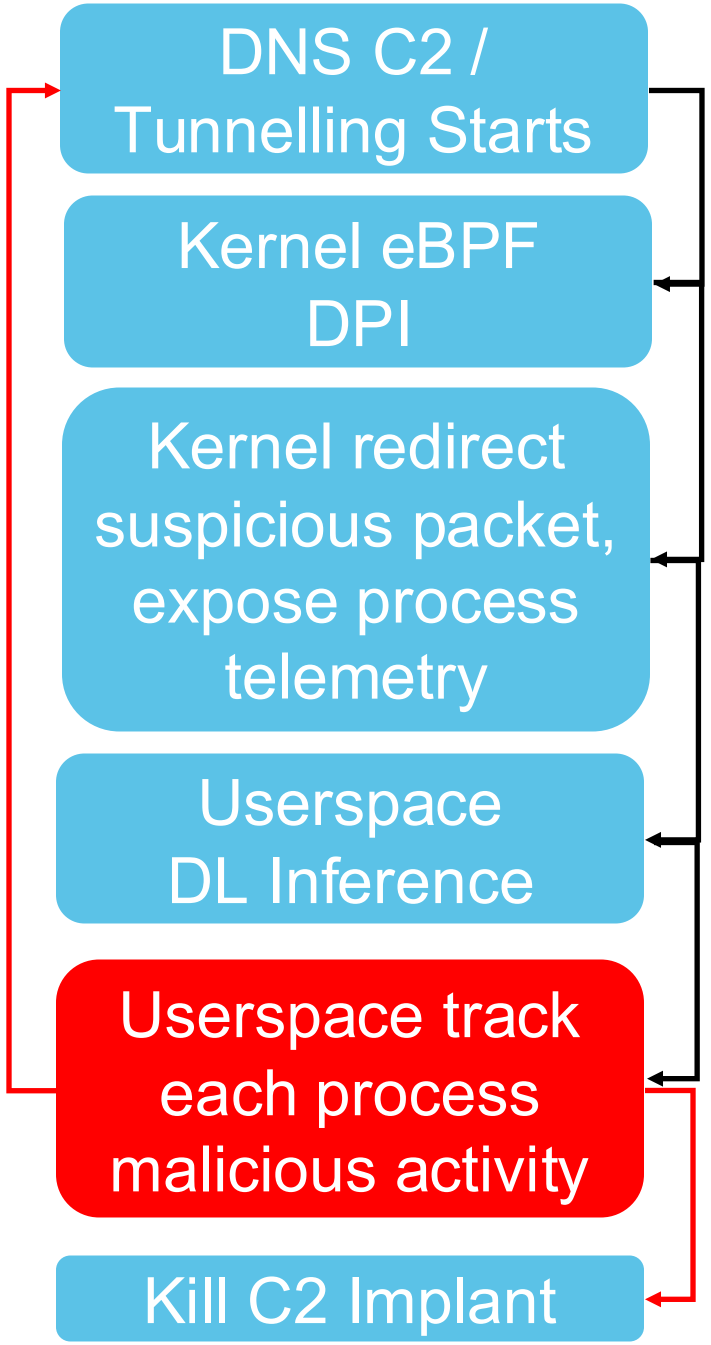
- eBPF Agent
- eBPF Agent Caches
- ONNX Quantized Deep Learning Model
- Events malicious metrics exporters

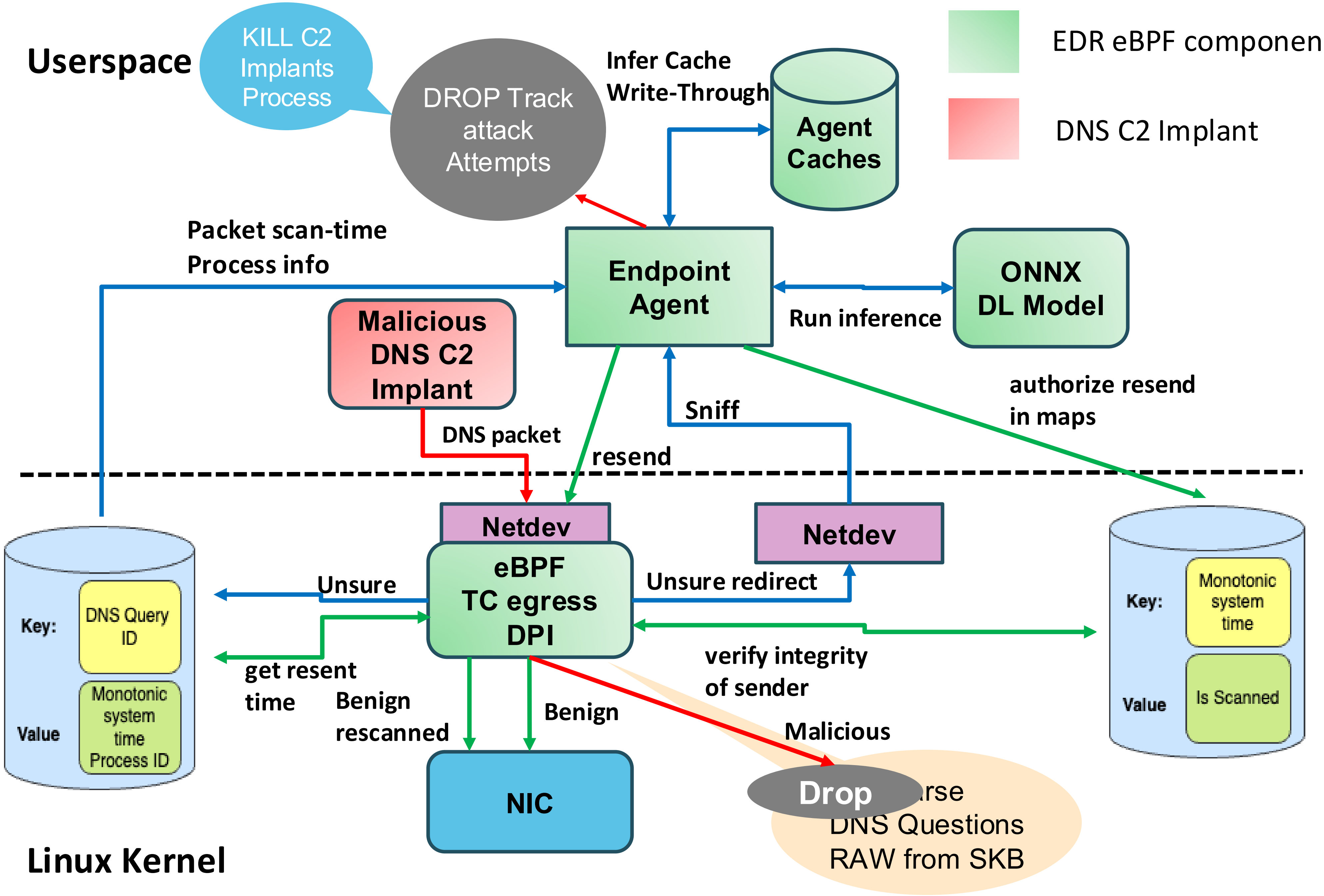
Linux Kernel

- eBPF Ring Buffers
- Network Stack (eBPF programs)
- Access Control Layer (eBPF programs)


User Space
EDR Agent Userspace Security Layer
→ Malicious Process KILL
eBPF Programs
Malicious DNS Client Process
SIGKILL
Metrics Exporters
Agent LRU Caches
ONNX DL Inference
Malicious Process Track
eBPF Agent
Send Packet
KILL – exceeded Exfil threshold
Read Update Malicious Process Telemetry
Access Control (MAC)
BPF LSM
Track child forks of malicious processes
Egress
Sycall
DNS DPI + Track malicious process
eBPF Maps
Network Stack
Linux Kernel
EDR eBPF Programs Kernel Security Layer

EDR Agent Active Process Security Enforcement

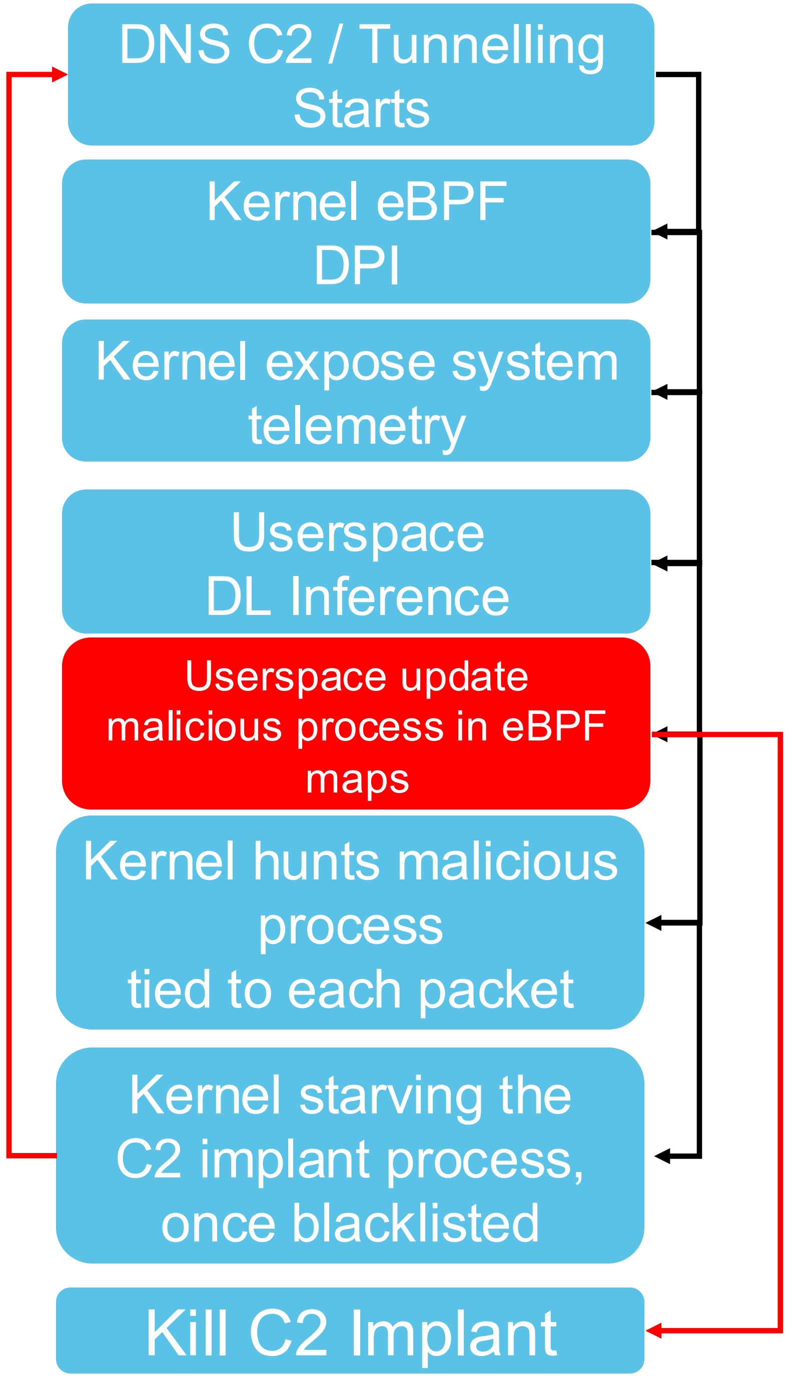

DNS C2 / Tunnelling Starts
Kernel eBPF DPI
Kernel redirect suspicious packet, expose process telemetry
Userspace DL Inference
Userspace track each process malicious activity
Kill C2 Implant



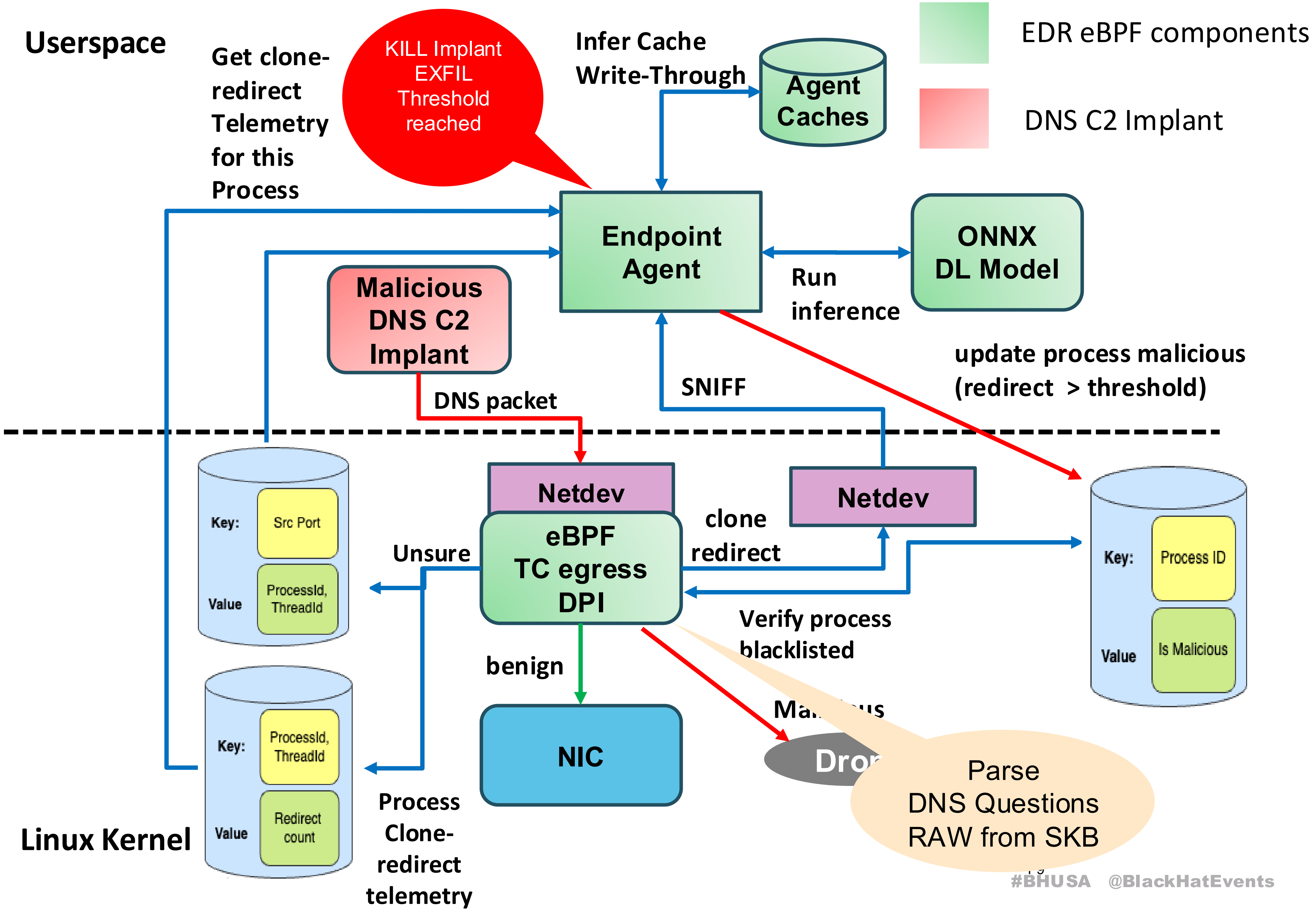
🟩 EDR eBPF components

🟥 DNS C2 Implant

EDR Agent Passive Process Security Enforcement



- DNS C2 / Tunnelling Starts
- Kernel eBPF DPI
- Kernel expose system telemetry
- Userspace DL Inference
- Userspace update malicious process in eBPF maps
- Kernel hunts malicious process tied to each packet
- Kernel starving the C2 implant process, once blacklisted
- Kill C2 Implant



🟩 EDR eBPF components

🟥 DNS C2 Implant

Userspace

Get clone-redirect Telemetry for this Process

KILL Implant EXFIL Threshold reached

Infer Cache Write-Through

Agent Caches

Endpoint Agent

ONNX DL Model

Run inference

Malicious DNS C2 Implant

DNS packet

SNIFF

update process malicious (redirect > threshold)

Key: Src Port

Value: ProcessId, ThreadId

Netdev

eBPF TC egress DPI

Netdev

Unsure

clone redirect

Verify process blacklisted

Key: Process ID

Value: Is Malicious

benign

NIC

Ma us

Dro

Parse DNS Questions RAW from SKB

Key: ProcessId, ThreadId

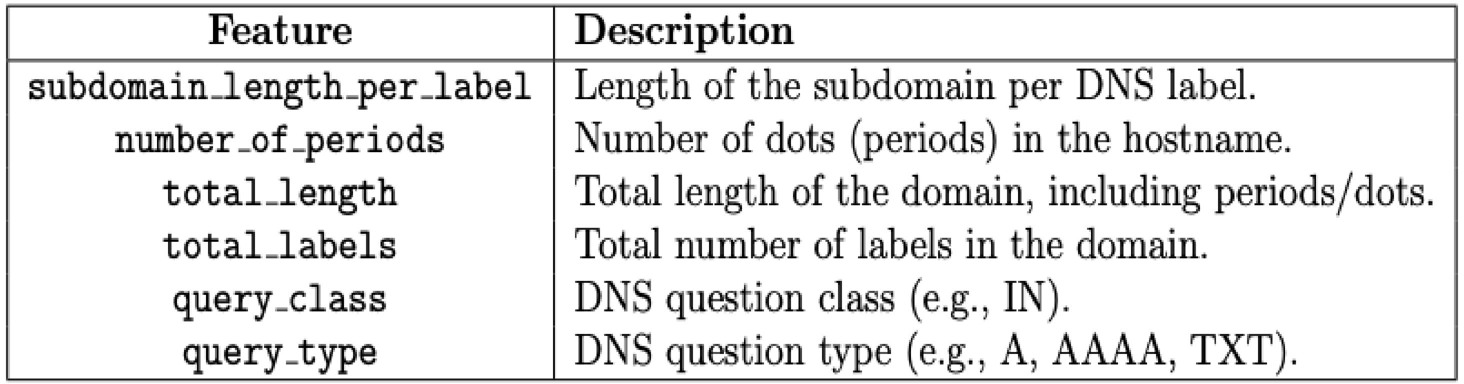
Value: Redirect count

Process Clone-redirect telemetry

Linux Kernel

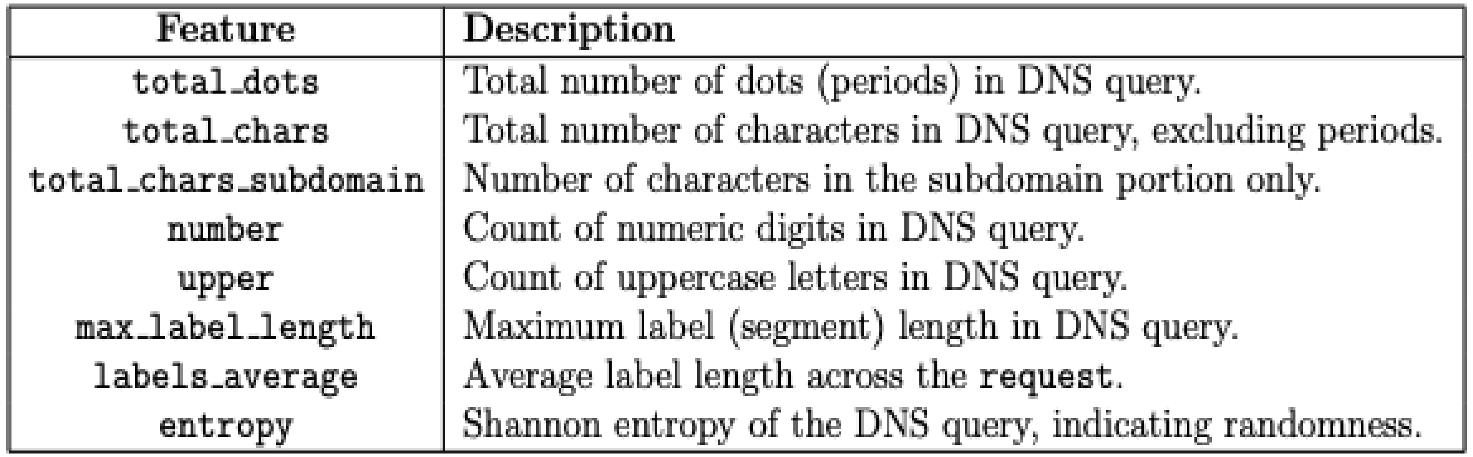
DNN based DNS Data Obfuscation Detection (Features)

- ❑ Kernel Features
- ❑ Limits for DPI in Kernel



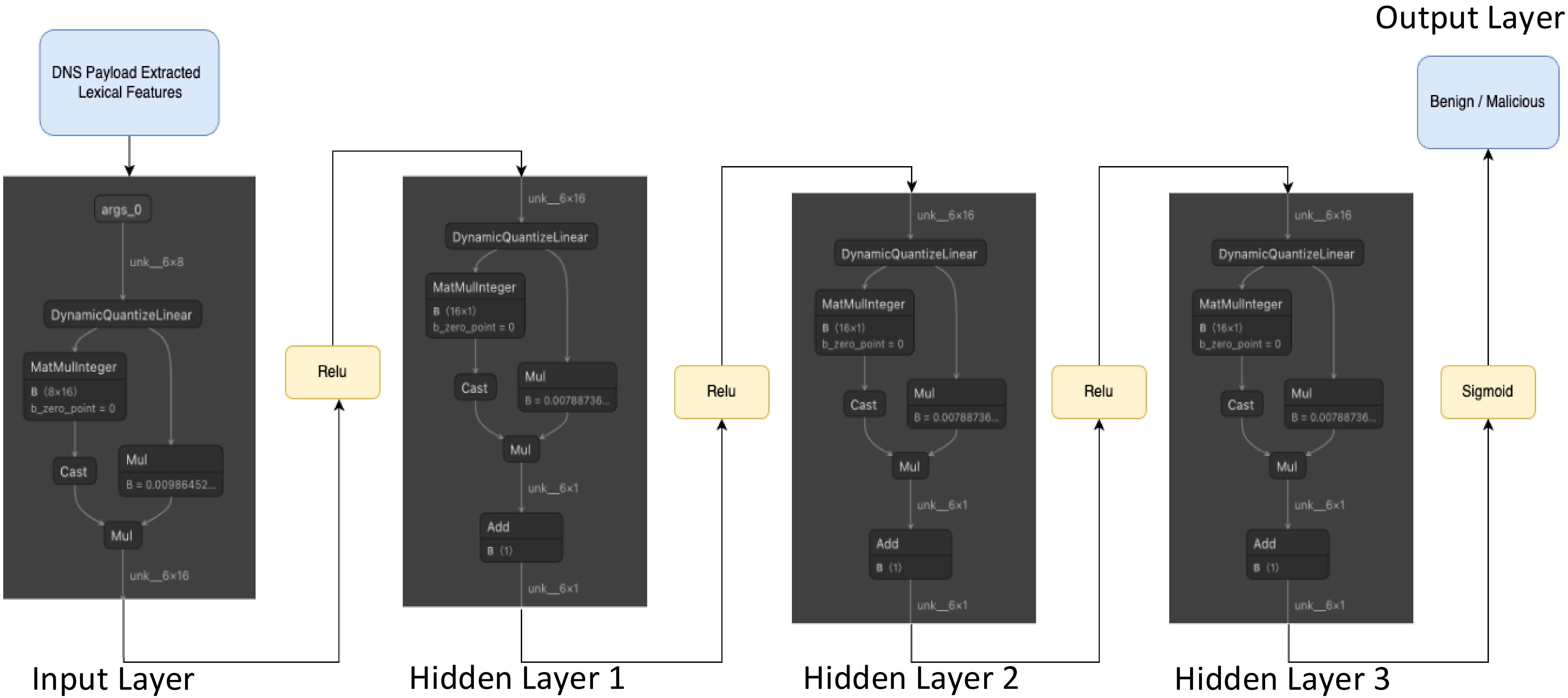
Feature	Description
subdomain_length_per_label	Length of the subdomain per DNS label.
number_of_periods	Number of dots (periods) in the hostname.
total_length	Total length of the domain, including periods/dots.
total_labels	Total number of labels in the domain.
query_class	DNS question class (e.g., IN).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

- ❑ Userspace Features
- ❑ Enhanced Lexical Features

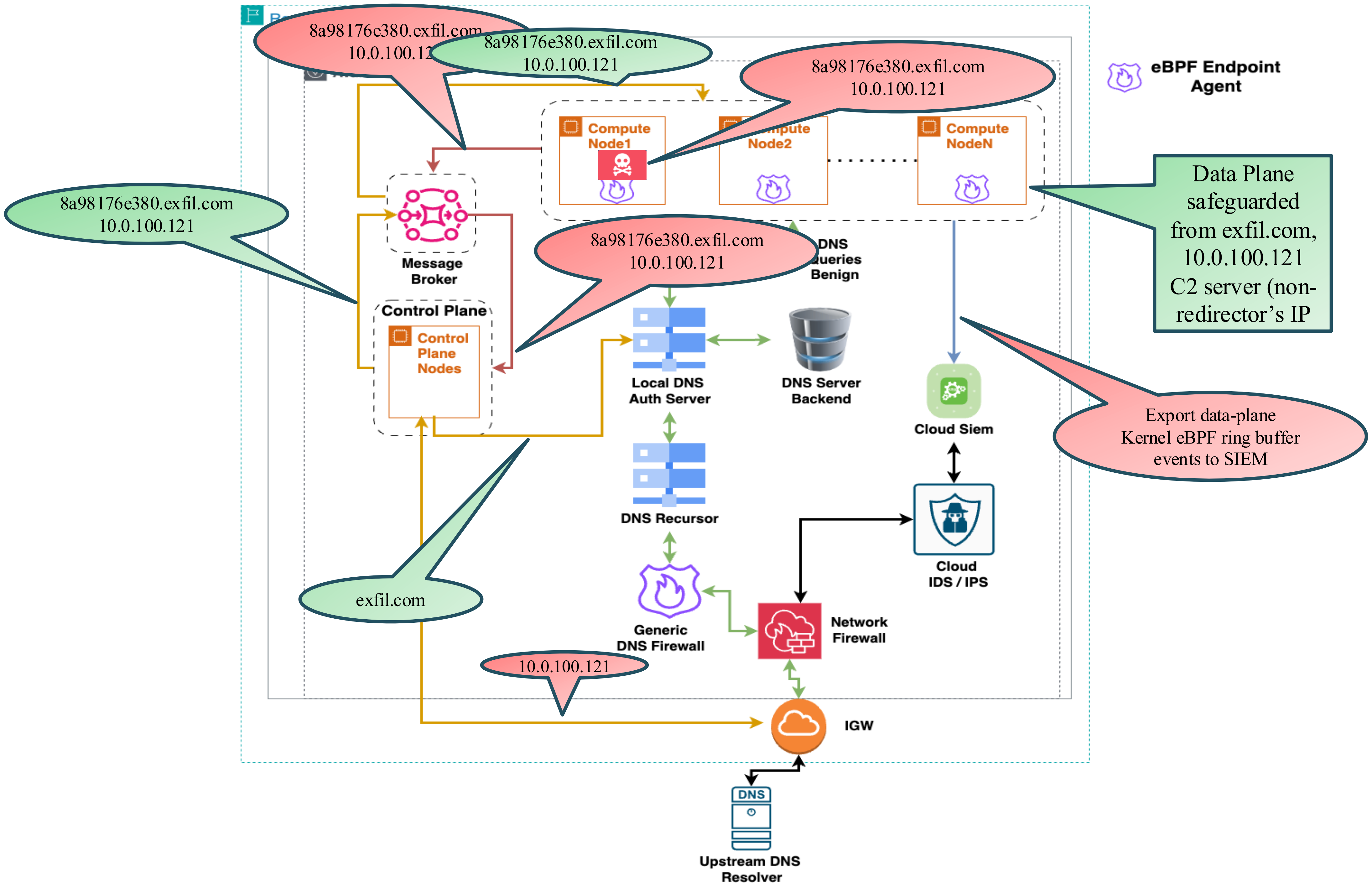


Feature	Description
total_dots	Total number of dots (periods) in DNS query.
total_chars	Total number of characters in DNS query, excluding periods.
total_chars_subdomain	Number of characters in the subdomain portion only.
number	Count of numeric digits in DNS query.
upper	Count of uppercase letters in 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 based DNS Data Obfuscation Detection Model Architecture



Framework Deployment in Cloud to combat C2 Infrastructure



8a98176e380.exfil.com 10.0.100.12

8a98176e380.exfil.com 10.0.100.121

8a98176e380.exfil.com 10.0.100.121

8a98176e380.exfil.com 10.0.100.121

8a98176e380.exfil.com 10.0.100.121

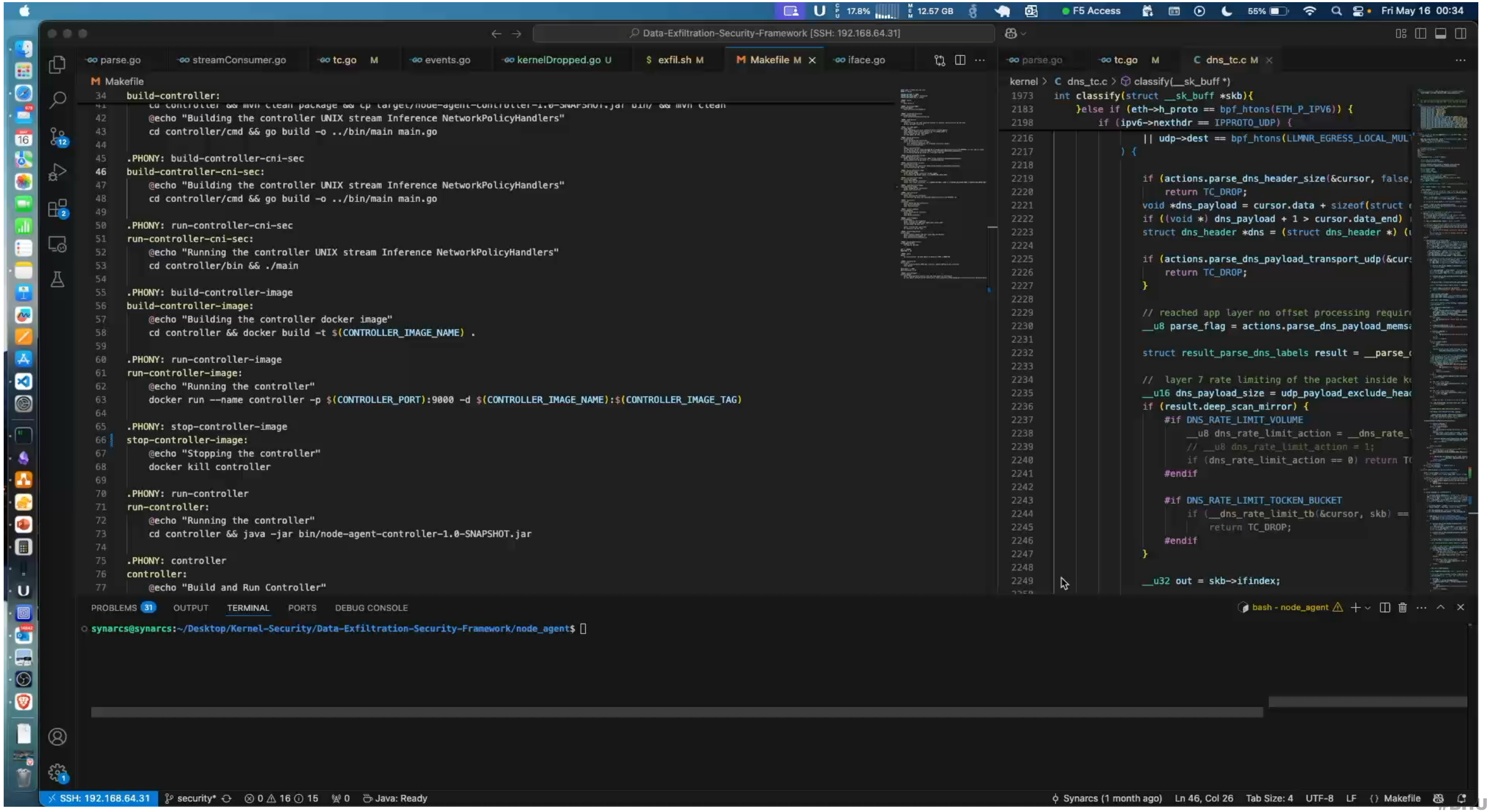
Data Plane safeguarded from exfil.com, 10.0.100.121 C2 server (non-redirector's IP

Export data-plane Kernel eBPF ring buffer events to SIEM

exfil.com

10.0.100.121

Demo

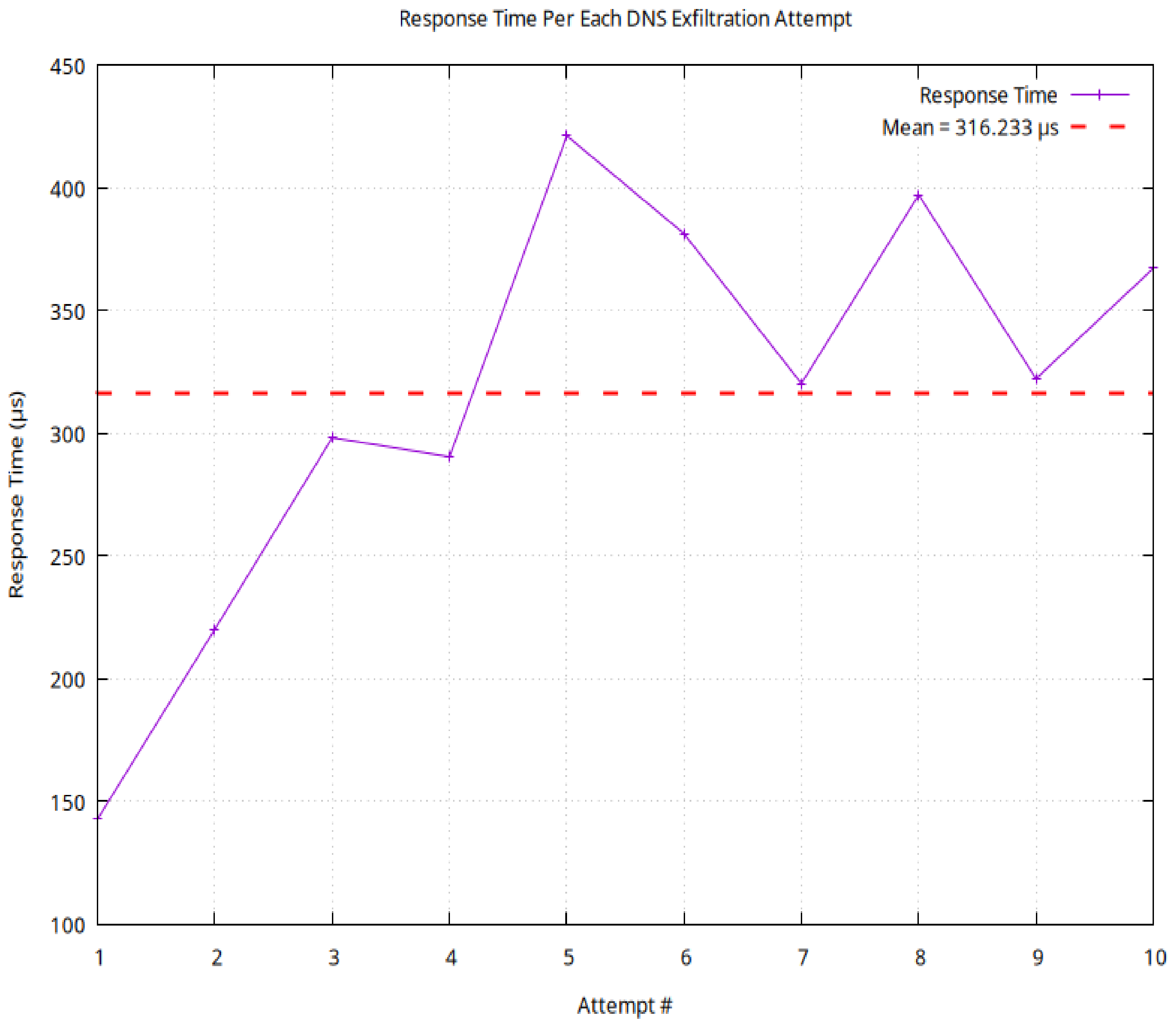


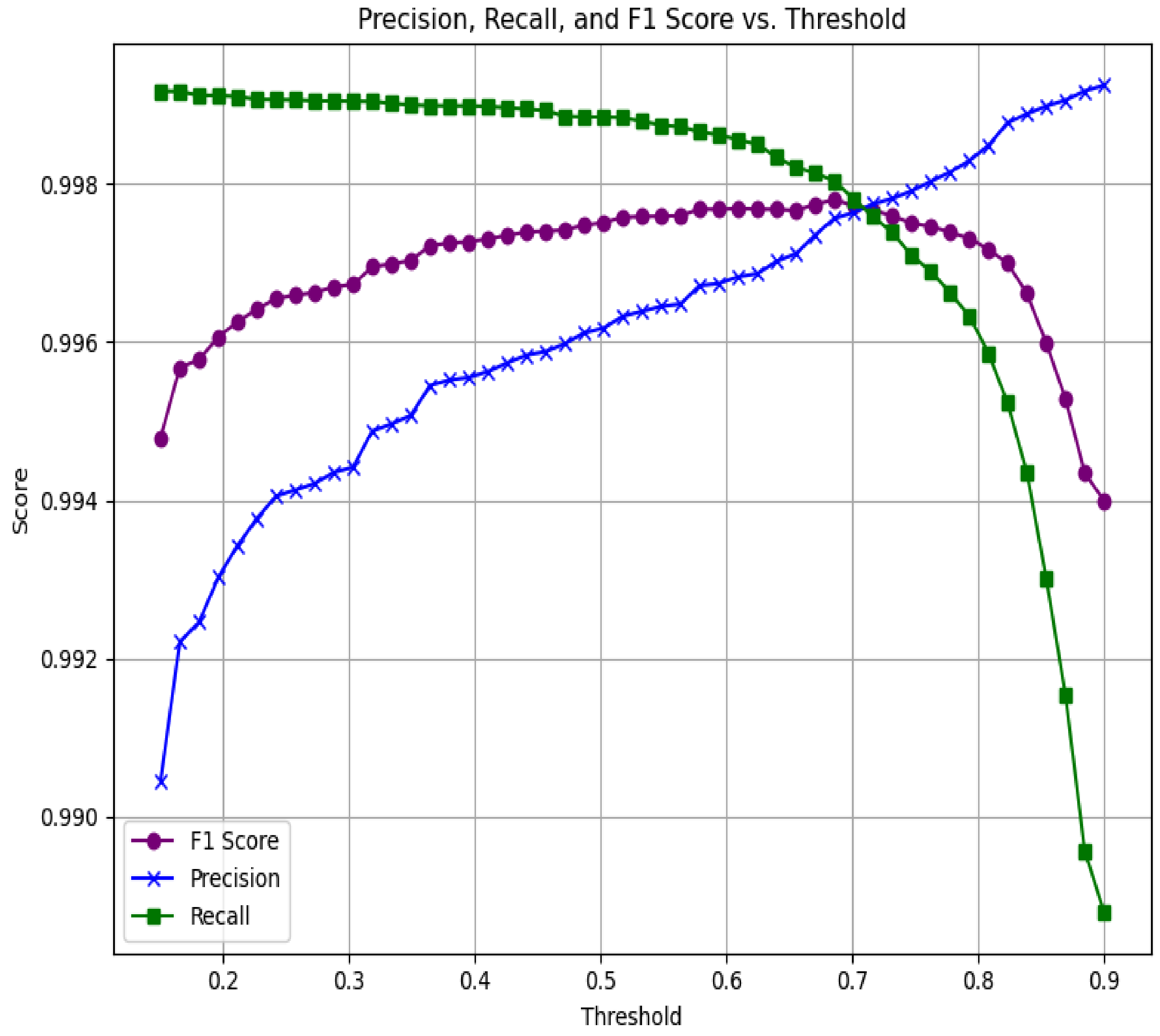
```makefile
build-controller:
    cd controller && mvn clean package && cp target/node-agent-controller-1.0-SNAPSHOT.jar bin/ && mvn clean
    @echo "Building the controller UNIX stream Inference NetworkPolicyHandlers"
    cd controller/cmd && go build -o ../bin/main main.go

.PHONY: build-controller-cni-sec
build-controller-cni-sec:
    @echo "Building the controller UNIX stream Inference NetworkPolicyHandlers"
    cd controller/cmd && go build -o ../bin/main main.go

.PHONY: run-controller-cni-sec
run-controller-cni-sec:
    @echo "Running the controller UNIX stream Inference NetworkPolicyHandlers"
    cd controller/bin && ./main

.PHONY: build-controller-image
build-controller-image:
    @echo "Building the controller docker image"
    cd controller && docker build -t $(CONTROLLER_IMAGE_NAME) .

.PHONY: run-controller-image
run-controller-image:
    @echo "Running the controller"
    docker run --name controller -p $(CONTROLLER_PORT):9000 -d $(CONTROLLER_IMAGE_NAME):$(CONTROLLER_IMAGE_TAG)

.PHONY: stop-controller-image
stop-controller-image:
    @echo "Stopping the controller"
    docker kill controller

.PHONY: run-controller
run-controller:
    @echo "Running the controller"
    cd controller && java -jar bin/node-agent-controller-1.0-SNAPSHOT.jar

.PHONY: controller
controller:
    @echo "Build and Run Controller"
```

```c
int classify(struct __sk_buff *skb){
    }else if (eth->h_proto == bpf_htons(ETH_P_IPV6)) {
        if (ipv6->nexthdr == IPPROTO_UDP) {
                    || udp->dest == bpf_htons(LLMNR_EGRESS_LOCAL_MUL
                ) {

                    if (actions.parse_dns_header_size(&cursor, false,
                        return TC_DROP;
                    void *dns_payload = cursor.data + sizeof(struct 
                    if ((void *) dns_payload + 1 > cursor.data_end) 
                    struct dns_header *dns = (struct dns_header *) (

                    if (actions.parse_dns_payload_transport_udp(&curs
                        return TC_DROP;
                    }

                    // reached app layer no offset processing requir
                    __u8 parse_flag = actions.parse_dns_payload_memsa

                    struct result_parse_dns_labels result = __parse_

                    //  layer 7 rate limiting of the packet inside k
                    __u16 dns_payload_size = udp_payload_exclude_hea
                    if (result.deep_scan_mirror) {
                        #if DNS_RATE_LIMIT_VOLUME
                            __u8 dns_rate_limit_action = __dns_rate_
                            // __u8 dns_rate_limit_action = 1;
                            if (dns_rate_limit_action == 0) return T
                        #endif

                        #if DNS_RATE_LIMIT_TOCKEN_BUCKET
                            if (__dns_rate_limit_tb(&cursor, skb) ==
                                return TC_DROP;
                        #endif
                    }

                    __u32 out = skb->ifindex;
```

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

Response Speed with Precision





Next Steps

- ❑ **Support for DNS-over-TCP:** Similar eBPF DPI and endpoint agent design for TCP
- ❑ **Kernel TLS Fingerprinting and Encrypted Tunnels**: eBPF for TLS fingerprinting(uprobes / KTLS) to detect, hunt kill DNS, HTTPS exfiltration over TLS.
- ❑ **Advanced Intelligence, process correlation:** eBPF kernel program and endpoint agent cross-protocol exfiltration attempt tied to prevented process.
- ❑ **eBPF Endpoint Agent a built-in guard for DNS NXDOMAIN flood at endpoint.**
- ❑ **AI-Driven Model Evolution**: Real-time drift detection, online learning, and confidence-based updates ensure precision against emerging DNS obfuscation tactics.

Black Hat Sound Bytes

- **Real-Time Kernel Threat Hunting & EDR Boost**: Hunt C2 implants dynamically in-kernel, accelerating user-space EDR with precise signals to stop C2 and breaches.
- **AI-Driven Kernel Enforcement**: Pair AI with eBPF to adaptively reprogram the kernel for intelligent, real-time threat blocking.
- **Dynamic Kernel & Cloud Firewalling**: 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 feeds rich behavioral signals into upstream SIEM, SOAR.

Thank You

Code: https://github.com/Synarcs/DNSObelisk

WhitePaper: https://github.com/Synarcs/DNSObelisk_Report