# Managing DDoS Attacks

Kevin Nassery

Managing Consultant, Cigital
Faculty, IANS Research

kevin@nassery.org | knassery@cigital.com
twitter: @knassery

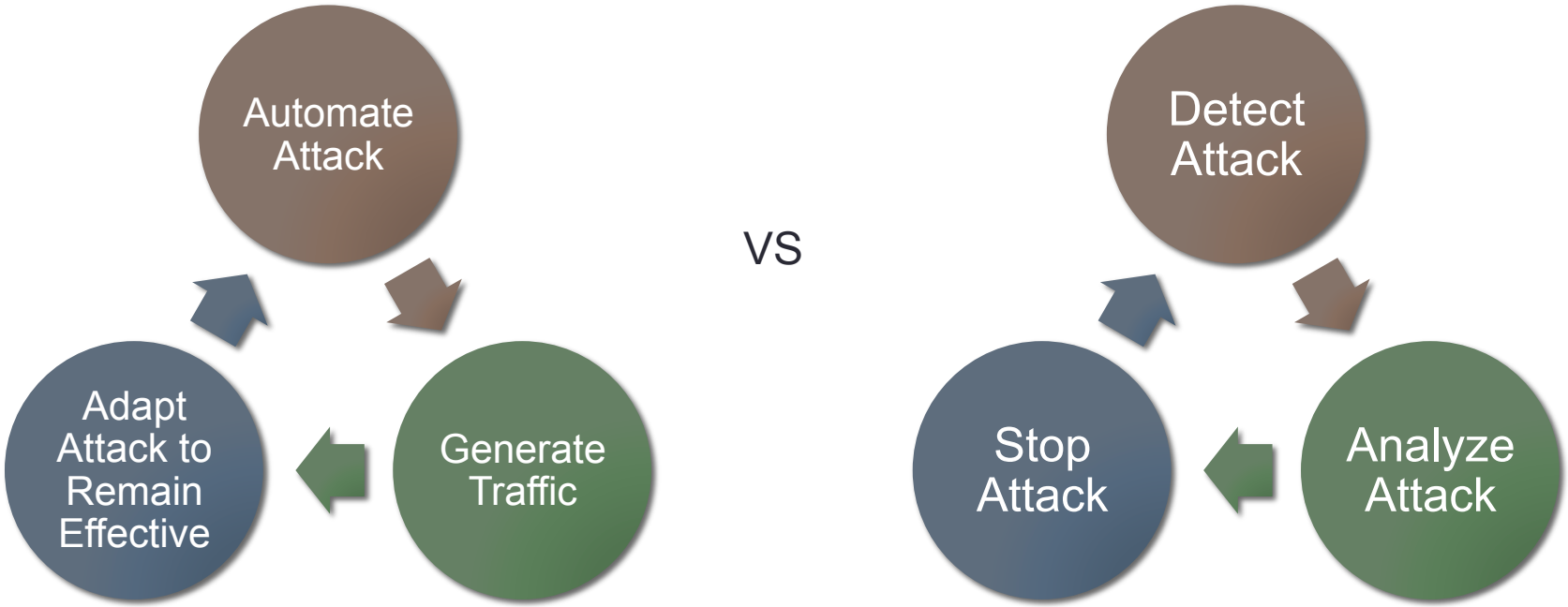
# me

- Former VP, Assessment Services @ US Bank where I acted as technical lead for the DDoS response efforts
  - Disclosure: This presentation is independent from my efforts at US Bank, and will not have any detail regarding the bank's response efforts or DDoS defenses
- Currently Managing Consultant at Cigital & IANS Faculty Member
- Background in ***performance engineering***, systems administration, network engineering, network security, security architecture, penetration testing, app sec...
- Based in Minneapolis

# talk

- Resource depravation attacks aimed at overwhelming a service from multiple sources
- Focus on public web services using HTTP/HTTPS
- Network layer attacks like SYN Flooding, UDP flooding against DNS, and others are still happening too
- Mechanics of both attacking and defending
- The unique challenge of the service exposure being the vulnerability itself, you MUST live in a state of vulnerability.

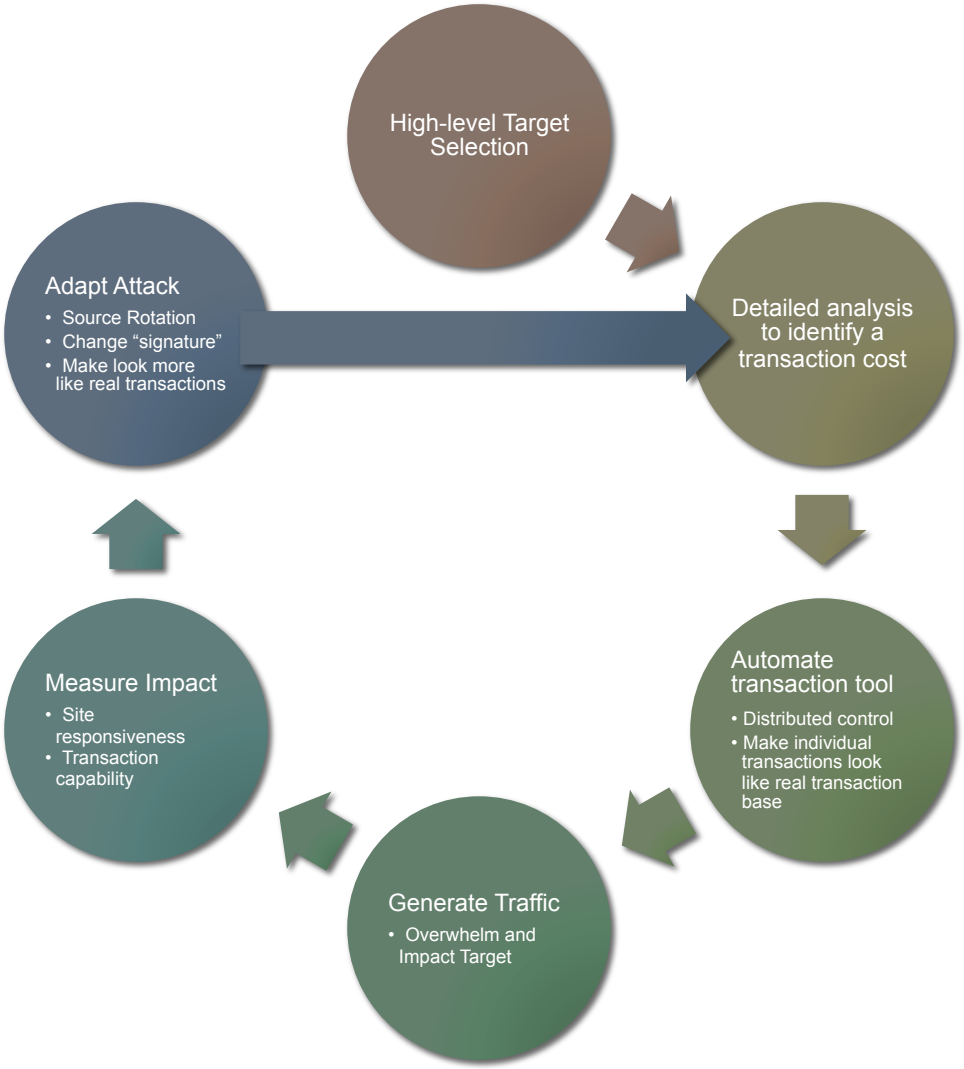
# arms race / life-cycles




Offense Life-cycle

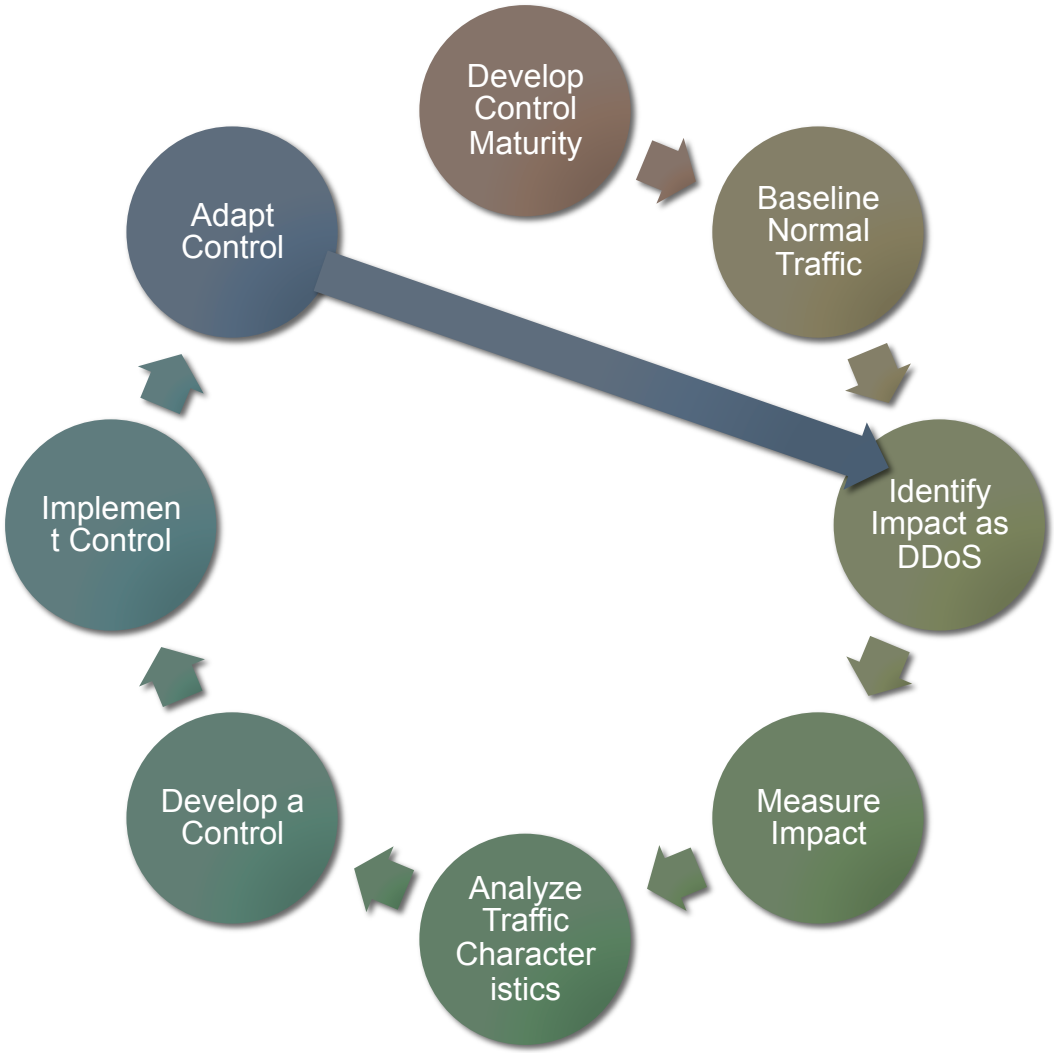
Defense Life-cycle

# attacker perspective




- Best attack against interactive web site would be infinite number of human users using the site as intended
- Making the automated tool indiscernible from legitimate transactions is the hardest piece for the attacker
- Good attackers iterate, adapt, and persist
- Relatively equal footing between offense and defense
- Attacker goal is impact, not sophistication
- The mechanical advantage an attacker has is a disproportionate client computing power

# defender perspective




- In order to discriminate attack traffic you have to know what your real traffic looks like
- Capacity is not an effective control
- Control capability must be mature in the following ways:
  - protocol visibility
  - flexibility to filter patterns
  - limited performance penalty
  - generally as close to the attacker as possible
- You must be creative with your ability to adapt controls
- Use your calm times wisely
- Don’t assume attacker sophistications when you don’t know where you are in the cycle.

# response analysis


- The goal is to identify a pattern, for which you can implement a filter, that best differentiates attack traffic from legitimate traffic.
  - A baseline of “normal” is almost essential.
- Network engineers who are most familiar with sniffers, tend to lean towards network layer analysis (bandwidth, connections/second)
  - This can be just wrong (most effective app attacks drop total bandwidth)
  - HTTP pipelining, and tunneling can hurt correlation of connects: requests
  - Think application level, http.requests/second vs connections
- If you have good protocol visibility (SSL), a good capture architecture (quickly go from edge to analyst’s workstation), and good people (understand traffic and controls) this is pretty clear.
- Experience helps long term control strategy.

# Response Analysis Examples, what are my top User-Agents?

“tshark -c 100 -R 'http.request.method=="GET"' -Tfields -e http.user_agent -r file | sort | uniq -c | sort –nr”

```
 18 Mozilla/5.0 (Windows NT 5.1; rv:15.0) Gecko/20100101 Firefox/15.0.1
  9 Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0) Gecko/20100101 Firefox/15.0.1
  7 Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_5_7; en-us) AppleWebKit/530.17 (KHTML, like Gecko) Version/4.0 Safari/530.17
  6 Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
  6 Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/534.57.2 (KHTML, like Gecko) Version/5.1.7 Safari/534.57.2
  6 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) ; .NET
  5 Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; FunWebProducts; BOIE9;ENUSMSNIP)
  5 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB7.4; .NET CLR 1.0.3705; .NET CLR 1.1.4322; Media Center PC 4
  3 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
  2 Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
  2 Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/21.0.1180.89 Safari/537.1
  2 Mozilla/5.0 (Macintosh; Intel Mac OS X 10_5_8) AppleWebKit/534.50.2 (KHTML, like Gecko) Version/5.0.6 Safari/533.22.3
  2 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.307
  2 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB7.4; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.
  2 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152;
  2 Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
  2 Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
  1 Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; ie9rrdl)
  1 Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; Trident/5.0; BOIE9;ENUSSEM)
  1 Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.4 (KHTML, like Gecko) Chrome/22.0.1229.79 Safari/537.4
  1 Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.4 (KHTML, like Gecko) Chrome/22.0.1229.79 Safari/537.4
  1 Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/21.0.1180.89 Safari/537.1
  1 Mozilla/5.0 (Windows NT 5.1; rv:10.0.2) Gecko/20100101 Firefox/10.0.2
  1 Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_1) AppleWebKit/536.25 (KHTML, like Gecko) Version/6.0 Safari/536.25
  1 Mozilla/5.0 (Linux; Android 4.0.4; LG-MS770 Build/IMM76I) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.166 Mobile Sa
  1 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; GTB7.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .
  1 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.307
  1 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; WOW64; Trident/4.0; GTB7.4; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0;
  1 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727; MS-RTC LM 8; .NET CLR 3.5.30729; .NE
  1 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; GomezAgent 3.0)
  1 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB7.4; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.450
  1 Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.4506.2152;
  1 Mozilla/4.0 (compatible; MSIE 8.0; AOL 9.7; AOLBuild 4343.27; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .N
  1 Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.5.30729; .NET CLR
  1 Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; GTB7.4; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET
  1 Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; .NET CLR 3.0.4506.2152; .NET CLR 3.5.
```

# Response Analysis Examples, what are my top requested objects?

“tshark -c 5000 -R 'http.request.method=="GET"' -Tfields -e http.request.uri -r file.cap | sort | uniq -c | sort –nr“

```
 159 /
 156 /index.ht
 118 /homepag
 104 /homepag
 101 /homepag                                              eader.png
 100 /images/
 100 /homepag
  97 /homepag
  97 /homepag                                              -all.js
  96 /homepag
  96 /homepag                                              png
  94 /js/omni
  93 /homepag
  91 /homepag                                              n.png
  91 /homepag                                              ient.png
  91 /foresee/
```

## Response Analysis Examples, what are my typical referers?

```
tshark -c 5000 -R 'http.request.method=="GET"' -Tfields -e http.referer -r
file.cap | sort | uniq -c | sort –nr
```

Extracting TCP Payload for additional analysis:
“tshark -c 1 -R 'tcp.reassembled.data contains "gif"' -Tfields -e ip.src -e *tcp.reassembled.data* -r file.cap”

# Want to use your favorite XML/Xpath tools? Say hello to PDML:

“tshark -c 1 -Tpdml -r file.cap”

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="pdml2html.xsl"?>
<!-- You can find pdml2html.xsl in /Applications/Wireshark.app/Contents/Resources/share/wireshark or at http://
anonsvn.wireshark.org/trunk/wireshark/pdml2html.xsl. -->
<pdml version="0" creator="wireshark/1.9.0-SVN-45942" time="Tue Apr 15 23:00:24 2014" capture_file="">
Capturing on en0
<packet>
  <proto name="geninfo" pos="0" showname="General information" size="66">
    <field name="num" pos="0" show="1" showname="Number" value="1" size="66"/>
    <field name="len" pos="0" show="66" showname="Frame Length" value="42" size="66"/>
    <field name="caplen" pos="0" show="66" showname="Captured Length" value="42" size="66"/>
    <field name="timestamp" pos="0" show="Apr 15, 2014 23:00:24.982234000 CDT" showname="Captured Time"
value="1397620824.982234000" size="66"/>
  </proto>
  <proto name="frame" showname="Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0"
size="66" pos="0">
    <field name="frame.interface_id" showname="Interface id: 0" size="0" pos="0" show="0"/>
    <field name="frame.encap_type" showname="Encapsulation type: Ethernet (1)" size="0" pos="0" show="1"/>
    …
…

      </field>
    </field>
  </proto>
</packet>
```

# attacking pitfalls

- Automation errors
  - Errors in protocol compliance
    - Bad whitespace
    - spelling
    - capitalization of headers/request
    - not calculating calculated fields (request length/etc)
  - Uncommon request parameters
    - user-agent, consistent request sizes, rare or exotic presence of headers
    - Oddly consistent or missing “referrer”
- Human vs Machine
  - It’s very difficult to make request timings look human and remain effective, and it’s rarely done. (Humans don’t run 500 searches per minute)
  - It’s also more complex for attacker tools to replicate the business logic (humans load the main page before they run a career search)
- **Insufficient confidentiality of control mechanisms**
  - Often times the resource pool is infiltrated or a compromised system is analyzed and the control mechanism is discovered.
  - Depending on the circumstances defenders often know the specific attacks before they’re being launched.
- Insufficient Source Rotation
  - Contrary to some contentions, source filtering is only ineffective in transaction based systems IF the attacker reserves sources to rotate.
- Up against effective threat intelligence (sometimes)

# defending pitfalls

- Making assumptions about attacker
  - Attack only needs to be complex enough to cause impact
  - Successful attackers start simple, and get complex causing the maximum lifecycle of impact
  - We generally have very little reliable information on threats, but sometimes because it's the only information we have prior to attack it can unbalance our defense strategy
- Lack of organization capability to coordinate response
  - To defend the system, and develop controls you need comprehensive understanding of network, app, and system architecture
  - Once a pattern is recognized you need people who can think creatively about how to implement a control in the best space based on that pattern
- Limited control maturity
  - Lack of protocol visibility (SSL) at network layer means coordinating a server level response
  - Not in position to quickly sample and analyze traffic
  - Lack of in-band pattern filtering mechanisms
- Bad causal analysis
  - Does bandwidth go up or down during an application layer DDoS?
  - Is the web server slow because it's serving more concurrent connections, or is it serving more concurrent connections because it's slow?
- Bad knowledge of normal
  - If you don't understand the business as usual traffic mechanics of your environment, response is much harder
  - The need for an up to date non-attack baseline traffic sample can be underemphasized
- Get wrapped around "detection."

# evolving defense

- Defensive programming around heavy transactions:
  - Captcha's, Tokens (ala CSRF), Business Logic enforcement, moving heavy operations behind authentication
- Be aware of security control DoS attacks like account lock-out, IPS filtering based on unauthenticated src-ip (land attack, syn flood)
- Strong in-band filtering solutions, the closer to the source the better
- Robust network and infrastructure layer protection for common attacks (DNS, UDP, Syn/Connect Flooding) to force the ARMS race into that app space
- Be prepared to quickly get real world data samples into hands of analysts (Data Access Networks, fast sample availability, good bandwidth to end users).
- Buffer solution with excess capacity, but don't try to scale up to DDoS
- Be creative, this is very much a problem solving battle having the right people responding is more important than owning the right products
- Understand the cost of unavailability, and budget defenses accordingly.
- Table-top exercises regularly, and real world attack/defense in a scheduled window with isolation between teams.