The Top 10 Ty's and Cy's of Cyber Resiliency

28 July 2022

Phenomenati Consulting

Scott Foote, Managing Director

Steve Foote, Managing Director

Top 10 Challenges (Ty's and Cy's) of Cyber Resiliency

Phenomenati

System-related Challenges

1. Cyber Entropy™
2. Complexity
3. Dependency
4. Vulnerability
5. Fragility

Acquisition-related Challenges

6. Urgency
7. Simplicity
8. Commodity
9. Efficiency
10. Fantasy

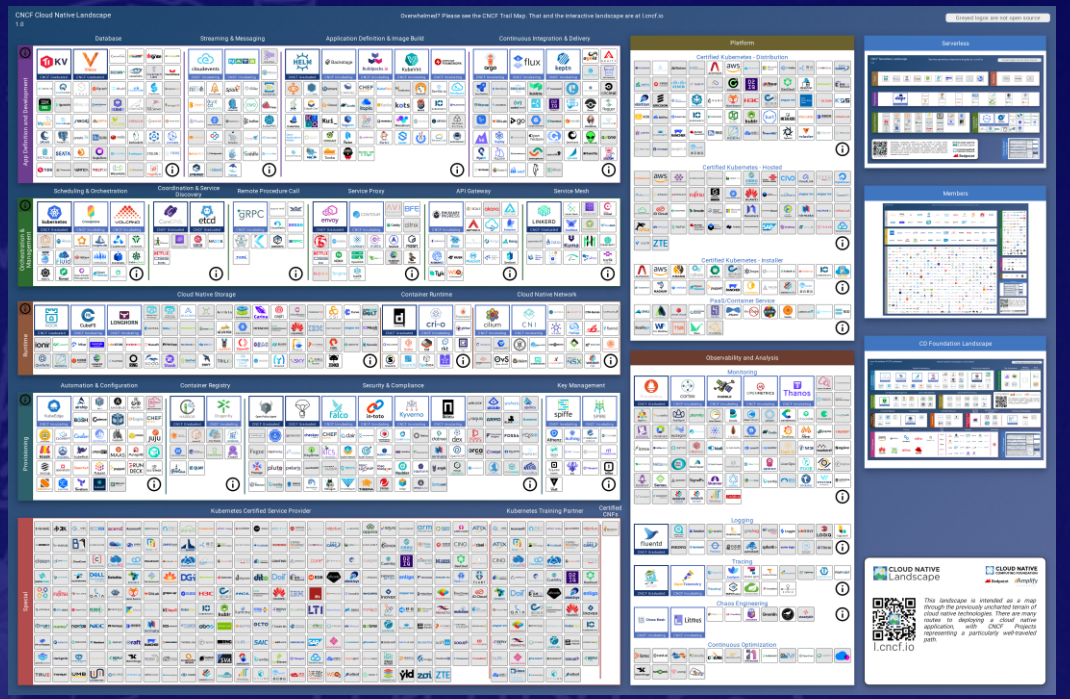
Cyber Entropy™

What it is

- Sprawling expansion of the IT, OT, ICS/SCADA, IoT landscape
- Continuously evolving supply chain of components and services from external providers
- Ever growing ***attack surface***

Examples

- "*Shadow*" IT
- Expansive *Cloud* Services Market
- Convergence / Integration of *OT* into IT
- *IoT* sprawl: Smart Bulbs (Checkpoint's research)



Cloud Native Computing Foundation

Remediation

- Organizational Commitment to Systems Engineering discipline
  - ***Designs***, Baselines, Asset Management, Change Management, Risk Management
- Continuous Discovery
- Significant focus on SDLC management (cradle-to-grave) for *every* mission-critical component
- Commitment to well-resourced Vendor Risk Management

3:00

Complexity

Phenomenati

What it is

- Simple, siloed, self-contained capabilities are all but gone
- Replaced by ***complex systems*** of independent, loosely integrated services from a wide range of providers
- Accelerating adoption of continuously evolving components rapidly results in mounting technical debt

Examples

- Transportation, Energy, SCADA, Financial Services, Healthcare, Entertainment, etc.
- "*Complexity is killing software developers*", Scott Carey, InfoWorld, 1 Nov 2021
- "There is a clear increase in complexity when you move (your business) to a pervasive microservices environment." Amazon CTO Werner Vogels during the AWS Summit in 2019

Remediation

- Essential Complexity vs Accidental Complexity – Be Deliberate
- Create & maintain a Software Bill of Materials (SBoM) for each complex, critical system
- Leverage abstractions to plan/communicate/coordinate… but capture the DETAILS
  - E.g., Architectural Decision Records (ADRs), "Sustainable Architecture Design Decisions", IEEE Software magazine
- Failure to document the details results in "***Willful Ignorance***"

3:00

Dependencies

What it is

- Complexity obscures Dependencies
- Complex webs of independent, loosely integrated services
- Every Dependency presents its own set of criticalities and vulnerabilities
- "… the average [software] repo will have nearly 700 indirect dependencies",
  - Maya Kaczorowski, GitHub, senior director of product management and software supply chain security

https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610

Examples

- DDoS Attack on the Belnet ISP
- "Dependency Confusion" in Building Software Applications
  - "How I Hacked Into Apple, Microsoft, and Dozens of other Companies"
  - https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610

So What? *What If?*

Nth Order Impacts

CONTEXT is EVERYTHING

Remediation

- Use Fewer Dependencies (which is becoming a more common practice; e.g., jQuery)
- **Plan for Contingencies**: avoid building dependencies that result in a ***single point of failure***
- Use Trusted Repositories (e.g., Google's Assured Open Source initiative)
- Use Obfuscated Package Names internally (until there are namespaces leveraged in your Build tools)

3:00

Vulnerability

What it is

- Any weakness (people, process, or technology) that presents risk to the system
- Not a “problem to be fixed”, but a continuous *attribute* of every component
- They present significant opportunity to adversaries, who ***will*** exploit them

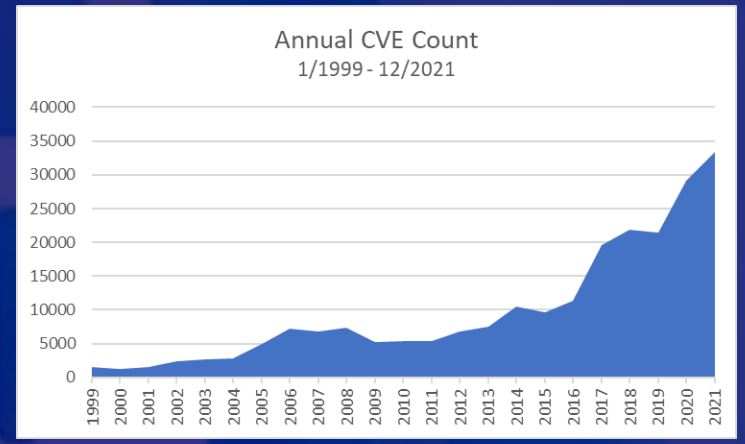
Examples

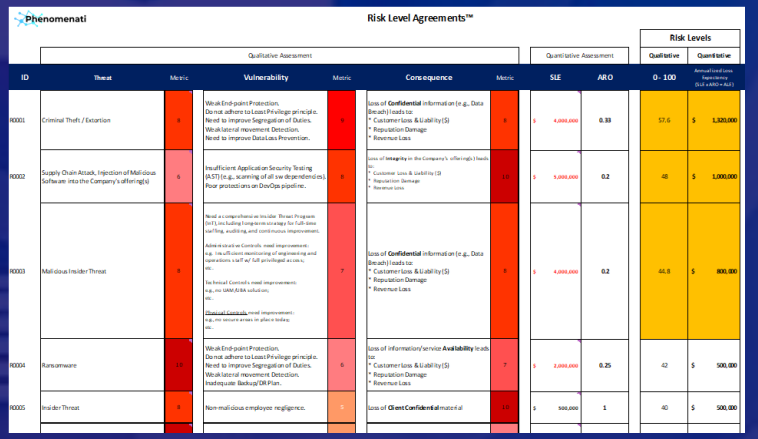
- Persistent Success of Social Engineering exploiting people and processes
- *End of Life* Components or Controls

Remediation

- Prioritize vulnerability remediation based upon Risk Assessment
- Corporate level Awareness and Risk Level Agreements™
- Anticipate vulnerabilities and account for them in system designs, procurement, and deployments
  - Inspect security profile of companies that provide you software-based systems
- Defensive Programming
  - Never trust calls to APIs; expect nefarious behavior and ruthlessly scrub incoming data to APIs
- Domain Primitives (“Secure By Design”, Manning Publishing)
  - Only make data (e.g., business objects) mutable when necessary;
  - Immutable should always be the default behavior

Targets \ Seriousness	Low	Medium	High
People	e.g., Ignorance e.g., Weak SFA (password) e.g., Bias toward being Helpful	e.g., Negligence, Misdelivery e.g., Blind Trust e.g., Naïve Arrogance	e.g., Ignore Obligations (Legal) e.g., Malicious Insider
Processes	e.g., Poorly Defined (Bus & IT) e.g., Asset & Config Mngmnt	e.g., Defined, but no Controls e.g., Help Desk	e.g., Short Cuts, circumvent Controls
Technology	e.g., Rush-to-Market e.g., Cost Pressures e.g., Weak Security Testing	e.g., Rush-to-Adopt e.g., Poorly Deployed e.g., No Risk Assessment (e.g., web app, email svr, desktop, phones/tablets, IOT)	e.g., Complexity Obfuscates Dependencies e.g., Lack of "Impact Analysis“ (e.g., RDP/VNC)





3:00

Fragility

What it is

- Reliance on single-points of failure, obsolete technologies, etc.
- An outcome of failing to address *criticalities* and *contingencies*
- The result of ignoring ***non-functional*** requirements (the "…ilities")

Examples

- Forrester's "Digital Fragility: The Ticking Time Bomb Within Enterprises"
- Massive Microsoft 365 outage caused by faulty Enterprise Configuration Service (ECS) deployment.
  - 23 July 2022 (this just happened)
  - https://www.bleepingcomputer.com/news/microsoft/massive-microsoft-365-outage-caused-by-faulty-ecs-deployment/

Remediation Examples

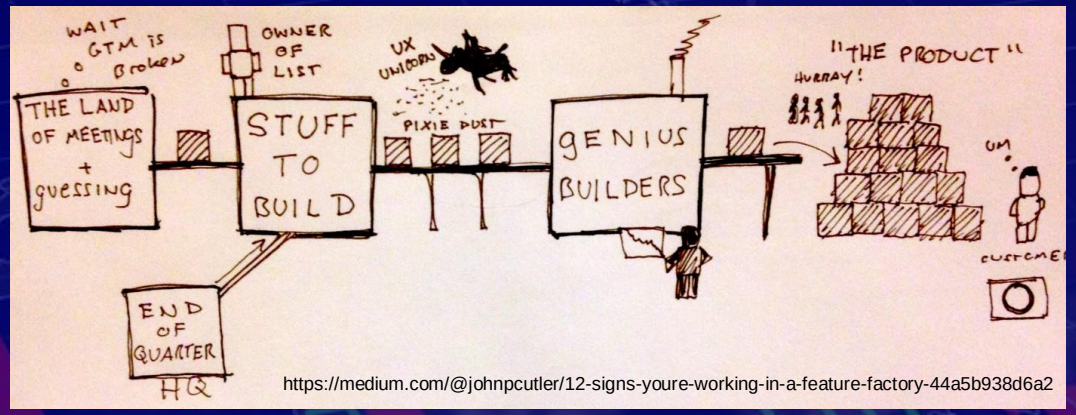
- "*Minimize the potential blast radius…*"
- Bulkheads (geographically [Geode], infrastructure deployments, code design)
  - Engage IT org; engage software engineers (architecture and design)
- Circuit Breakers, Fallbacks, Throttling
- Client-Side Load Balancing
  - But this increases Complexity and introduces a new Dependency [which is the required discovery service]

3:00

Urgency

What it is

- A market force that must be addressed, responsibly
- Fueled by Anxiety and Uncertainty…
  - and often an inability to cope with ***Imperfect Knowledge***
- A poor way to manage Risk



https://medium.com/@johnpcutler/12-signs-youre-working-in-a-feature-factory-44a5b938d6a2

Example

- Don't Become a Feature Factory (John Cutler)
  - "Features are implemented to close new deals. While not inherently wrong, the economic justifications are often flimsy (at best), and fail to account for the non-linear increase in product complexity….Low visibility for refactoring work and debt work-down. Low visibility for overall value delivery capabilities (such as resiliency)….Little appreciation for the health of the whole product as opposed to shiny new objects."
  - https://medium.com/@johnpcutler/12-signs-youre-working-in-a-feature-factory-44a5b938d6a2

Remediation

- Balance being *Responsive* with being *Responsible*
- Document the *Assumptions* and *Risks* associated with accelerated schedules and shortcuts
- Be "Agile", but not Stupid ("Agile 2"; Cliff Berg, Kurt Cagle, et al; Wiley & Sons; 2021)
- Evolve your project's use of agile methodologies
  - Start with Feature-Driven Development (FDD) to establish an initial user community and installed base
  - Migrate to Test-Driven Development (TDD) to increase the quality of your project's deliverables and keep your user base
  - Aim to arrive at Behavior-Driven Development (BDD) with Resiliency prioritized by the business owners as a top behavior

$$U = \frac{1}{t_R}$$

3:00

Simplicity

What it is

- A natural response to Entropy and emergent Complexity
- Different audiences have different needs for levels of detail

Example

- Ever heard these: "*Can we have a **common** architecture?*" "*Let's establish a **common** infrastructure.*"
- Over-reliance on a single **vendor**, or single **platform**, or **language** as a means to reduce *Complexity*
- 1990s…a Wall Street bank's CIO "standardized" on NT to **simplify** IT operations & maintenance
  - A zero-day vulnerability exploit in Windows NT compromised almost all of the bank's computers & nearly bankrupted the firm

www.soctaxonomy.com

Remediation

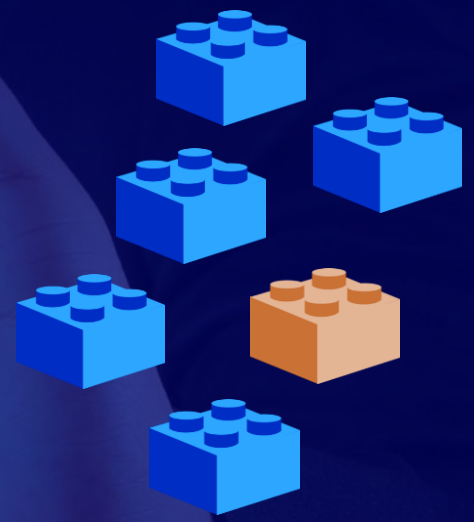
- Leverage *Analogies* and *Abstractions* to communicate with those who cannot cope with the complexity
- Iterative *Decomposition*… from Simple ***Concept Diagrams*** to Detailed ***Design Specifications***
- McKinsey Quarterly, "Organizations must build digital resilience to protect their most valuable information assets"
  - "There are no shortcuts or pat solutions. Indeed, any cybersecurity program for a sizable institution will involve hundreds of individual design and implementation decisions. Senior, cross-functional oversight is essential to avoid a mere patchwork of compromises that will undermine digital resilience. Given the stakes, nothing else will do."

3:00

Commodity

What it is

- The pervasive use of commonly available computing infrastructure, tools, and applications
- A path to realize *Simplicity* and *Efficiency*
- A path to *Predictability*, an attacker's dream
- Easily acquired, tested (probed for vulnerabilities), and ***exploited***



Examples

- Apache's Log4J vulnerability (CVE-2022-23302)
- Atlassian's recent vulnerabilities (CVE-2022-26136 & CVE-2022-26137)
- Apple's recent two zero-day vulnerabilities (CVE-2022-22674 & CVE-2022-22675)
- Google's recent Chrome (and Edge) vulnerability (CVE-2022-1096)

Remediation

- Commodity components work best with patterns of massive redundancy (*disposability*)
- Managed technological heterogeneity; varies attack surfaces, & eliminates single points of failure
- Use other operating systems (e.g. OpenVMS)
- Use other (non TCP/IP) network protocols (e.g., DECnet) to protect critical internal computer systems

3:00

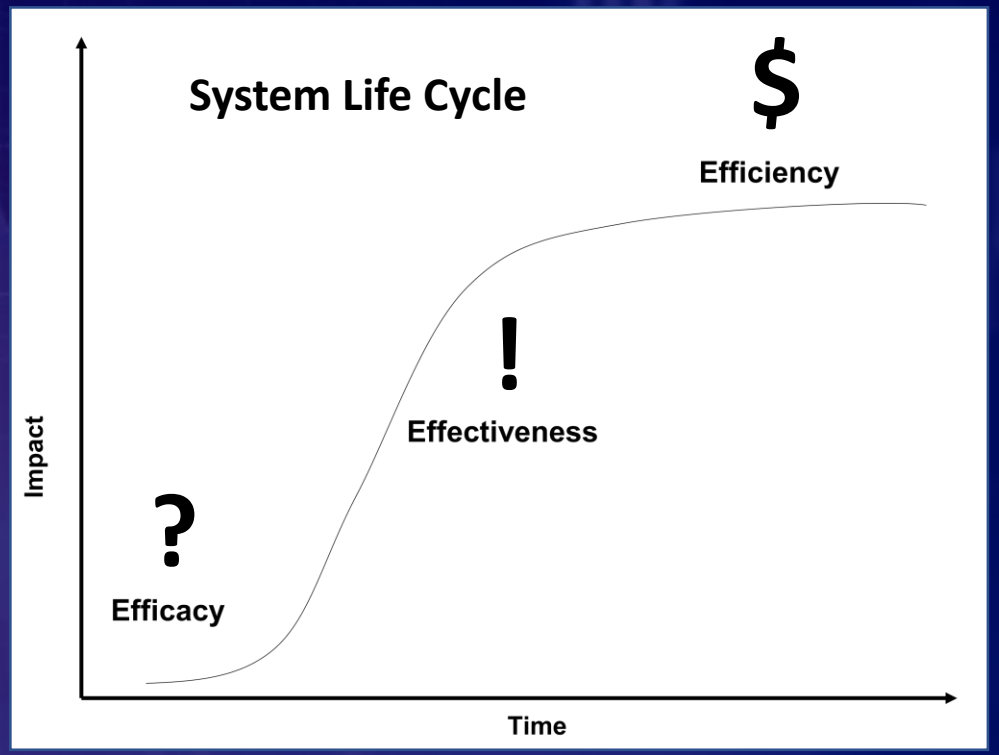
Efficiency

What it is

- An attribute of a ***mature*** system or component
- What comes AFTER ***Efficacy*** and ***Effectiveness***
- A market force that *directly undermines* Resiliency

Examples

- The first light bulb was not cheap to create
- The first laser printer required more than 1000 watts of power
- Data Center owners focus on *Availability* before efficient use of power or HVAC



Remediation

- Recognize ***where*** you are in the Life Cycle of the system
- Engineering Trades (Decisions) should be Transparent
  - e.g., Cost vs. Performance
- Geode pattern (in infrastructure -> “bring the compute to the data”)
  - Engage your cloud architects, systems administrators, IT org (system administrators & network administrators)

3:00

Fantasy

What it is

- **Failure** to perform Detailed ***Analysis*** and make the difficult Engineering ***Tradeoffs***
- "***Willful Ignorance***" of Fundamental ***Assumptions*** and Inherent ***Risks***

Examples

- Political Nirvana: **Free**, **Perfect**, and **Now**… the Conspiracy of Wishful Thinking
- Quest for a panacea
- The Reactive Manifesto
  - A bit of wishful thinking (wishes away stateful transactions, synchronous processing, deterministic workloads, etc.)

Remediation

- Investment in ***Feasibility*** Studies and PROOF-of-Concept ***Prototyping***
- Commitment to **RACI**: Responsibility, *Accountability*, Consulted, and Informed
- There is no one solution that will achieve Resiliency in a Contested Environment
- Cyber Resiliency is a Team Sport that demands a documented ***Strategy*** that is a corporate asset
  - Recognizing, Acknowledging and Addressing the Market Forces enumerated today
  - Documenting the Engineering Trades (decisions) made throughout the Life Cycle

https://en.wikipedia.org/wiki/Feasibility_study
https://en.wikipedia.org/wiki/Proof_of_concept

3:00

Summary

Cyber Resiliency is a team sport: engage business mgmt., IT operations, security experts, cloud architects, software engineering, procurement, end-users, and legal counsel

Develop a comprehensive corporate Resiliency Strategy

System-related Challenges

1. Cyber Entropy™
2. Complexity
3. Dependency
4. Vulnerability
5. Fragility

Acquisition-related Challenges

6. Urgency
7. Simplicity
8. Commodity
9. Efficiency
10. Fantasy

Phenomenati Leadership Team

Scott Foote

CISO, DPO, IT Auditor, cybersecurity executive, product executive, board member, board advisor, industry thought leader, with 35+ years experience designing security and privacy into digital transformation initiatives for a range of organizations – DEC, Oracle, OpenVision, Veritas, MITRE, several US Gov't organizations, and a series of high-tech startups.

Steve Foote

CISSP, industry analyst, serial entrepreneur, and accomplished software engineering executive.
35+ yrs experience leading organizations of 600+ engineers, applying his extensive knowledge of advanced software engineering, computer architecture, cyber security, mobile technologies, and enterprise applications for a variety of clients including finance, healthcare, pharmaceutical, commerce, law enforcement, intelligence, military, judicial, and US treasury

Questions?