FINAL REVIEW PRESENTATION

PowerPoint Slide Content (15-20 Slides)

SLIDE 1: TITLE SLIDE

FINAL REVIEW

Real-Time DDoS Mitigation System using Machine Learning and eBPF/XDP

Presented by:
[Your Name] - [Roll Number]

Guide:
[Guide Name]

Department of Computer Science and Engineering
[College Name]
Academic Year: 2025-2026

SLIDE 2: PROBLEM MOTIVATION

Why DDoS Mitigation Matters?

Real-World Impact:

- 📊 84% of organizations hit by DDoS in 2023
- 💰 Cost: $20,000-$40,000 per hour downtime
- 📈 Attack volumes: >1 Tbps recorded
- ⏱️ Average duration: 4-6 hours

Recent Attacks:

- GitHub (2018): 1.35 Tbps
- AWS (2020): 2.3 Tbps
- Google (2022): 46 million requests/second

Challenge: Traditional solutions too slow or too expensive

SLIDE 3: PROBLEM STATEMENT

The Core Challenge

Design a DDoS mitigation system that:

✅ Processes packets at **line rate** (5M+ pps)
✅ Detects attacks in **<1 second**

☑ Distinguishes **legitimate surges** from attacks
☑ Minimizes **false positives** (<5%)
☑ Operates with **low CPU overhead** (<20%)

Key Question:
How to combine speed of kernel-level filtering with intelligence of machine learning?

SLIDE 4: LITERATURE REVIEW SUMMARY

Existing Approaches

Approach	Pros	Cons
Firewalls	Simple, fast	No intelligence, high false positives
Hardware Appliances	High throughput	Expensive ($100K+), inflexible
ML-Only	Intelligent	Slow (user-space), low throughput
DPDK	Very fast	Complex, dedicated cores

Research Gap:
No solution combines **kernel-level speed** with **ML intelligence**

SLIDE 5: RESEARCH GAP

What's Missing?

Gap 1: Performance vs Intelligence

- Fast systems lack intelligence
- Intelligent systems lack speed

Gap 2: False Positives

- Rule-based: Block legitimate users
- ML-only: Cannot distinguish flash crowds

Gap 3: Deployment Complexity

- Hardware: Too expensive
- DPDK: Too complex

Our Solution: Hybrid eBPF/XDP + ML approach

SLIDE 6: PROPOSED SYSTEM ARCHITECTURE

```
┌─────────────────────────────────────┐
│      Network Interface (NIC)         │
```

```
└──────────────────────────────────────┘
               │ Packets
               ▼
┌──────────────────────────────────────┐
│  KERNEL SPACE (eBPF/XDP)             │
│ • Parse packets (<1µs)               │
│ • Check blacklist                    │
│ • Update statistics                  │
│ • XDP_PASS or XDP_DROP               │
└──────────────────────────────────────┘
               │ Statistics (1 sec)
               ▼
┌──────────────────────────────────────┐
│  USER SPACE (Python)                 │
│  ┌──────────────┬──────────────────┐ │
│  │ Statistical  │  ML Classifier   │ │
│  │ Analysis     │  (Random Forest) │ │
│  └──────────────┴──────────────────┘ │
│          Hybrid Decision              │
└──────────────────────────────────────┘
               │ If Attack
               ▼
      Alert + Blacklist IP
```

Key Innovation: Two-layer defense (kernel + user space)

SLIDE 7: WHY eBPF/XDP + ML?

Hybrid Design Rationale

eBPF/XDP Layer (Kernel):

- ⚡ Ultra-fast: <1µs per packet
- 🚀 High throughput: 5M+ pps
- 🛡 Immediate blocking (blacklist)
- 📊 Statistics collection

ML Layer (User Space):

- 🧠 Intelligent classification
- 🎯 Attack type identification
- 📈 Adaptive learning
- ✅ False positive reduction

Synergy:
Kernel handles speed, ML handles intelligence

SLIDE 8: SYSTEM MODULES

Three Core Components

1. eBPF/XDP Module (C)

- Packet parsing
- Blacklist enforcement
- Statistics collection
- Maps: flow_map, ip_tracking_map, blacklist_map

2. ML Module (Python + scikit-learn)

- Feature extraction (64 CIC features)
- Random Forest classifier
- Attack type classification
- Confidence scoring

3. Monitoring Dashboard (Flask)

- Real-time metrics
- Alert history
- Blacklist management
- Performance graphs

SLIDE 9: DATASET & TRAFFIC SIMULATION

Data Sources

Training Data:

- **CIC-DDoS-2019 Dataset**
  - 50+ million flows
  - 7 attack types
  - 64 statistical features
  - Labeled (BENIGN vs ATTACK)

Attack Types:

- SYN Flood
- UDP Flood
- DrDoS DNS/LDAP/NTP
- HTTP Flood
- ICMP Flood

Simulation:

- Custom attack simulator (Python)
- Configurable rates (100-100K pps)
- Multiple attack vectors

SLIDE 10: ML MODEL DETAILS

Random Forest Classifier

Features (64 total):

- Flow duration, packet counts
- Bytes/packets per second
- Inter-arrival times (IAT)
- TCP flags (SYN, ACK, FIN, RST)
- Packet length statistics

Model Configuration:

- Algorithm: Random Forest
- Trees: 100
- Max depth: 15
- Training samples: 250,000
- Test accuracy: **95.3%**

Why Random Forest?

- Fast inference (<10ms)
- Handles high-dimensional data
- Robust to overfitting
- Feature importance analysis

SLIDE 11: eBPF/XDP IMPLEMENTATION

Kernel-Level Packet Filtering

XDP Program Flow:

```
1. Packet arrives at NIC
2. XDP hook triggered
3. Parse headers (Ethernet, IP, TCP/UDP)
4. Check blacklist → If YES: XDP_DROP
5. Update statistics maps
6. Check SYN flood → If YES: XDP_DROP
7. Return XDP_PASS
```

eBPF Maps:

- `stats_map`: Per-CPU global statistics
- `ip_tracking_map`: Per-IP counters (131K entries)
- `flow_map`: 5-tuple flow tracking (65K entries)
- `blacklist_map`: Blocked IPs (10K entries)

Performance:

- Processing time: <1 microsecond

- Throughput: 5M+ packets/second

SLIDE 12: EXPERIMENTAL SETUP

Testbed Configuration

Hardware:

- CPU: Intel Core i7-9700K (8 cores, 3.6 GHz)
- RAM: 16 GB DDR4
- NIC: Intel X550-T2 (10 Gbps)

Software:

- OS: Ubuntu 22.04 LTS
- Kernel: 5.15.0 (XDP support)
- Python: 3.10
- BCC: 0.25.0
- scikit-learn: 1.2.0

Network Topology:

```
[Attacker] → [DDoS System] → [Protected Server]
```

SLIDE 13: RESULTS - PERFORMANCE METRICS

Key Performance Indicators

Metric	Target	Achieved	Status
Detection Latency	<1 sec	0.8 sec	☑
Throughput	5M+ pps	5.2M pps	☑
ML Accuracy	>90%	95.3%	☑
False Positive Rate	<5%	1.8%	☑
CPU Overhead	<20%	18.2%	☑
ML Inference Time	<10ms	8.3ms	☑

Graphs:

- Detection latency vs packet rate
- CPU utilization over time
- Accuracy vs different attack types

SLIDE 14: COMPARATIVE ANALYSIS

Performance Comparison

System	Throughput	Latency	Accuracy	False Positives	Cost
Traditional Firewall	1M pps	High	70%	15%	Medium
Hardware Appliance	10M+ pps	Low	85%	8%	$100K+
ML-Only (User Space)	100K pps	Very High	94%	3%	Low
DPDK-based	10M+ pps	Low	80%	10%	Medium
Our System	**5.2M pps**	**Very Low**	**95.3%**	**1.8%**	**Low**

Key Advantage: Best balance of speed, accuracy, and cost

SLIDE 15: KEY OBSERVATIONS

What Worked Best

☑ **Successes:**

1. Hybrid detection significantly reduced false positives
2. eBPF/XDP achieved target throughput (5M+ pps)
3. ML inference fast enough for real-time (<10ms)
4. Blacklist enforcement effective (instant drops)
5. Dashboard provided valuable insights

⚠ **Challenges:**

1. eBPF verifier constraints (limited loops)
2. Feature extraction overhead
3. Balancing detection sensitivity
4. Handling encrypted traffic

💡 **Key Insight:**
Combining kernel speed with ML intelligence is the optimal approach

---

SLIDE 16: CONCLUSION

Project Outcomes

Achievements: ☑ Implemented hybrid DDoS mitigation system
☑ Achieved 5.2M pps throughput
☑ Detection latency: 0.8 seconds
☑ ML accuracy: 95.3%
☑ False positive rate: 1.8%
☑ CPU overhead: 18.2%

Contributions:

1. Novel hybrid eBPF/XDP + ML architecture
2. Real-time feature extraction pipeline
3. Adaptive baseline learning mechanism
4. Open-source implementation

Impact:
Practical, cost-effective DDoS mitigation for enterprise networks

---

SLIDE 17: FUTURE SCOPE

Enhancements & Extensions

Short-term (6 months):

- IPv6 support
- Deep learning models (LSTM, CNN)
- Hardware offload to SmartNICs
- Multi-interface support

Long-term (1-2 years):

- Distributed multi-node deployment
- Auto-scaling in cloud environments
- Encrypted traffic analysis
- Integration with SIEM systems

Research Directions:

- Online learning for zero-day attacks
- Federated learning across nodes
- Explainable AI for attack attribution

SLIDE 18: DEMO SCREENSHOT

System Dashboard

[Insert screenshot of web dashboard showing:]

- Real-time packet rate graph
- Current PPS and baseline
- Top source IPs
- Blacklist status
- Recent alerts
- ML classification results
- CPU/memory usage

Live Demo: http://localhost:5000

SLIDE 19: PUBLICATIONS & TOOLS

Technologies Used

Core Technologies:

- eBPF/XDP (Kernel-level filtering)
- BCC (BPF Compiler Collection)
- Python 3.10
- scikit-learn (Random Forest)
- Flask (Web dashboard)
- NumPy, pandas (Data processing)

Dataset:

- CIC-DDoS-2019 (Canadian Institute for Cybersecurity)

Development Tools:

- Linux Kernel 5.15+
- GCC/Clang (eBPF compilation)
- Git (Version control)
- pytest (Testing)

Potential Publication: *"Hybrid DDoS Mitigation using eBPF/XDP and Machine Learning"*

SLIDE 20: THANK YOU & Q&A

Questions?

Contact:
[Your Name]
[Email]
[GitHub/Project Repository]

Project Repository:
github.com/[username]/rapid-corona

Documentation:
Complete technical documentation available in project repository

---

Prepared for Final Review
Department of Computer Science and Engineering
[College Name]
Academic Year 2025-2026

PRESENTATION TIPS

Delivery Guidelines

Timing:

- Total: 15-20 minutes
- Introduction: 2 min
- Literature & Gap: 3 min
- Architecture: 3 min
- Implementation: 4 min
- Results: 4 min
- Conclusion: 2 min
- Q&A: 5-10 min

Key Points to Emphasize:

1. **Problem significance** (Slide 2)
2. **Research gap** (Slide 5)
3. **Hybrid approach** (Slide 7)
4. **Performance results** (Slide 13)
5. **Comparative advantage** (Slide 14)

Anticipated Questions:

- Why eBPF/XDP over DPDK?
- How does ML inference not slow down the system?
- What happens with encrypted traffic?
- Can this scale to multiple nodes?
- How do you handle false positives?

Demo Preparation:

- Have dashboard running
- Show live attack simulation
- Display real-time detection
- Show blacklist in action

End of PPT Content