HONEYPOTS 2.0: DEFENDING INDUSTRIAL SYSTEMS WITH DYNAMIC DECEPTION

Lane Thames, PhD
Senior Security Researcher
Tripwire, Inc.
@Lane_Thames
Motivation

Who has the upper hand in cybersecurity? The good guys or the bad buys? Why?
Agenda

- Industrial Internet of Things
- Cybersecurity challenges for the Industrial Internet of Things
- Deception Technologies for Cybersecurity
  - Honeypots
- Dynamic Deception
  - Next generation Honeypots
  - Scale
INDUSTRIAL INTERNET OF THINGS AND ITS CYBERSECURITY CHALLENGES
Industrial Internet of Things

- Smart Power Grids, Smart Logistics, Smart Inventory, Smart Machine Diagnostics
- Self-monitoring, Group-monitoring
- Self-configuration, Group-configuration
- Self-healing, Group-healing

Provides:
- Operational Efficiencies
- Outcome-driven Processes
- Machine-to-Human Collaboration

Countless Value Creation Opportunities
What is a “Digital Twin”?  

Wikipedia:

Digital twin refers to a digital replica of physical assets, processes and systems that can be used for various purposes. The digital representation provides both the elements and the dynamics of how an Internet of Things device operates and lives throughout its life cycle.

Digital Twins integrate artificial intelligence, machine learning and software analytics with data to create living digital simulation models that update and change as their physical counterparts change.
Industrial Internet of Things
Industrial Internet of Things: What will prevent us from achieving its full potential?
Our Approximate Cybersecurity Solution
Time is always against us. How can we change that?

**Prevention Gap**
Time to put preventative measures in place to avoid repeated attacks

**Detection Gap**
Time between actual breach and discovery

**Response Gap**
Time between discovery to remediation to limit damage

*Can we avoid this from happening again?*

*Have we been breached?*

*How bad is it?*
DECEPTION TECHNOLOGIES FOR CYBERSECURITY AND DYNAMIC DECEPTION
Deception Technologies

deceive
/dəˈsiːv/

verb

(of a person) cause (someone) to believe something that is not true, typically in order to gain some personal advantage.
"I didn't intend to deceive people into thinking it was French champagne"

synonyms: swindle, defraud, cheat, trick, hoodwink, hoax, dupe, take in, mislead, delude, fool, outwit, lead on, inveigle, beguile, double-cross, gull; More

- (of a thing) give a mistaken impression.
  "the area may seem to offer nothing of interest, but don't be deceived"

- fail to admit to oneself that something is true.
  "enabling the rulers to deceive themselves about the nature of their own rule"
Deception-based Cyberattacks - General

- Social Engineering
- Phishing
- Spam
Deception-based Cyberattacks – IIoT Specific

- Spoofed Signals
  - Sensor measurements
  - Control inputs
  - Timestamps
  - Identity information

Deception-based Cybersecurity

- **Honeypots**
  - A computing asset used for detecting, deflecting, or counteracting authorized use of information systems (Wikipedia)
  - Can be used to create “Confusion”
    - Confusion induces a time delay on the attack source
    - Gives us more time to counteract appropriately
  - Can be used to increase to cost of attack thereby reducing attack motivation
  - Scale was once upon a time an issue
Deception-based Cybersecurity

- Honeypots & Dynamic Deception
  - IP-based dynamics
    - DevOps Tool Chains
  - Port-based dynamics
    - Software-based implementation
    - Managed/Deployed via DevOps Tool Chains

- Goals:
  - Primary: Create significant confusion via scale for attackers in such a way to cause delays for their activities
  - Secondary: Use dynamic deception at scale to detect real-time attacks, to generate threat intelligence, and to implement real-time controls
Deception-based Cybersecurity
Dynamic Deception: Port-based Dynamics

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import socket
import random

server = None
resp = "HTTP/1.1 200 OK
nConnection: close\n\n"

while True:
    if server:
        server.shutdown(socket.SHUT_RDWR)
        server.close()
    else:
        server = socket.socket()
        server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        host = socket.gethostname()
        port = random.randint(80, 90)
        server.bind((host, port))
        server.listen(1)
        print ("Listen on port: %s" % port)

    while True:
        client, address = server.accept()
        print ("REAL FROM: %s" % str(address))
        client.send(resp)
        client.close()
        server.shutdown(socket.SHUT_RDWR)
        server.close()
        server = None
        break
```
Dynamic Deception: Port-based Dynamics
Dynamic Deception: Port-based Dynamics

```python
# port/hit/new.py
#
# coding: utf-8
#
import SocketServer
import socket
import threading
import time
import random

class SocketThreadedServer(SocketServer.ThreadingMixIn, SocketServer.TCPServer):
    def __init__(self, host, port):
        self.host = socket.gethostname()
        self.port = port
        self.allow_reuse_address = True
        try:
            print("Starting server on host: %s on port: %d" % (self.host, self.port))
            self.server = SimpleThreadedServer((self.host, self.port), SimpleTCPServer)
            self.server_thread = threading.Thread(target=self.server.serve_forever)
            self.server_thread.daemon = True
            self.server_thread.start()
        except Exception, e:
            print("Error creating server: Exception: %s" % str(e))

    def spin_up(self):
        population = range(8000, 9000)
        num_ports = 20
        ports = random.sample(population, num_ports)
        servers = list()
        for port in ports:
            s = SimpleServer(port)
            servers.append(s)
        return servers

    def spin_down(servers):
        for s in servers:
            if s.servers:
                s.servers = None
                s.servers = None
```

```python
if __name__ == '__main__':
    servers = spin_up()
    time.sleep(15)
    spin_down(servers)
```
Dynamic Deception: Port-based Dynamics

```
root@lthemes-digio:/ics-dyndec# python simple-multiprot-thread-rand.py

Starting @ port: 8194
Starting @ port: 8117
Starting @ port: 8064
Starting @ port: 8477
Starting @ port: 8587
Starting @ port: 8754
Starting @ port: 8515
Starting @ port: 8109
Starting @ port: 8671
Starting @ port: 8242

Starting @ port: 8214
Starting @ port: 8363
Starting @ port: 8081
Starting @ port: 8219
Starting @ port: 8649
Starting @ port: 8514
Starting @ port: 8297
Starting @ port: 8215
Starting @ port: 8619
Starting @ port: 8780

Server @ Thread-21 handling client (43626) request
```

```
root@lthemes-digio:/ics-dyndec# netstat -tan | grep LIST

tcp 0 0 0.0.0.0:8587 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:8109 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8242 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8514 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8671 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:8242 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8514 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8671 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8242 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8514 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8671 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:8242 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8514 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8671 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8242 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8514 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:8671 0.0.0.0:* LISTEN
```

```
root@lthemes-digio:/ics-dyndec# telnet 0.0.0.0 8780
Trying 0.0.0.0...
Connected to 0.0.0.0.
Escape character is '^[].
HTTP/1.1 200 OK
Date: Tue, 17 Oct 2017 19:47:29 GMT
Expires: -1
Content-Type: text/html; charset=ISO-8859-1

Connection closed by foreign host.
```

```
root@lthemes-digio:/ics-dyndec# |
```
Problems with the aforementioned approach?
- Code complexity
- Light-weight honeypot interaction

We can solve these problems with ‘Twisted’!
What is Twisted?

- An event-driven networking engine written in Python
  - Based on a reactive programming model
  - Essentially lets you work with highly asynchronous applications

- Comes “with batteries”
  - Web servers, Mail Servers, SSH servers, Chat servers and many more

- Let’s the programmer focus on the Application Protocol

- Many projects available based on Twisted that fit well with creating honeypots
  - IoT based projects
  - OT (Operational Technology) based projects
Dynamic Deception: Port-based Dynamics

Wait for Events

Reactor Loop

A socket is ready

Twisted code

Our code

reader.doRead()
Dynamic Deception: Port-based Dynamics

```python
from twisted.web.server import Site
from twisted.web.static import File
from twisted.internet import reactor

import random

def rrun():
    reactor.removeAll()
    port = random.randint(8000, 8100)
    print "Listening: %s" % port
    resource = File('web')
    factory = Site(resource)
    reactor.callLater(25, rrun)
    reactor.listenTCP(port, factory)

reactor.callLater(1, rrun)
reactor.run()
```
Dynamic Deception: Port-based Dynamics

```python
from twisted.web.server import Site
from twisted.web.static import File
from twisted.internet import reactor
import random

class SimpleWeb(object):
    def __init__(self, port_low, port_high):
        self.port = random.randrange(port_low, port_high)
        self.factory = Site(File('web'))
        print "Listening @ %s" % self.port
        reactor.listenTCP(self.port, self.factory)

if __name__ == '__main__':
    s1 = SimpleWeb(80, 90)
    s2 = SimpleWeb(91, 100)
    s3 = SimpleWeb(8000, 8100)
    s4 = SimpleWeb(8101, 8200)

    reactor.run()
```
Dynamic Deception: Port-based Dynamics
Dynamic Deception: Port-based Dynamics

```python
from twisted.web.server import Site
from twisted.web.static import File
from twisted.internet import reactor

import random

class SimpleWeb(object):
    def __init__(self, port_low, port_high):
        self.port_low = port_low
        self.port_high = port_high
        self.factory = Site(File('web'))
        self.spinup()

    def spinup(self):
        self.port = random.randint(self.port_low, self.port_high)
        print(f"listening @ %d" % self.port)
        reactor.listenTCP(self.port, self.factory)

    def run(servers):
        print(f"\n\nRestoring listeners.")
        reactor.removeAll()
        for server in servers:
            server.spinup()
        reactor.callLater(20, run, servers)

if __name__ == '__main__':
    s1 = SimpleWeb(50, 50)
    s2 = SimpleWeb(60, 60)
    s3 = SimpleWeb(8000, 8100)
    s4 = SimpleWeb(8101, 8200)
    servers = [s1, s2, s3, s4]
    reactor.callLater(20, run, servers)
    reactor.run()
```
Dynamic Deception: Port-based Dynamics

```
root@thomas-digilo:/ics-dyndece# netstat -tan | grep LIST
  tcp 0 0 0.0.0.0:8189  0.0.0.0:*   LISTEN
  tcp 0 0 0.0.0.0:8020  0.0.0.0:*   LISTEN
  tcp 0 0 0.0.0.0:8021  0.0.0.0:*   LISTEN
  tcp 0 0 0.0.0.0:8022  0.0.0.0:*   LISTEN
  tcp 0 0 0.0.0.0:8023  0.0.0.0:*   LISTEN
  tcp 0 0 0.0.0.0:8189  0.0.0.0:*   LISTEN
  tcp 0 0 0.0.0.0:8020  0.0.0.0:*   LISTEN
  tcp 0 0 0.0.0.0:8021  0.0.0.0:*   LISTEN
  tcp 0 0 0.0.0.0:8022  0.0.0.0:*   LISTEN
  tcp 0 0 0.0.0.0:8023  0.0.0.0:*   LISTEN
```

```
HTTP/2.1 200 OK
Content-Length: 16
Server: TwistedWeb/36.0.0
Date: Sat, 21 Oct 2017 23:33:49 GMT
Content-Type: text/html

<html>
  <body>
    Hello World!
  </body>
</html>
```
def run(factory):
    reactor.removeAll()
    port = random.randrange(500, 599)
    print "Listening on %s" % port
    reactor.listenTCP(port, factory)
    reactor.callLater(10, run, factory)

store = ModbusSlaveContext(
    di = ModbusSequentialDataBlock(0, [1]*100),
    co = ModbusSequentialDataBlock(0, [1]*100),
    hr = ModbusSequentialDataBlock(0, [1]*100),
    ir = ModbusSequentialDataBlock(0, [1]*100))
context = ModbusServerContext(slaves=store, single=True)

identity = ModbusDeviceIdentification()
identity.VendorName = 'Pymodbus'
identity.ProductCode = 'PM'
identity.VendorUrl = 'http://github.com/bashwork/pymodbus/'
identity.ProductName = 'Pymodbus Server'
identity.ModelName = 'Pymodbus Server'
identity.MajorMinorRevision = '1.0'
framer = ModbusSocketFramer
factory = ModbusServerFactory(context, framer, identity)

print "Starting Reactor...."
reactor.callLater(2, run, factory)
reactor.run()
Dynamic Deception: Scale
Summary

- Industrial Internet of Things
- Dynamic Deception
  - Dynamic & Static Honeypots
    - Port Based Dynamics
    - IP Based Dynamics
    - Scale
- Python Twisted Networking Framework
- Code available at Github:
  - https://github.com/jlthames2/ddt
Apply What You Have Learned Today

- **Next week you should:**
  - Consult with your IT/IS teams. Consider taking advantage of Honeypots and scalability with DevOps Tool Chains

- **In the first three months following this presentation you should:**
  - Deploy honeypots within your networks using unused IP space.
  - Consider using the DDT as a guide to have your IT/IS staff implement honeypots with a mixture of static (traditional) and dynamic instances

- **Within six months you should:**
  - Integrate data collected by your new honeypots into your threat intelligence feeds, and possibly be creating real-time security controls based on this intelligence
  - Consider sharing your threat intelligence with the larger community, at least in terms of IP sources and other indicators of compromise
THANKS FOR ATTENDING!

QUESTIONS?