FAU

CYBER THREAT INTELLIGENCE LAB
College of Engineering & Computer Science
Florida Atlantic University

ZEEK (BRO) INTRUSION DETECTION SYSTEM (IDS)

ELIAS BOU-HARB, Ph.D.

Assistant Professor

ANTONIO MANGINO

Research Assistant

NSF Award 1829698

CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers

July 23rd, 2019

Training Workshop for Network Engineers and Educators on Tools and Protocols for High-Speed Networks

Zeek (Bro) IDS Outline

- Network Intrusion Detection Systems
- Network Traffic Signatures
- Zeek (Bro) IDS
- Network Scanning Detection with Zeek
- Denial of Service Detection with Zeek
- Internet Measurements using Zeek for IoT Security

Network Intrusion Detection Systems

- Software/hardware systems that actively monitor live networks for malicious traffic, policy violations and unidentified anomalies

- Deployed to protect operational networks without disturbing normal/benign packet traffic flows

- In contrast to firewalls, NIDS are most often passive, although they can operate as NIPS

Network Traffic Signatures

- Typically, IDS search for identified packet signatures to determine malicious or unsolicited network activity

- Zeek leverages an event-based engine to monitor possible intrusions, permitting more versatile handling of malicious traffic

- Zeek supports signature conversion, resulting in traditional signature-matching while combining the adaptability of the event-based engine

Network Traffic Signatures: A *Snort* Signature

- Follows a rule-based format

(Action) (Protocol) (Source Address) (Source Port) (Direction) (Destination Address) (Destination Port)

alert tcp any 80 -> 192.168.x.x any (msg: “TCP Packet”; sid:100)

Rule Header

Rule Option

```
alert tcp any any -> [a.b.0.0/16,c.d.e.0/24] 80 (msg: “WEB-ATTACKS
conf/httpd.conf attempt”; nocase; sid:1373; flow:to_server, established;
content:“conf/httpd.conf”; […] )
```

Network Traffic Signatures: A *Zeek* Signature

- Follows a variable/data object-based format
- Variables support strings, integers and floats

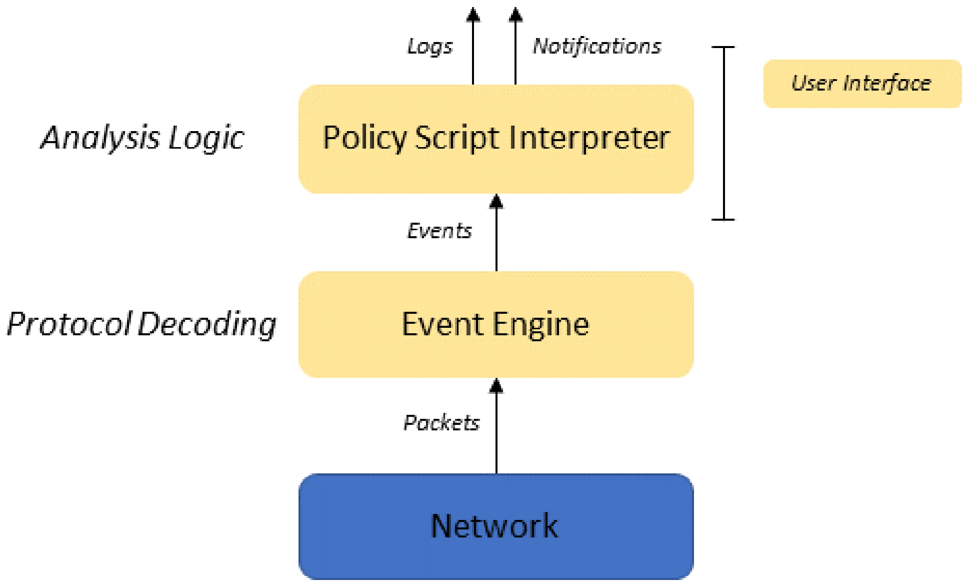
```
signature sid-1371 {
                ip-proto == tcp
                dst-ip == a.b.0.0/16,c.d.e.0/24
                dst-port == 80
payload /.*conf/\httpd\.conf/
tcp-state established, originator
event “WEB-ATTACKS conf/http.conf attempt”
}
```

Zeek (Bro) IDS

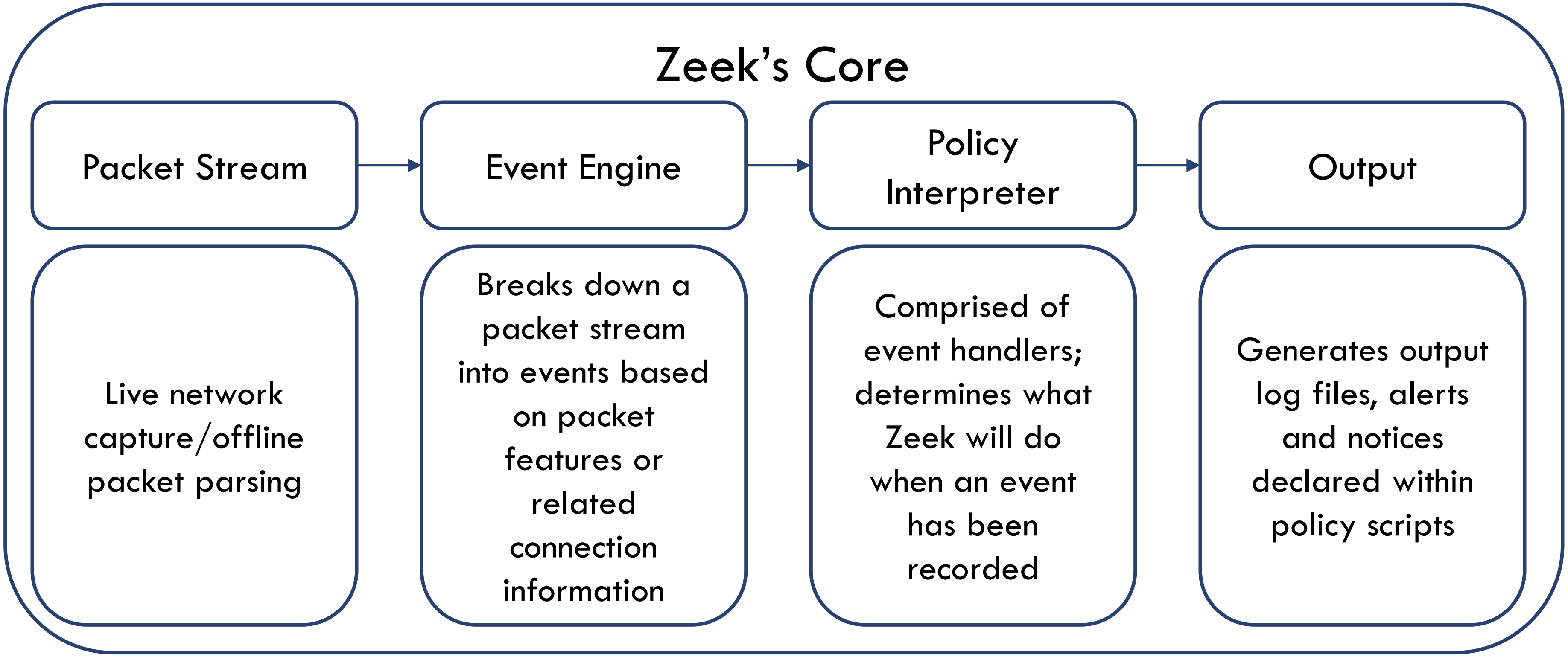
- Development began in 1995 by Vern Paxon
- Real-time notifications of possible network intrusions
- Zeek's scripting language creates a versatile environment for fine-grained anomaly-related detection and processing
- Diverse log files containing distributed information
- Versatile formatting of output data for preprocessing and advanced analytics

Zeek (Bro) IDS: Event Engine

- Zeek processes live and captured network traffic to generate events
- Each event triggers a corresponding policy script
- Policy scripts determine the actions taken when an event is recorded



Zeek (Bro) IDS: Event Engine



Zeek's Core

Packet Stream → Event Engine → Policy Interpreter → Output

Packet Stream: Live network capture/offline packet parsing

Event Engine: Breaks down a packet stream into events based on packet features or related connection information

Policy Interpreter: Comprised of event handlers; determines what Zeek will do when an event has been recorded

Output: Generates output log files, alerts and notices declared within policy scripts

Zeek (Bro) IDS: Log Files

- □ After processing network traffic, Zeek will output statistical log files
- □ By default, log files will be separated by the transport protocol and related characteristics
- □ At a basic level, these log files can be used to determine the presence of an anomaly
- □ Zeek log files can be formatted and exported to external processing software

Zeek (Bro) IDS: Log Files

- Connection:
  - conn.log: collection of all TCP/UDP/ICMP connections
  - files.log: analysis results
  - x509.log: X.509 certificate information

Connection	Protocol-Specific	Detection	Observations
conn.log	http.log	notice.log	known_certs.log
files.log	ftp.log	signatures.log	known_services.log
x509.log	dns.log	traceroute.log	weird.log

Zeek (Bro) IDS: Log Files

- Protocol-Specific:
  - http.log: collection of all packets using the Hyper Text Transport Protocol (HTTP)
  - ftp.log: collection of all packets using the File Transport Protocol (FTP)
  - dns.log: collection of all packets using Domain Name System (DNS)

Connection	Protocol-Specific	Detection	Observations
conn.log	http.log	notice.log	known_certs.log
files.log	ftp.log	signatures.log	known_services.log
x509.log	dns.log	traceroute.log	weird.log

Zeek (Bro) IDS: Log Files

- Detection:
  - notice.log: Zeek event notices
  - signatures.log: collection of matched signatures
  - traceroute.log: detected traceroute traffic

Connection	Protocol-Specific	Detection	Observations
conn.log	http.log	notice.log	known_certs.log
files.log	ftp.log	signatures.log	known_services.log
x509.log	dns.log	traceroute.log	weird.log

Zeek (Bro) IDS: Log Files

- Observations:
  - known_certs.log: collection of SSL certificates
  - known_services.log: collection of active software on the network
  - weird.log: unexpected or anomalous activity statistics

Connection	Protocol-Specific	Detection	Observations
conn.log	http.log	notice.log	known_certs.log
files.log	ftp.log	signatures.log	known_services.log
x509.log	dns.log	traceroute.log	weird.log

Zeek (Bro) IDS: Policy Scripts

- The Zeek scripting language is used to develop and implement filters and policies for the event-based engine

- Event-based scripts are used to customize the output of Zeek processing

- Scripts can be implemented to permanently update Zeek’s event handling or used as a non-permanent filter

Zeek Filters

- Script events include (but are not limited to):
  - Protocol-specific events
  - Application-level headers
  - Unknown/broken connection handling

- Packet data is accessible within the filters to be used for calculations or to be exported into separate log files

Example: Protocol-oriented Zeek Filter

- □ Filter with UDP Request and UDP Reply events

- □ If a processed packet is using the UDP protocol, source and destination information will be printed

```
event udp_request(u:connection){
		print fmt(“A UDP Request was found!”);
		print fmt(“Source Address: %s Destination Port: %s”,
				u$id$orig_h, u$id$resp_p);
}
event udp_reply(u: connection){
		print fmt(“A UDP Reply was found!”);
		print fmt(“Source Address: %s Destination Address: %s”,
				u$id$orig_h, u$id$resp_h);
}
```

Example: Protocol-oriented Zeek Filter

- □ Filter using a connection-based event
- □ If a processed packet uses the HTTP service that is different port 80, the source IP address will be printed

```
event new_connection(c: connection){

		if (c$id$service == “http” && c$id$resp_p != 80){
				print fmt(“Traffic Anomaly Detected!”);
				print fmt(“Source Address: %s”, c$id$orig_h);
		}
}
```

Network Scanning Detection with Zeek

- Network scanning is a preliminary action to infer aliveness, available services or vulnerabilities
- Various techniques are used by network scanners to bypass firewalls and avoid detection
- Scanning traffic includes an array of transport and application layer protocols
- Scanning traffic can be identified by header flags, destination patterns and related packet information

Network Scanning Detection with Zeek: An example

- Develop a detector based on the number of TCP connections initiated by a source IP address within a continuous time interval
- When a scanner is targeting a single port on multiple destination addresses, it is known as horizontal scanning

```
export {
const addr_scan_interval = 5min &redef;
const addr_scan_threshold = 20 &redef;
}
function horizontal_scanning(c: connection):bool {
        if (num_requests(c$id$orig_h) > addr_scan_threshold &&
                time_alive(c$connection) < addr_scan_interval) {
                print fmt(“Horizontal Scanner Detected!”);
                return c$id$orig_h;
        }
}//end function
```

Network Scanning Detection with Zeek: An example

- □ Develop a detector based on the number of failed TCP connections initiated by a source IP address within a continuous time interval
- □ When a scanner is targeting multiple ports on a single destination address; vertical scanning

```
export {
const port_scan_interval = 5min &redef;
const port_scan_threshold = 30 &redef;
}
function vertical_scanning(c: connection):bool {
          if((c$orig$state == TCP_SYN_SENT && c$resp$state == TCP_RESET) ||
          (c$orig$state == TCP_RESET && c$resp$state == TCP_SYN_ACK_SENT){
                    if (num_requests(c$id$orig_h) > port_scan_threshold &&
                              time_alive(c$connection) < addr_scan_interval) {
                              print fmt("Vertical Scanner Detected!");
                              return c$id$orig_h;
          }
}//end function
```

Denial of Service Detection with Zeek

- Denial of Service (DoS) attacks are launched to render a target machine or resource unavailable to its intended users
- DoS techniques utilize the Internet architecture to overwhelm their victim
- DoS attacks can be identified by packet distribution thresholds (unidirectional traffic) or backscatter (passive one-way traffic)

Denial of Service Detection with Zeek: An example

- Develop a threshold based on the connection state, duration and number of bytes per packet sent by a source IP address during an HTTP flood attack

```
export {
const addr_traffic_interval = 5min &redef;
}

function http_request(c: connection):bool {
        if (c$proto = “HTTP” && c$orig$state == S0 &&
                (c$duration < 1 || c$orig_bytes <= 0){
                        print fmt(“HTTP Flood Detected!”);
                        return c$id$orig_h;
        }
}//end function
```

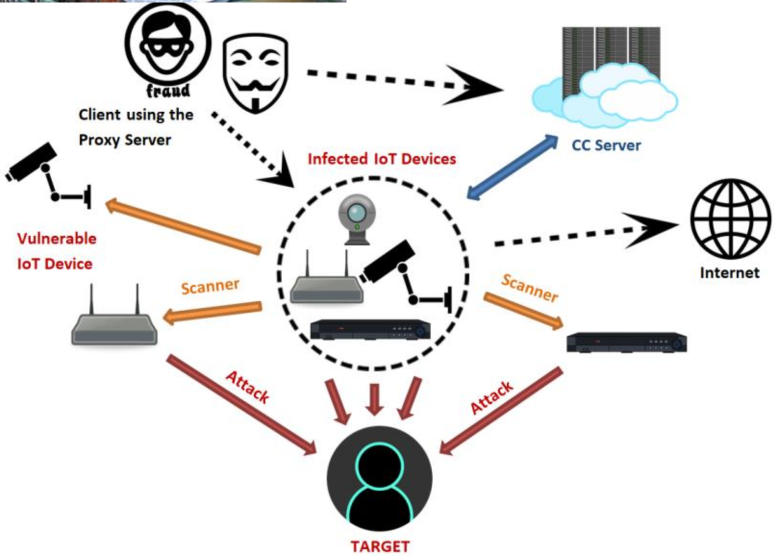
The Internet-of-Things (IoT)

- Internet connected devices and systems
  - Limited resources and functionalities
  - Facilitate data collection, monitoring, and sharing

- Types of IoT
  - Consumer IoT (e.g., routers, printers, IP cameras)
  - CPS - Cyber-Physical Systems (e.g., power utilities, factory automation, smart buildings)

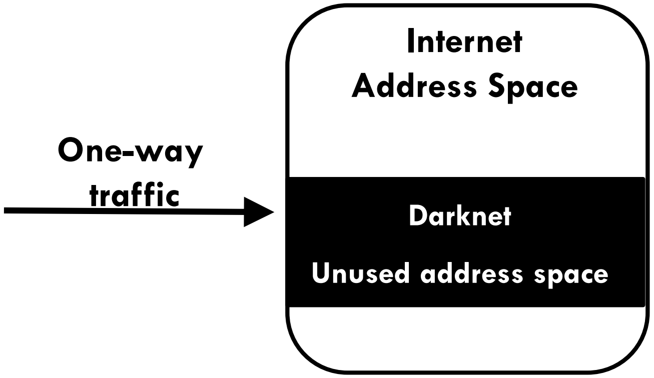
- Worldwide deployment
  - Projected increase with 5G

IoT Security



Passive darknet data

- One-way traffic collected at unused address space (darknet)
  - UCSD Real-Time Network Telescope data provided by CAIDA
  - One of largest darknets (16.7M IPv4 destination addresses)

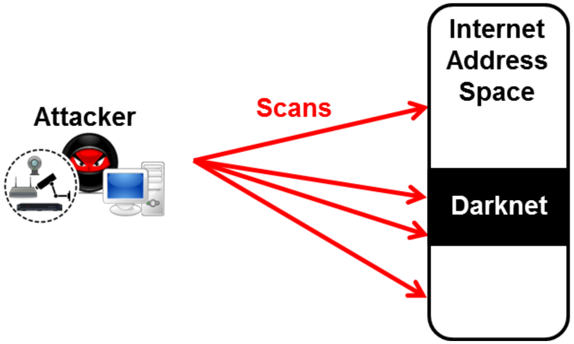


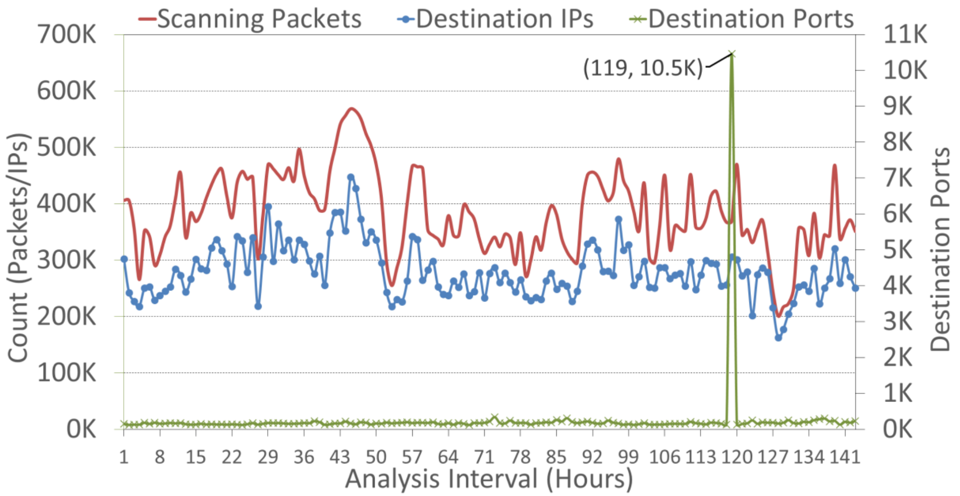
- Obtained data
  - 5 TB of darknet
  - Generated flow information (flowtuples)

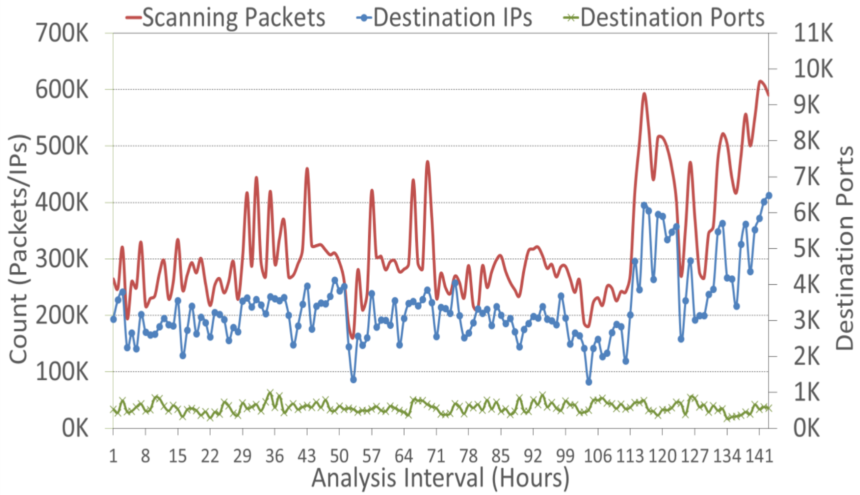
Source IP	Source Port	Dest. IP	Dest. Port	TTL	Protocol	Packets	TCP Flags	IP Length

Leveraging Zeek for inferring IoT-generated scanning traffic

- About 75% of all darknet traffic
- Malicious scans from compromised IoT devices
  - 0.23% ICMP Echo requests (56 IoT devices)
  - 100M TCP packets (99.9% TCP SYN requests)
  - 12.4K devices (55% Consumer IoT)



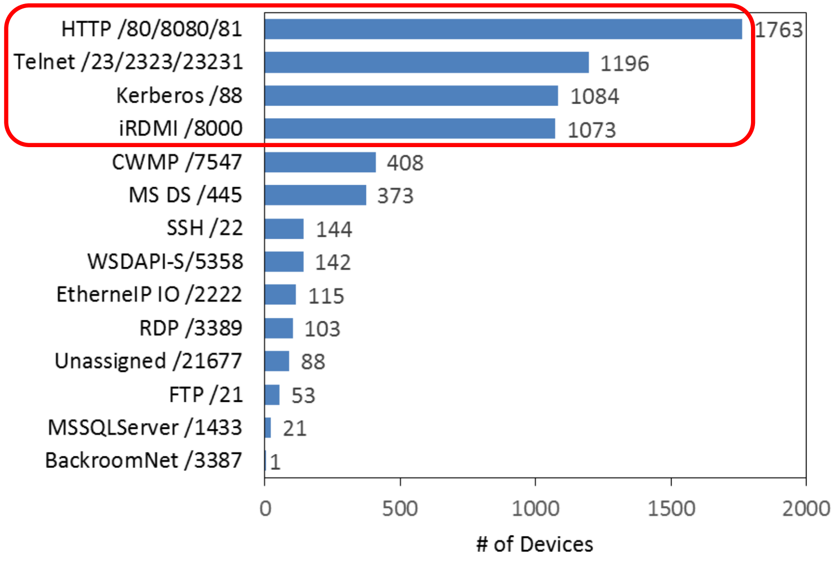




Leveraging Zeek for inferring IoT-generated scanning traffic

Scanned ports	% of packets	
Telnet /23/2323/23231	50.2	
HTTP /80/8080/81	9.4	95% Consumer IoT
SSH /22	7.7	
BackroomNet /3387	6.2	100% CPS
CWMP /7547	4.5	
WSDAPI-S /5358	4.1	
MSSQLServer /1433	3.3	
Kerberos /88	2.7	99% Consumer IoT
MS DS /445	2.5	
EtherneIP IO /2222	0.7	
iRDMI /8000	0.7	99% Consumer IoT
Unassigned /21677	0.6	100% CPS
RDP /3389	0.5	
FTP /21	0.3	

Number of IoT devices (scanners) per port/service



Questions

ELIAS BOU-HAB, Ph.D.
Assistant Professor

ANTONIO MANGINO
Research Assistant

NSF Award 1829698
CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers

CYBER THREAT INTELLIGENCE LAB
College of Engineering & Computer Science
Florida Atlantic University