black hat BRIEFINGS

AUGUST 6-7, 2025

MANDALAY BAY / LAS VEGAS

From Packet to Process: Hunting and Killing DNS C2 inside Kernel using eBPF

Vedang Parasnis (Synarcs)

University of Washington

Agenda

- DNS a critical backdoor for enterprise networks
- DNS Exfiltration Attack Vectors
- DNS C2 Attack Infrastructure
- Existing Approaches
- Limitations
- AI-Driven Linux Kernel Enforced Endpoint Security
- Cloud Deployment Architecture for scale and to combat C2 infrastructures
- Demo
- Key Takeaways & Future Directions
- Q&A

They Breach Through DNS — Every Time

Compromise National Defense

DNS C2 in SolarWinds enabled deep, undetected federal access

Cloud & Hyperscaler's Breached

DNS tunneling let attackers persist across tenant boundaries

Critical Infrastructure Infiltrated

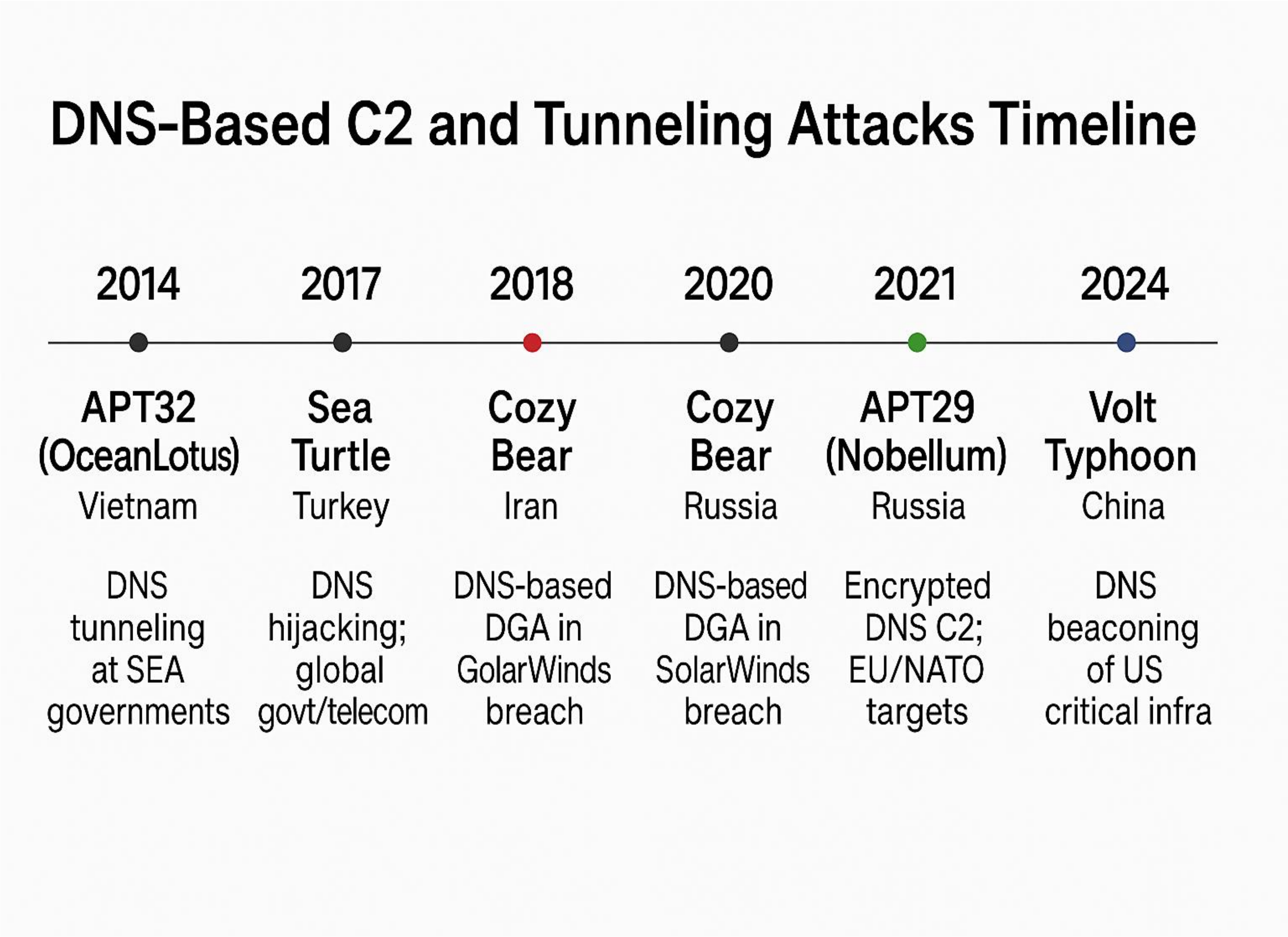
Volt-Typhoon used DNS beaconing in power and telecom networks

Mass Credential Theft

DNS hijacks enabled widescale credential harvesting

Same Tools, Same Abuse

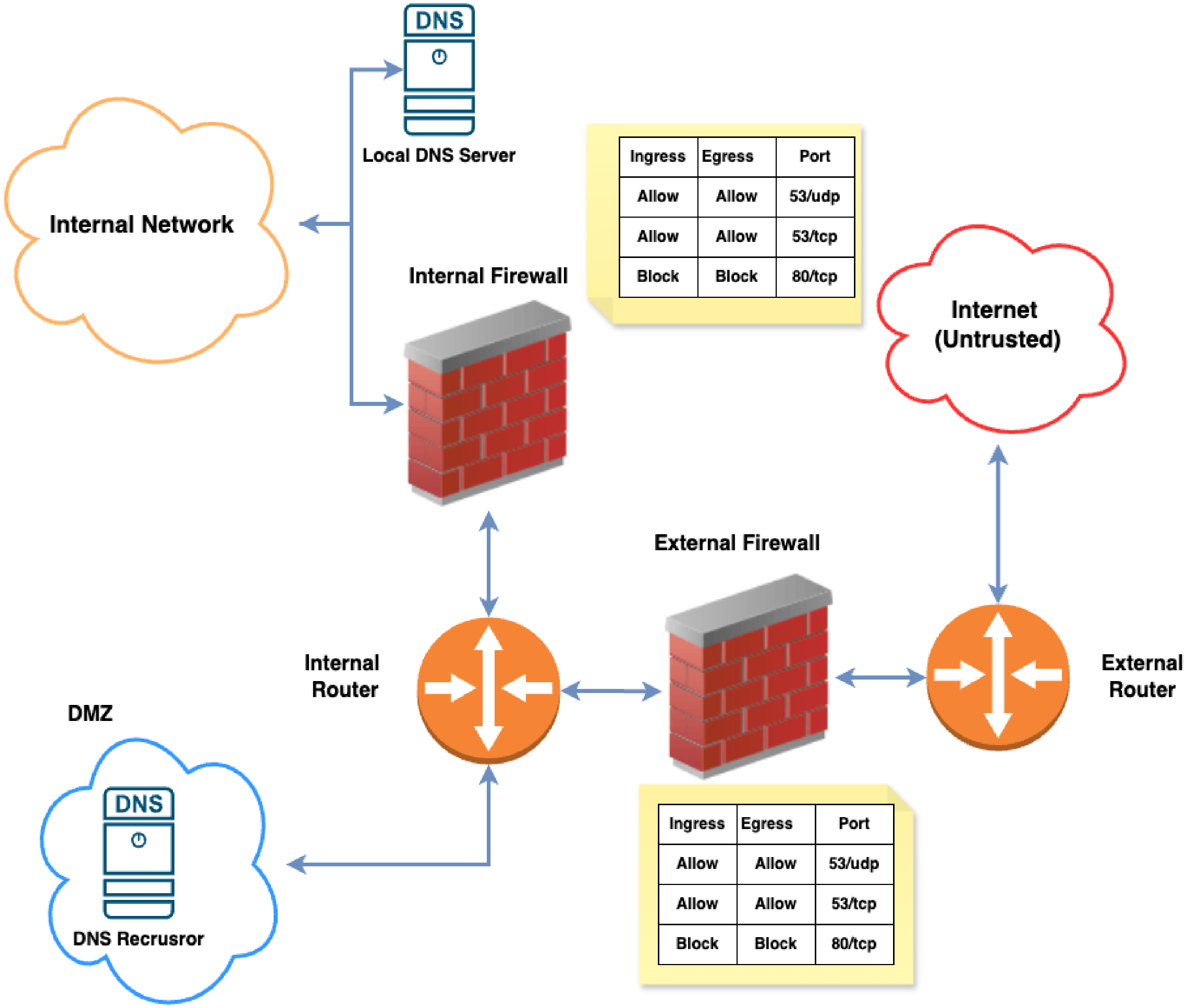
Sliver, DNSCat2, and Cobalt Strike power both red teams and APTs



90%+ of APT's employ DNS for C2 and data breaches

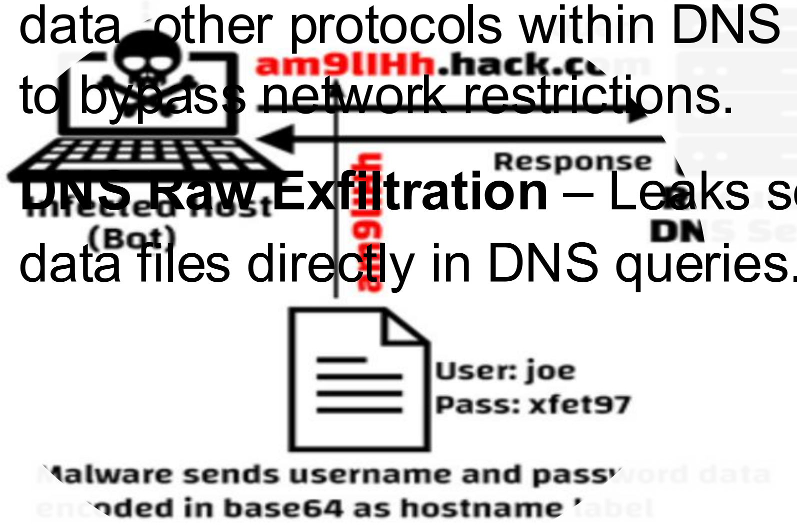
DNS a Blind spot to compromise networks

- **Unencrypted by Default:** Attackers hide payloads in plain sight
- **Rarely Monitored Deeply:** DNS logs are ignored, giving a free channel
- **Firewall Blindspot:** DNS Port stays open, bypassing defenses
- **Stateless Protocol:** No handshake = easy to spoof, replay, and operate from throwaway attack infrastructure.



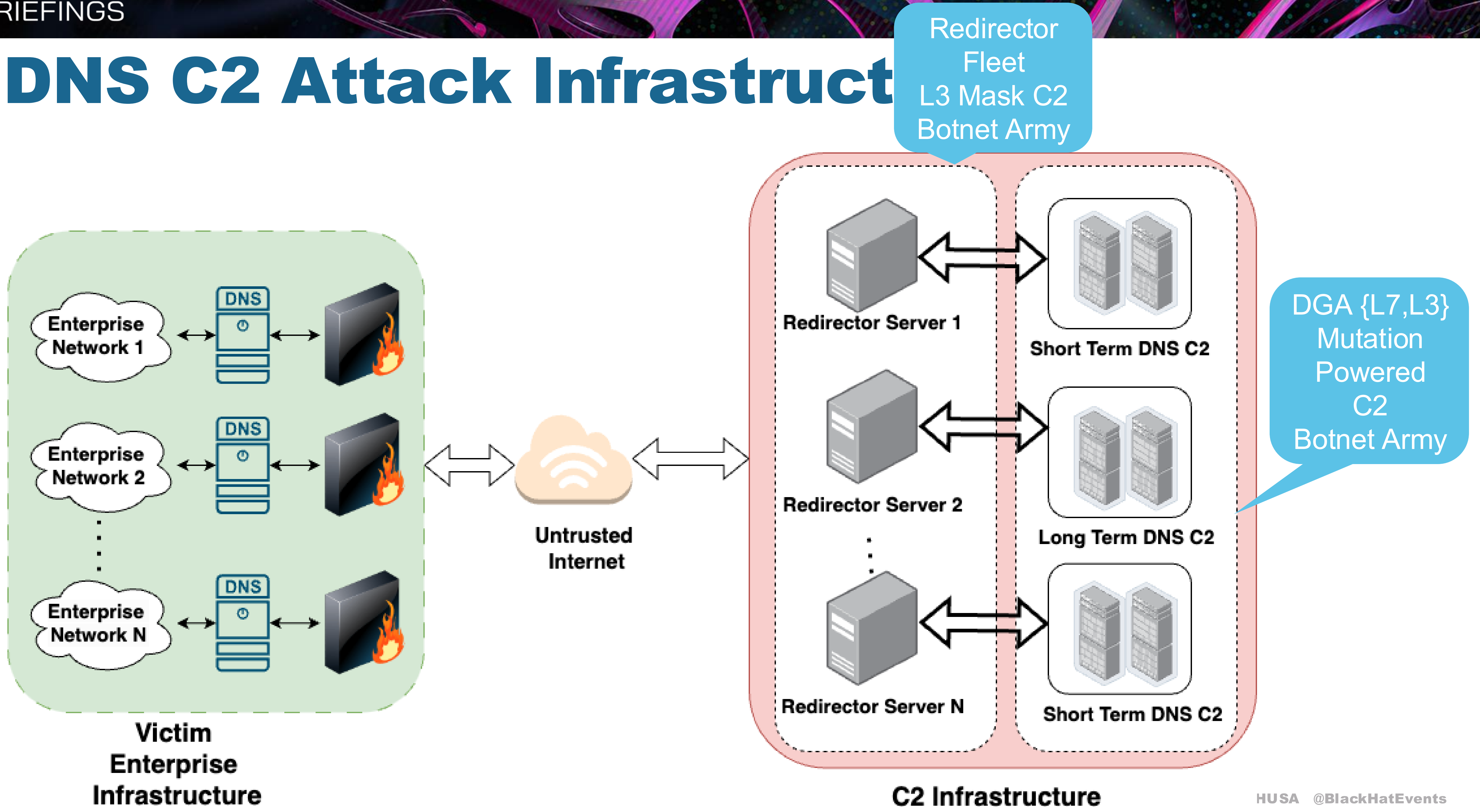
DNS: Not Just For Name Resolution Anymore. Next channel deliver zero-day attacks.

- **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.



- **Remote Code Execution (RCE)**
  - Shell code exploits
  - Script executions, File corruptions
  - Process Side channeling exploits
  - Example: **Sliver C2, Hexane, APT29 (Cozy Bear), Skitnet.**
- **Persistent Backdoors**
  - Deployment rootkits, ransomwares
  - Example: **Turla** group
- **Network Pivoting (Port Forwarding)**
  - Compromised machines act as proxies to reach deeper into private infrastructure
  - Example: **Cobalt Strike, Hexane, DNSSystem**

DNS C2 Attack Infrastructure



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 IOCs** – 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
bhack-1.com
bhack2.com

Wordlist DGAs

Wordlist dictionary
catsun.net
reddog.org

Character-Based or Randomized DGAs

Pseudo random chars
sdas232.bleed.io

Challenges in Real-Time Disruption from C2 Infrastructure over DNS

EVOLVING SCALE OF C2 INFRASTRUCTU RE

INCREASED COMPLEXITY

FOR REAL-TIME PREVENTION.

DIFFICULT ACHIEVE GOAL OF ZERO DATA LOSS AND C2

COMMUNICATION.

NEED ACCURATE

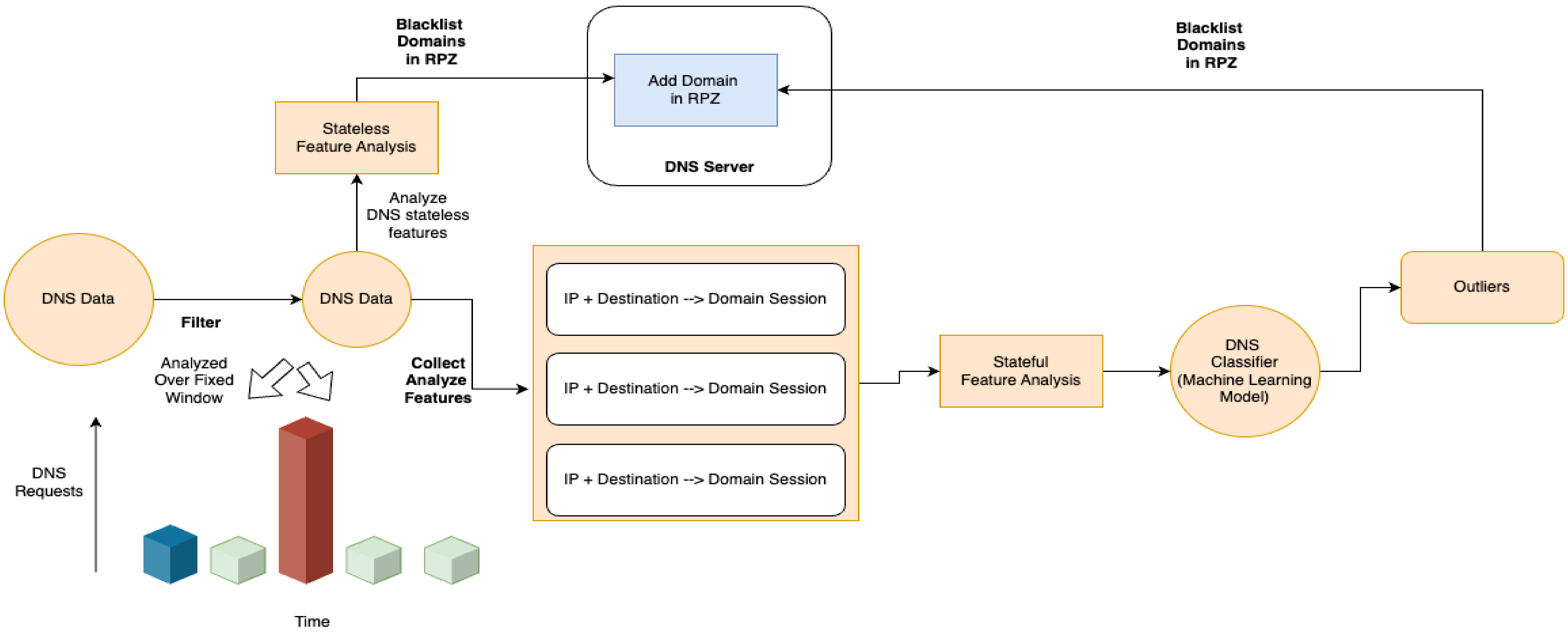
AND FAST TERMINATION OF THREATS.

HIGHLY STEALTHY AND MUTATIVE

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



Issues with current approaches

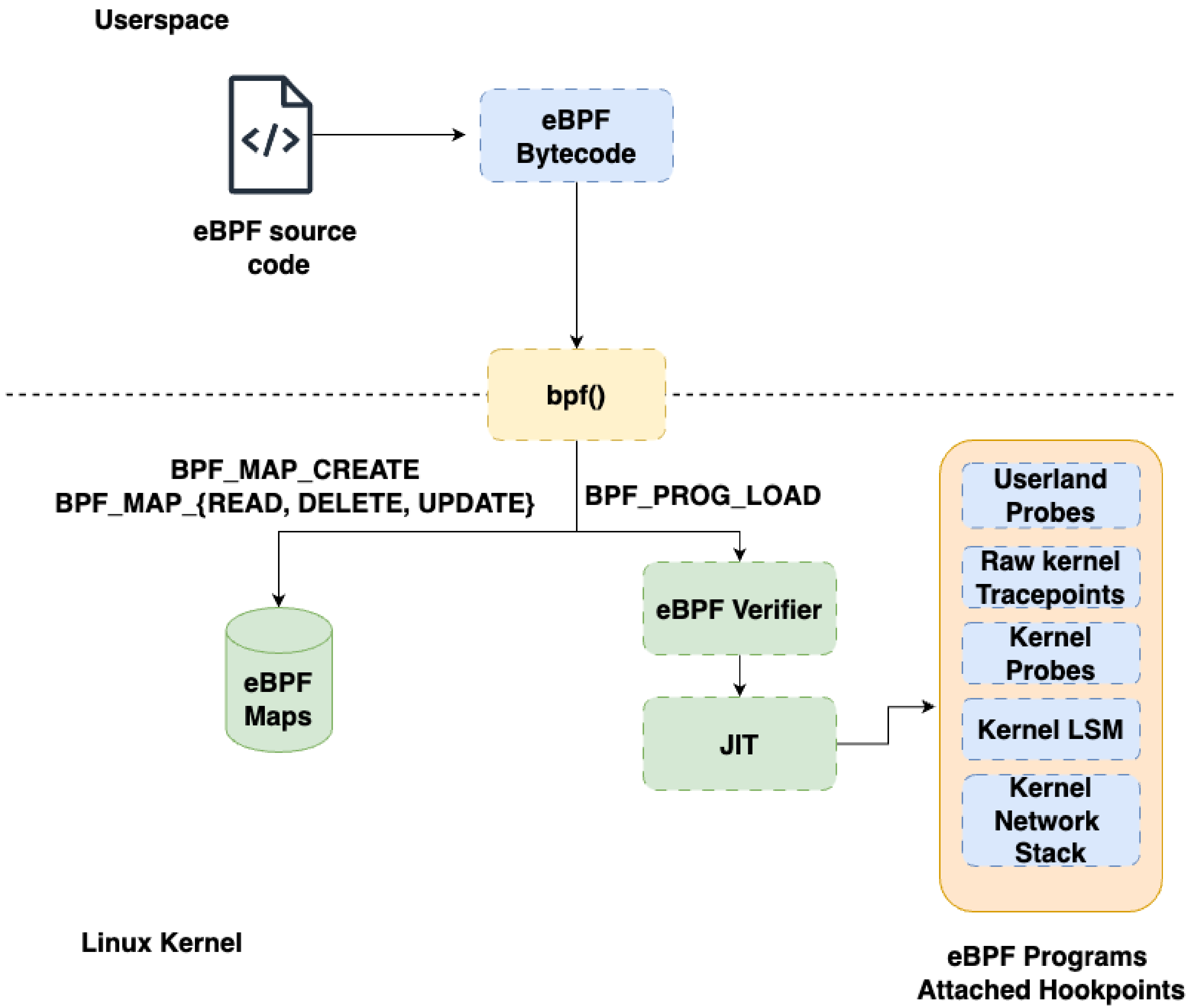
- ❑ **Slow Detection → Slow Response → High Dwell Time → More Damage**
- ❑ **Slow and easy bypass to Advanced C2 Attacks:** C2 infrastructure employing multiplayer mode (C2 Botnet Army)
- ❑ **Don't fully protect against Domain Generation Algorithms, IP mutation**
- ❑ **Unwanted latency for proxy-based DPI on benign traffic**
- ❑ **No Assurance for zero data loss or no command execution.**
- ❑ **Dynamic Threat Patterns**:
  - ❑ Varying Throughput, encryption, encodings
  - ❑ Varying DNS Payload Types (MX, TXT, NULL, AAAA)
  - ❑ Port Obfuscation

Solution:

Run EDR inside Kernel reactively (RING-0) closest to wire where no userland evasion can hide rather being proactive over DNS traffic patterns

eBPF

- Reprogram the Linux kernel in safe way
- Safe, modern way to write kernel modules
- Runs BPF virtual machine inside kernel
- Custom BPF bytecode
- Uses 512 bytes of stack
- eBPF Maps as heap
- CPU architecture agnostic, Linux kernel version agnostic (BTF)



DNS Protocol Specifications (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

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
                     Length                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

DNS Header for TCP

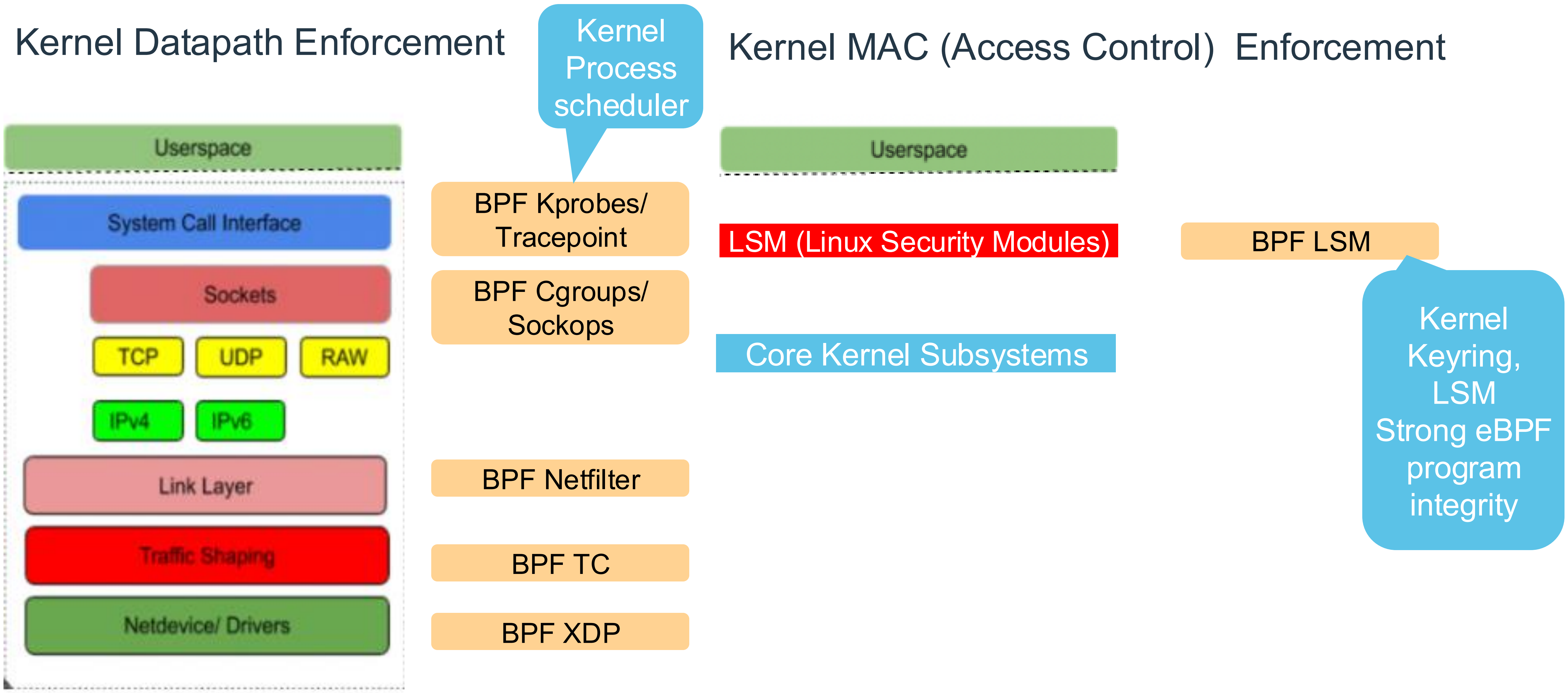
```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

DNS Header for UDP

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

DNS Application Payload

EDR Agent Linux Kernel eBPF Hooks



Kernel Enforced Endpoint Security for DNS

Agent based Endpoint Security

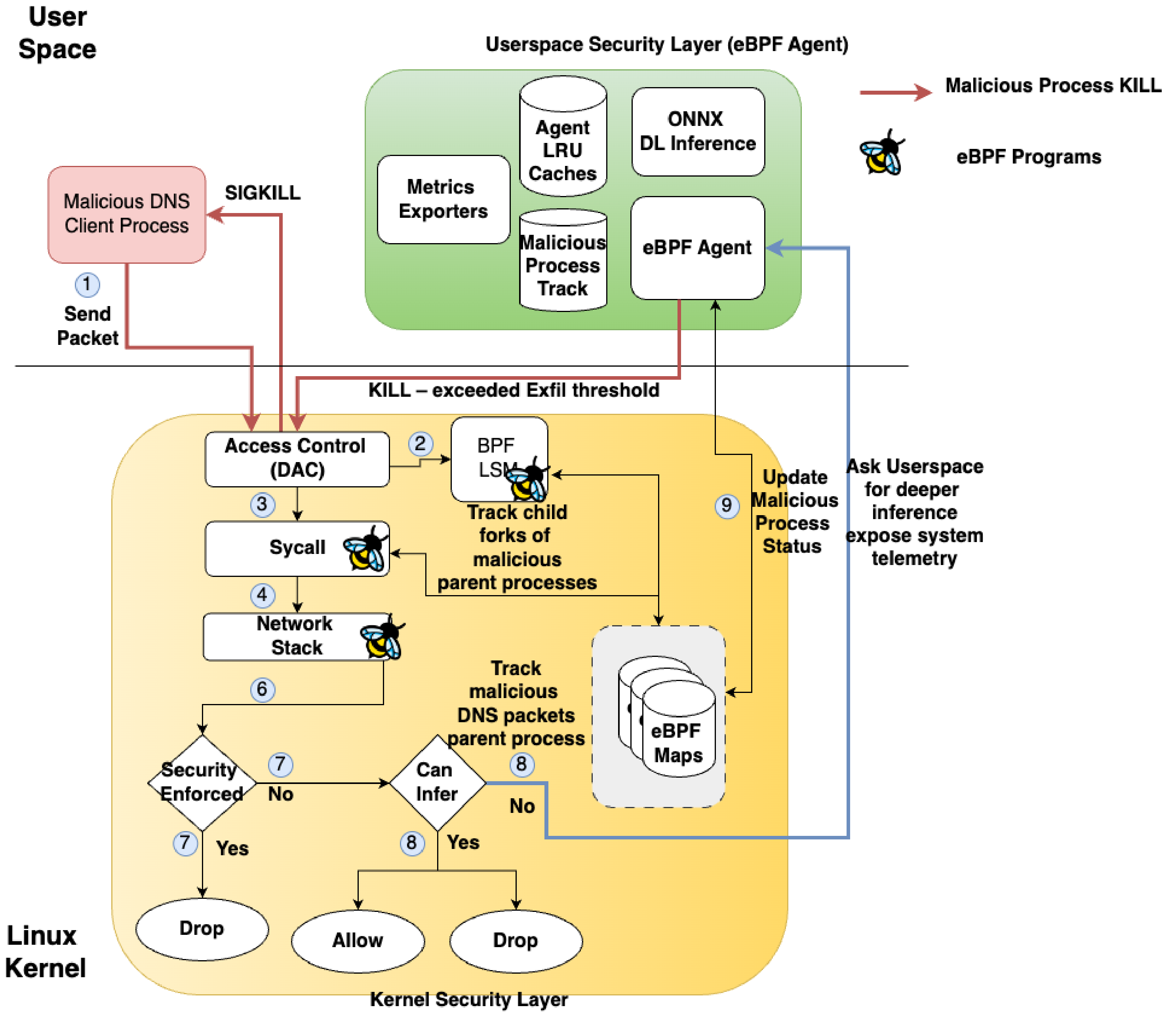
Continuous Security Enforcement Event Loop

Userspace

- eBPF Agent
- eBPF Agent LRU Caches
- ONNX Quantized Deep Learning Model
- Kernel malicious metrics exporters (Prometheus)

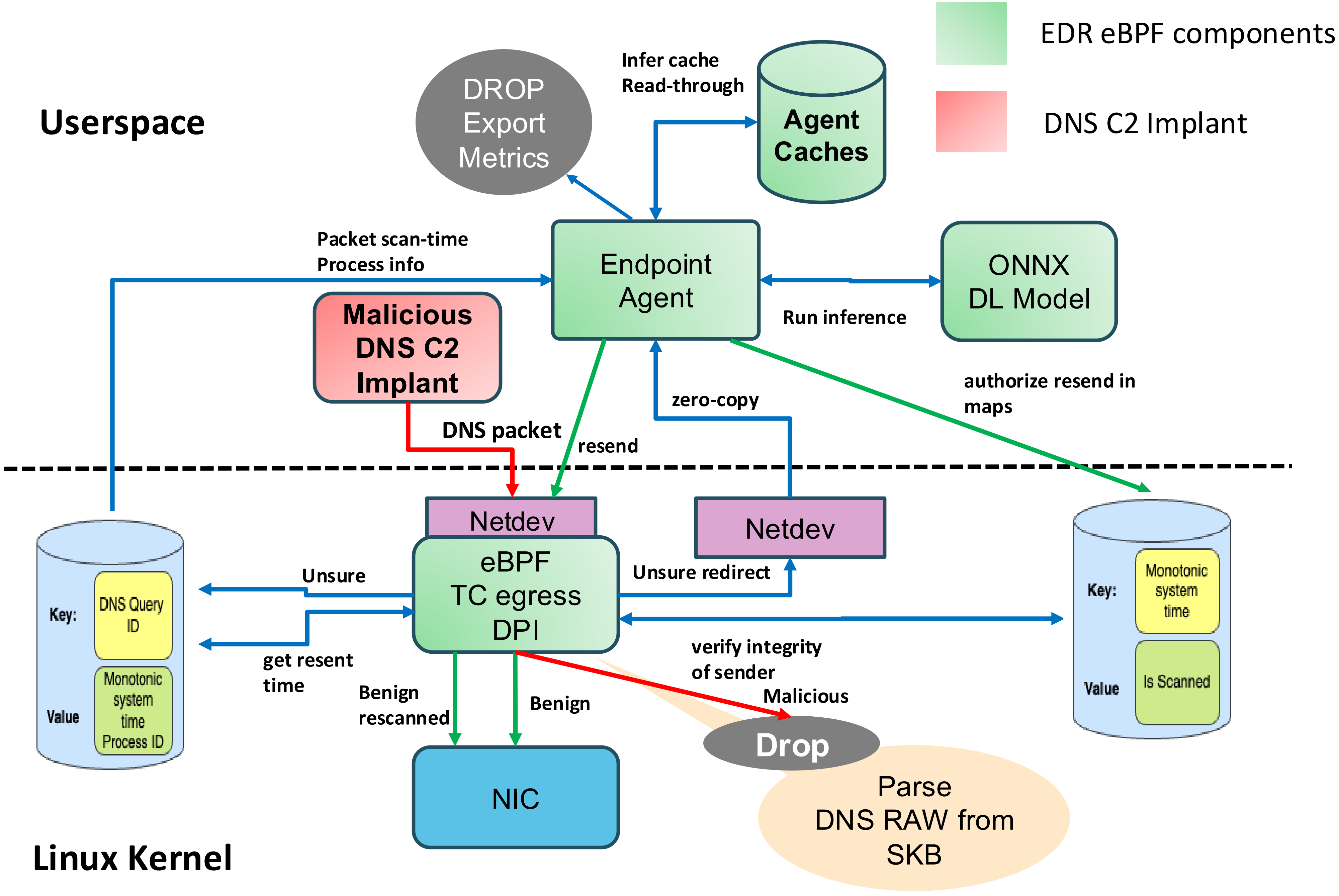
Linux Kernel

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

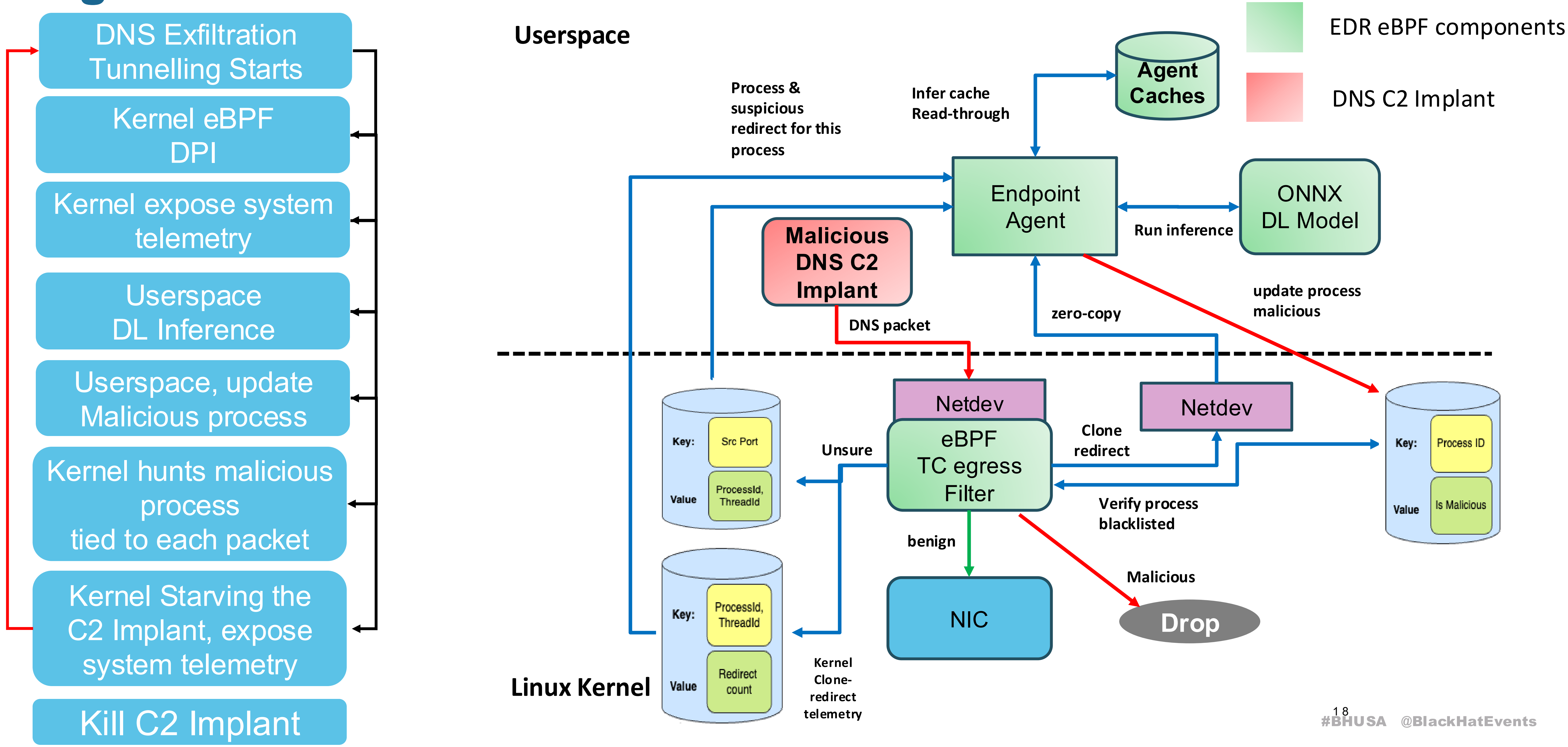


Egress Active Malicious Process Strict Security Enforcement

- DNS Exfiltration Tunnelling Starts
- Kernel eBPF DPI
- Kernel expose system telemetry
- Userspace DL Inference
- Userspace track Each process malicious activity
- Kernel starving the C2 implant process, system telemetry
- Kill C2 Implant



Egress Passive Malicious Process Threat-Hunt Enforcement



black hat BRIEFINGS

What Makes DNS contain C2 commands or exfiltrated data

High Entropy QNAME: Random encoding / encrypted, binary payloads

Excessive Label Count or Length Long chains of subdomains to chunk exfil data

Non-Dictionary Tokens No real words — resembles encoded data, not legit words in subdomains

Time-based or Patterned Generation DGA-style domain structure — predictable but meaningless

Out-of-Order or Sparse TTL Behavior DNS queries with abnormal TTLs used for signaling or state

•**Rare NXDOMAIN Frequency or "Ghost" Domains** C2 testing infrastructure — sends data to non-existent domains, no resolution needed

black hat BRIEFINGS

DNN based DNS Data Obfuscation Detection (Features)

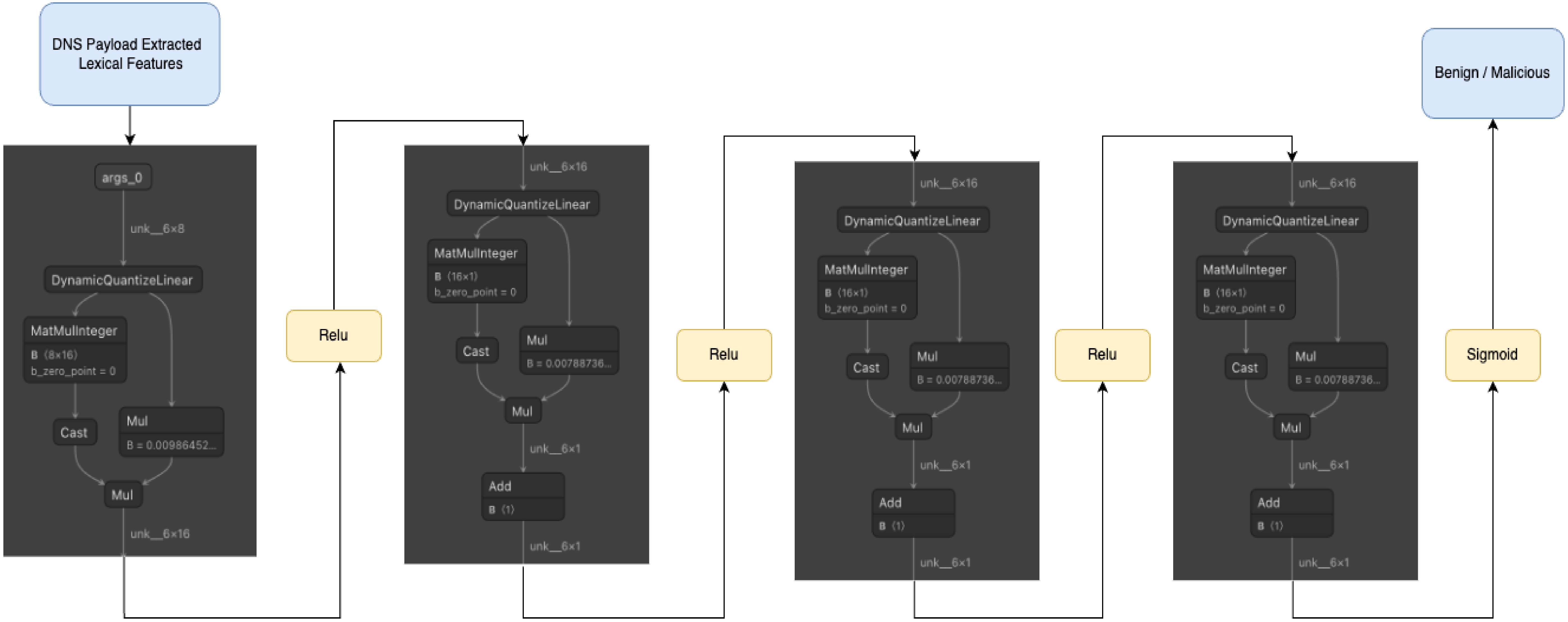
- ❑ Kernel Features
- ❑ Limits for DPI in Kernel

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 Model Lexical Features

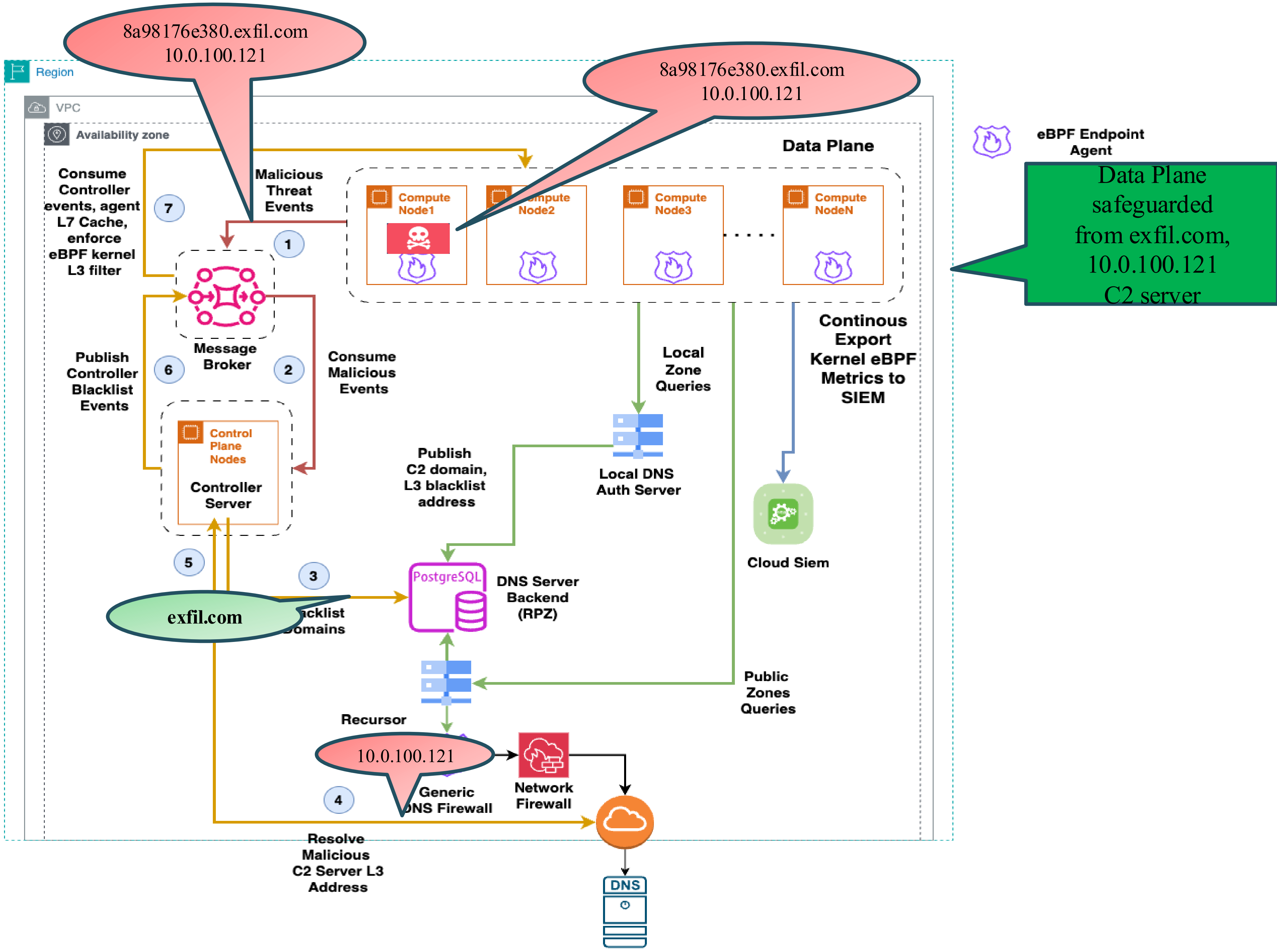
Feature	Description
total_dots	Total number of dots (periods) in DNS query.
total_chars	Total number of characters in DNS query, excluding periods.
total_chars_subdomain	Number of characters in the subdomain portion only.
number	Count of numeric digits in DNS query.
upper	Count of uppercase letters in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

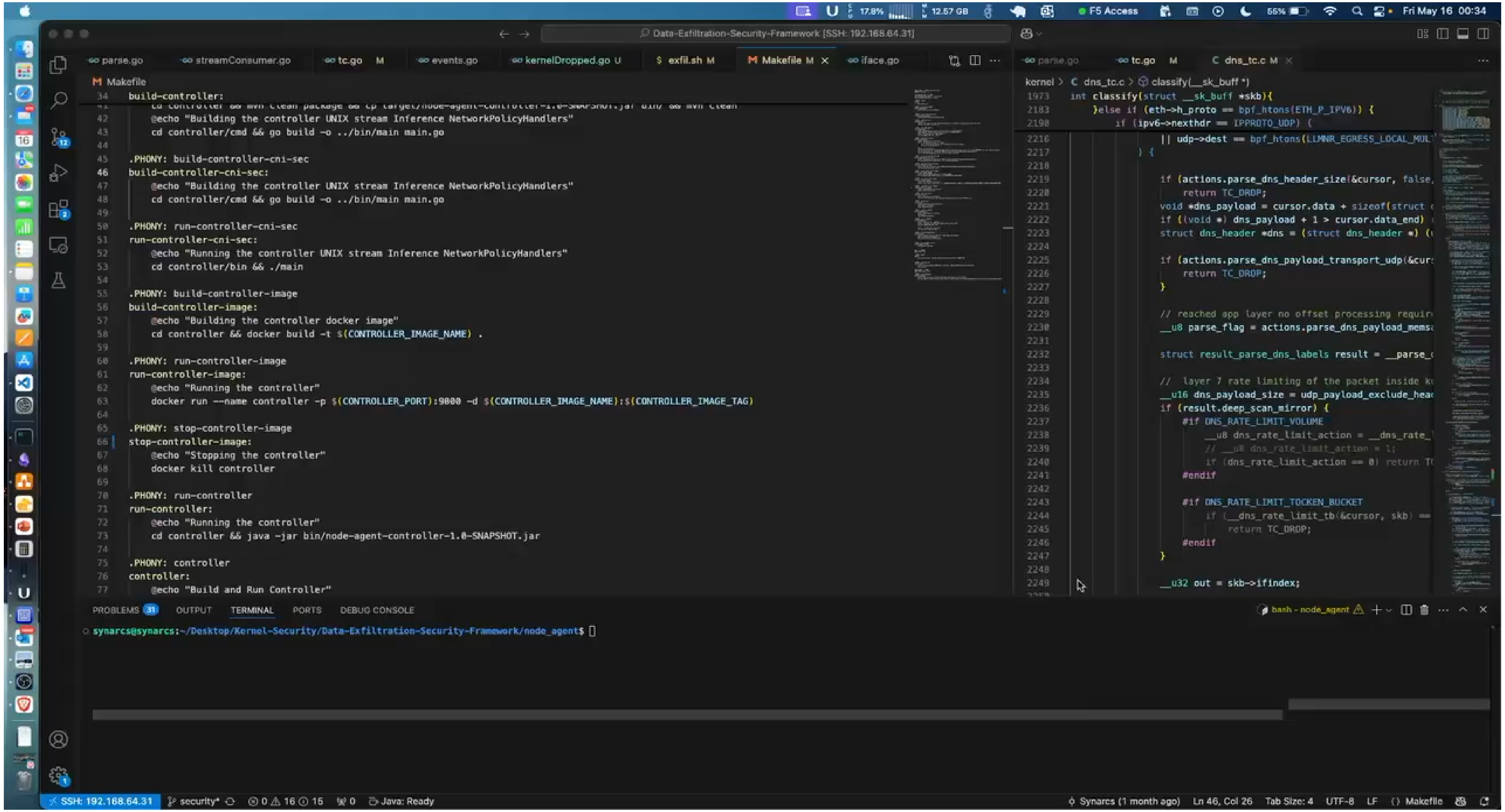


Input Layer   Hidden Layer 1   Hidden Layer 2   Output Layer

Scalable Framework Deployment to combat C2 Attack Infrastructure

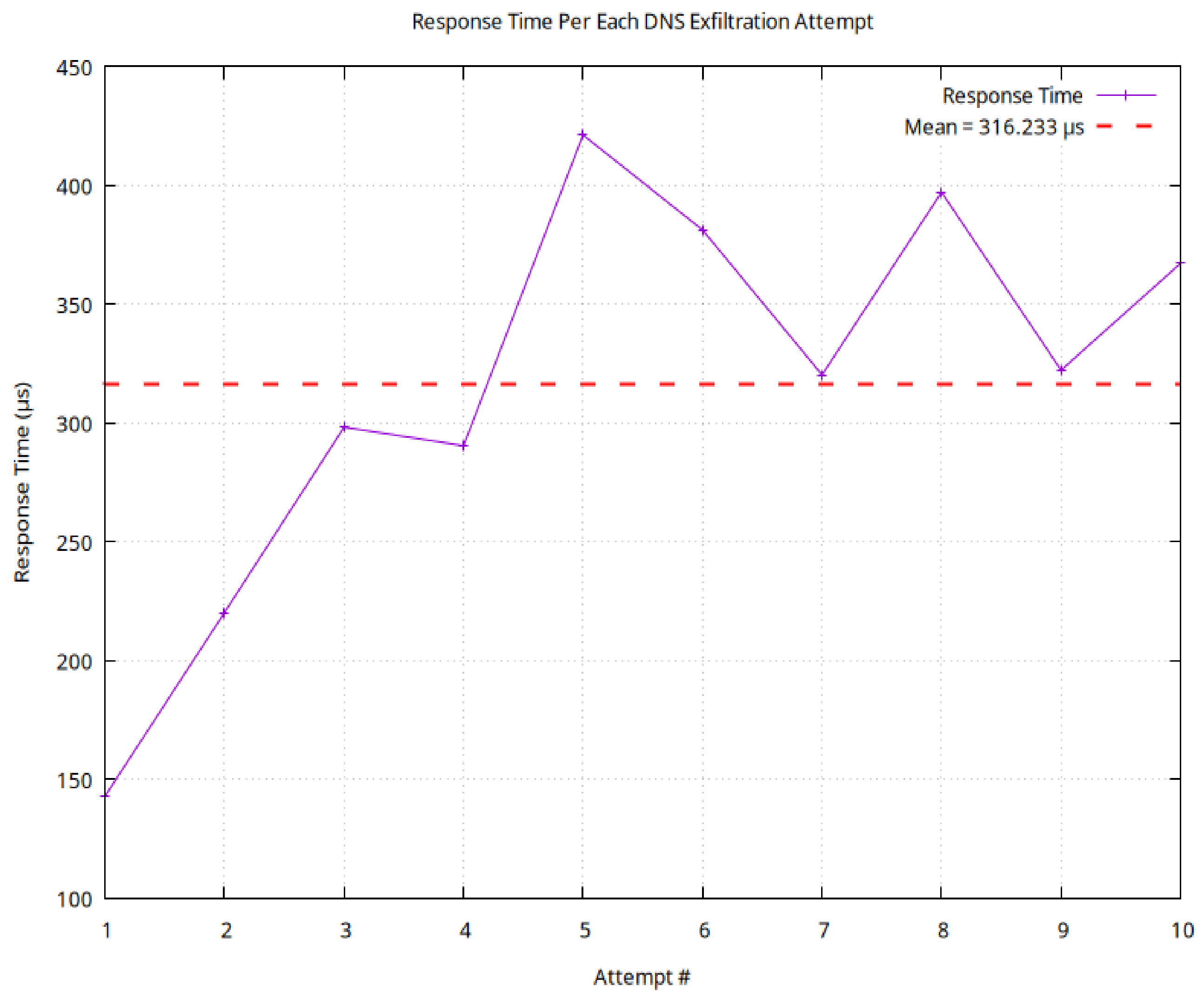


Demo


Fri May 16 00:34
Data-Exfiltration-Security-Framework [SSH: 192.168.64.31]
parse.go
streamConsumer.go
tc.go
events.go
kernelDropped.go
exfil.sh
Makefile
iface.go
dns_tc.c
Makefile
build-controller:
@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"
kernel > dns_tc.c > classify(__sk_buff *)
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(&cur
return TC_DROP;
// reached app layer no offset processing requir
__u8 parse_flag = actions.parse_dns_payload_mems
struct result_parse_dns_labels result = __parse_
// layer 7 rate limiting of the packet inside k
__u16 dns_payload_size = udp_payload_exclude_head
if (result.deep_scan_mirror) {
#if DNS_RATE_LIMIT_VOLUME
__u8 dns_rate_limit_action = __dns_rate_
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;
PROBLEMS
OUTPUT
TERMINAL
PORTS
DEBUG CONSOLE
synarcs@synarcs:~/Desktop/Kernel-Security/Data-Exfiltration-Security-Framework/node_agent$
SSH: 192.168.64.31
security*
Java: Ready
Synarcs (1 month ago)
Ln 46, Col 26
Tab Size: 4
UTF-8
LF
Makefile

Summary

- **Disrupt DNS covert C2 channel attacks, data exfiltration:** Deep packet inspection and enforcement inside kernel via eBPF to block all forms of DNS exfiltration channels.
- **AI-Assisted Threat Detection:** Deep learning in userspace to detect advanced obfuscated exfiltration payloads with high accuracy aiding kernel network enforcements.
- **Malicious Process Aware Active Response (Threat-Hunt and Kill):** Link exfiltration attempt to parent process and kill implants processes, preventing lateral movement and further damage.
- **Dynamic Cross-Layer Policy Enforcement:** Enforce in-kernel L3 network policies adaptively and domain blacklisting on DNS server, L3 firewall filters to combat DGA



black hat BRIEFINGS

Next Steps

- **Support for DNS-over-TCP:** Implement in-kernel eBPF-based detection for DNS-over-TCP replicating TCP state machine over kernel socket layer, paired with userspace DPI via Envoy proxy.
- **Kernel TLS Fingerprinting and Encrypted Tunnels**: Use eBPF for TLS fingerprinting(uprobes / KTLS) to detect DNS, HTTPS exfiltration over TLS (DOH), DNS over TLS, WireGuard.
- **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**
- **Controller driven continuous Model Evolution:** Drift detection, online learning, and confidence-based live updates to maintain precision against emerging DNS obfuscation tactics.
- **Dynamically reprogram Endpoint Agents**

Takeaways

- **eBPF driven endpoint security:** Stop data breaches & C2 implants exploiting DNS dynamically, in real-time, directly within the kernel using eBPF.
- **Real-time Kernel Threat Hunting & EDR Acceleration:** Achieve dynamic, in-kernel C2 malicious implant hunting; dramatically boosting user-space EDR speed and precision.
- **AI-Driven Dynamic Kernel Enforcement:** Pair deep learning with eBPF for intelligent, adaptive defense dynamically reprogramming kernel
- **Dynamic Kernel, Cloud Firewalling:** Enforce adaptive network filters at endpoint inside kernel via eBPF and cloud firewalls to combat DGA and evolving C2 infrastructure attacks.
- **Unprecedented OS Telemetry for SIEM/SOAR:** eBPF-driven deep OS visibility fuels superior adversary behavior analysis and enriches upstream SIEM/SOAR deep learning models.

Thank You

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

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