# ECE 509: Cyber Security: Concept, Theory and Practices

Salim Hariri

Fall 2022

D2l.arizona.edu

# Today's Lecture Outline

- Threat Modeling
  - – STRIDE Methodology
  - – DREAD Methodology

# What will be covered in this class?

**Application Security & Resilience**

User and Web Applications | Mobile Platforms | Web Protocols

Encryption | Forensic Analysis | Insider Threats

**Operating System Security**

Basic Control Hijacking | Rootkits, Isolation

**Computer Networks and Protocols Security**

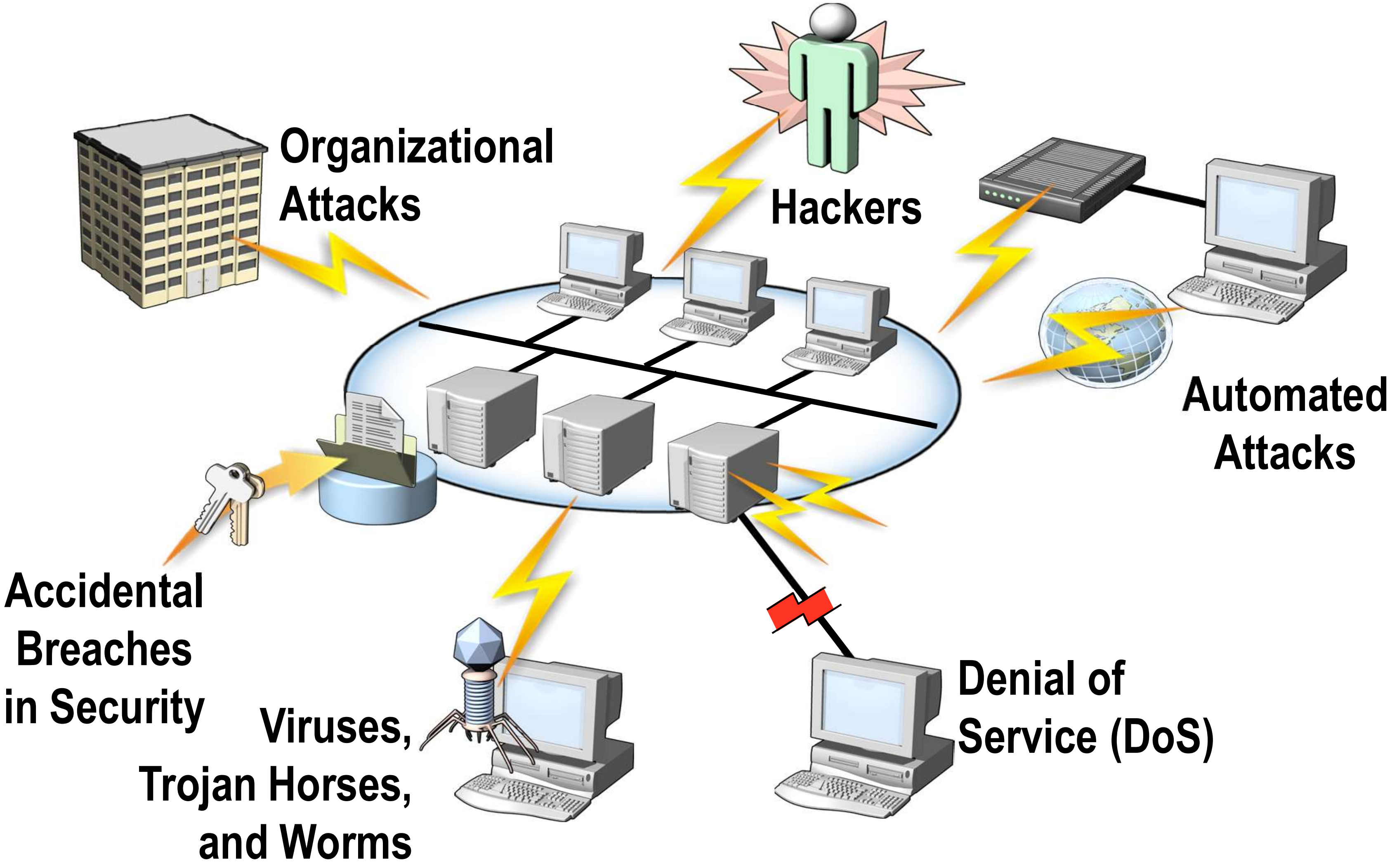
Computer Networks: Wireless | Wired

Communication Protocols: IP Based | Non IP Based

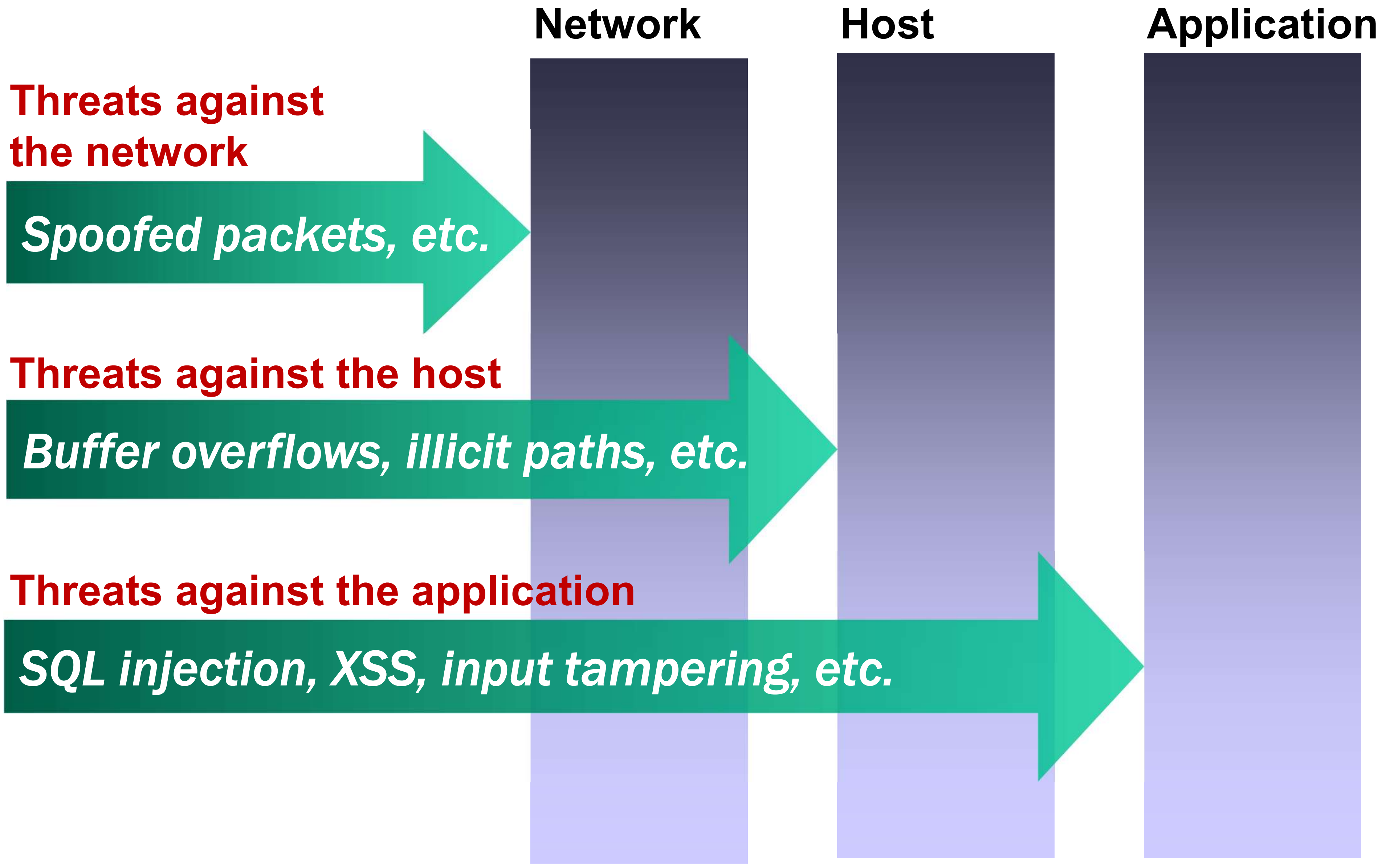
# Threat Modeling

- Threat modeling aims at finding security problems.
- Using a model means use abstraction to obtain a bigger picture, rather than the code itself
- Threat modeling involves answering the following four key questions:
  - What are you building?
  - What can go wrong?
  - What should you do about those things that can go wrong?
  - Did you do a decent job of analysis?

# What can go wrong?



# Types of Threats



**Network** **Host** **Application**

**Threats against the network**

*Spoofed packets, etc.*

**Threats against the host**

*Buffer overflows, illicit paths, etc.*

**Threats against the application**

*SQL injection, XSS, input tampering, etc.*

# Threats Against the Network

| Threat | Examples |
|---|---|
| Information gathering | Port scanning |
| | Using trace routing to detect network topologies |
| | Using broadcast requests to enumerate subnet hosts |
| Eavesdropping | Using packet sniffers to steal passwords |
| Denial of service (DoS) | SYN floods |
| | ICMP echo request floods |
| | Malformed packets |
| Spoofing | Packets with spoofed source addresses |

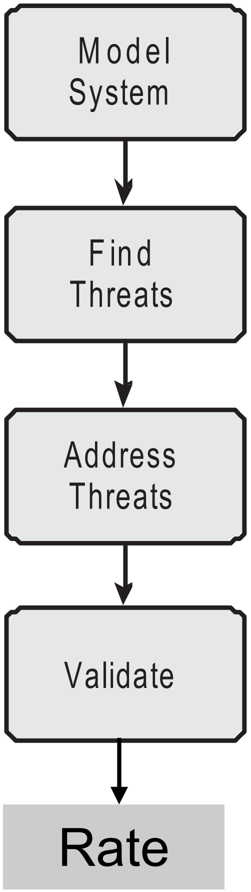
# Threats Against the Host

| Threat | Examples |
|---|---|
| Arbitrary code execution | Buffer overflows in ISAPI DLLs (e.g., MS01-033) |
| | Directory traversal attacks (MS00-078) |
| File disclosure | Malformed HTR requests (MS01-031) |
| | Virtualized UNC share vulnerability (MS00-019) |
| Denial of service (DoS) | Malformed SMTP requests (MS02-012) |
| | Malformed WebDAV requests (MS01-016) |
| | Malformed URLs (MS01-012) |
| | Brute-force file uploads |
| Unauthorized access | Resources with insufficiently restrictive ACLs |
| | Spoofing with stolen login credentials |
| Exploitation of open ports and protocols | Using NetBIOS and SMB to enumerate hosts |
| | Connecting remotely to SQL Server |

# Threats Against the Application

| Threat | Examples |
|---|---|
| SQL injection | Including a DROP TABLE command in text typed into an input field |
| Cross-site scripting | Using malicious client-side script to steal cookies |
| Hidden-field tampering | Maliciously changing the value of a hidden field |
| Eavesdropping | Using a packet sniffer to steal passwords and cookies from traffic on unencrypted connections |
| Session hijacking | Using a stolen session ID cookie to access someone else's session state |
| Identity spoofing | Using a stolen forms authentication cookie to pose as another user |
| Information disclosure | Allowing client to see a stack trace when an unhandled exception occurs |

# Threat Model Framework



1. Model the system/assets i.e. identify the assets and analyze them
2. Find threats using that model
3. Address threats using the approaches.
4. Validate your work for completeness and effectiveness.
5. Rate/Prioritize threats based on their impacts

# Step 1. Modelling of the System/ Asset

- There are three ways the term asset is commonly used in threat modeling:
  - Things attackers want
    - User passwords or keys
    - Social security numbers or other identifiers
    - Credit card numbers
    - Your confidential business data
  - Things you want to protect
    - Unlike the tangible things attackers want, many of these assets are intangibles
  - Stepping stones to either of these
    - For example, every computer has CPU and storage that an attacker can use
- Software architecture diagrams, UML diagrams, and attacker intention understanding can be used for asset modelling

# Step 2. Find Threats

- In this step the security expert will identify the threats that can be exploited to target the different assets and systems identified
- Device and system domain knowledge is used to identify the threats targeting the devices
- Datasets like the national vulnerability database, and MITRE CVE are used to identify the threats

NVD: https://nvd.nist.gov

MITRE CVE:
https://cve.mitre.org/about/cve_and_nvd_relationship.html

# Step 2. Formal methods for Threat Identification

- Method 1: Threat List
  - Create a list of possible threats
  - Identify the threats that will target the concerned system

- Method 2: STRIDE
  - Categorized list of threat types
  - Identify threats by type/category

- Method 3: Threat trees
  - Root nodes represent attacker’s goals

# STRIDE

- STRIDE is a mnemonic for things that go wrong in security.
  - **Spoofing** is pretending to be something or someone you're not.
  - **Tampering** is modifying something you're not supposed to modify.
    - It can include packets on the wire (or wireless), bits on disk, or the bits in memory.
  - **Repudiation** means claiming you didn't do something (regardless of whether you did or not).
  - **Denial of Service** are attacks designed to prevent a system from providing service, including by crashing it, making it unusably slow, or filling all its storage.
  - **Information Disclosure** is about exposing information to people who are not authorized to see it.
  - **Elevation of Privilege** is when a program or user is technically able to do things that they're not supposed to do.

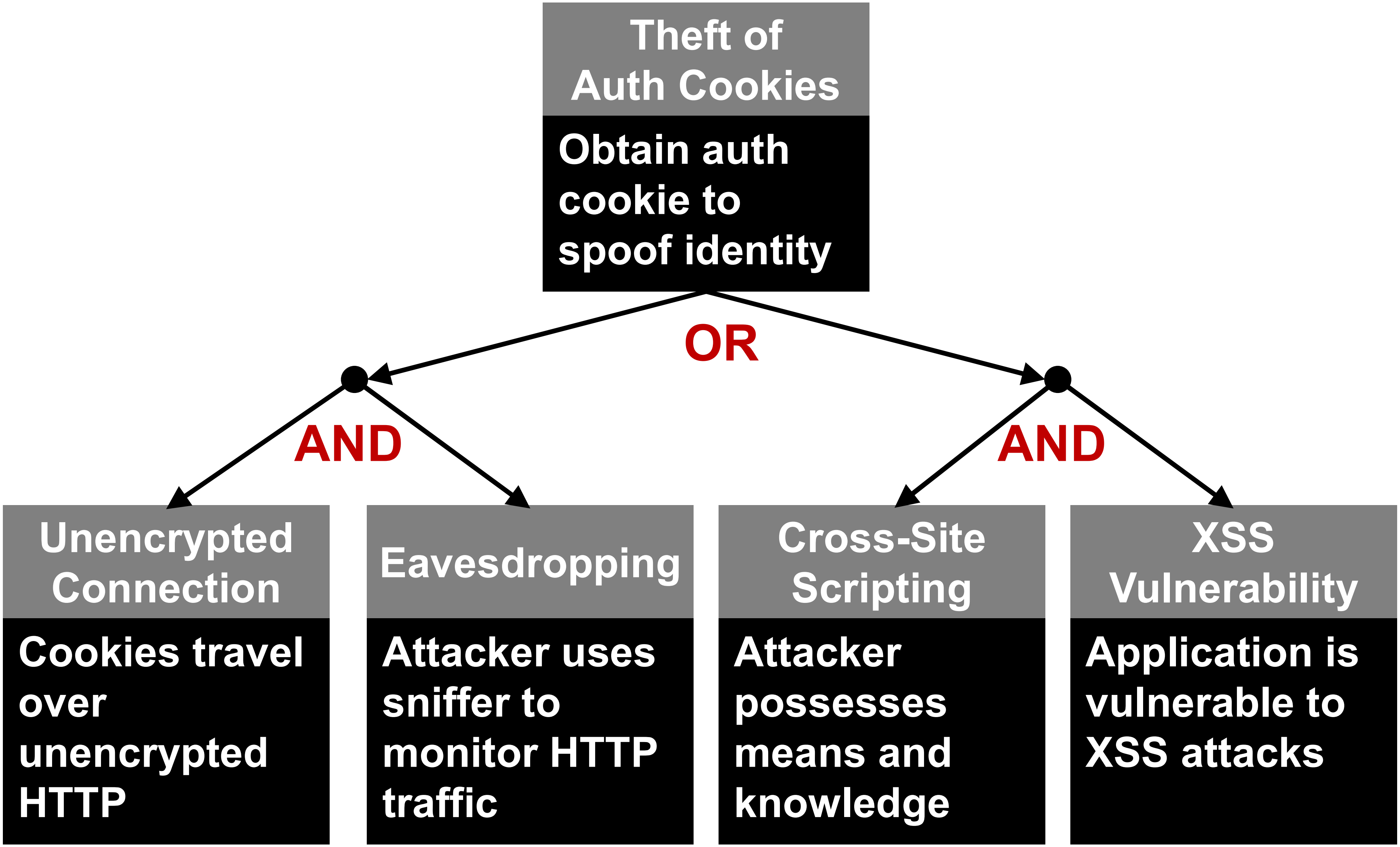
# Attack Trees

- Attack Tree
  - There are three ways you can use attack trees to enumerate threats:
    - You can use an attack tree someone else created to help you find threats.
    - You can create a tree to help you think through threats for a project you’re working on.
    - Or you can create trees with the intent that others will use them
  - Once you’ve modeled your system with a DFD or other diagram, you use an attack tree to analyze it
  - Creating New Attack Tree
    - 1 Decide on a representation (AND, OR, etc.)
    - 2. Create a root node.
    - 3. Create subnodes.
    - 4. Consider completeness.
    - 5. Prune the tree.
    - 6. Check the presentation.

# Attack Tree – Creating a root node

- The root node can be the component that prompts the analysis, or an adversary’s goal.
- If the root node is a component, the subnodes should be labeled with what can go wrong for the node.
- If the root node is an attacker goal, consider ways to achieve that goal. Each alternative way to achieve the goal should be drawn in as a subnode
- Some possible structures for first-level subnodes include:
  - Attacking a system:
    - physical access
    - subvert software
    - subvert a person
  - Attacking a system via:
    - People
    - Process
    - Technology

# Threat Trees: Goal Root Node Example


Theft of Auth Cookies
Obtain auth cookie to spoof identity
OR
AND
AND
Unencrypted Connection
Cookies travel over unencrypted HTTP
Eavesdropping
Attacker uses sniffer to monitor HTTP traffic
Cross-Site Scripting
Attacker possesses means and knowledge
XSS Vulnerability
Application is vulnerable to XSS attacks

# Step 3. Address Threats

- Address the security mechanisms to counter or prevent the threats identified in step 3
- Domain specific knowledge is used to address these threats
- The NVD and MITRE CVE list provide mitigation techniques for some of the attacks

# How to Address Each Threat?

- **Mitigating threats** aim at making it harder to exploit a threat.
  - Requiring passwords to control who can log in mitigates the threat of spoofing.
- **Eliminating threats** is almost always achieved by eliminating features
  - If you have a threat that someone will access the administrative function /url/admin
  - You can eliminate it by removing the interface, handling administration through the command line.
- **Transferring threats** by letting someone or something else handle the risk.
  - For example, you could pass authentication threats to the operating system, or trust boundary enforcement to a firewall product.
  - You can also transfer risk to customers, by asking them to click through lots of hard-to-understand dialogs before they can do the work they need to do.
- **Accepting the risk** of the identified threats.
  - For most organizations, searching everyone on the way in and out of the building is not worth the expense and job satisfaction of those workers.
  - Other organizations such as government agencies take a different approach
  - The cost of preventing someone from inserting a back door in the motherboard is expensive, so you might choose to accept the risk.

# Addressing Spoofing Threats

- Table 1-1 and the list that follows show targets of spoofing, mitigation strategies that address spoofing, and techniques to implement those mitigations

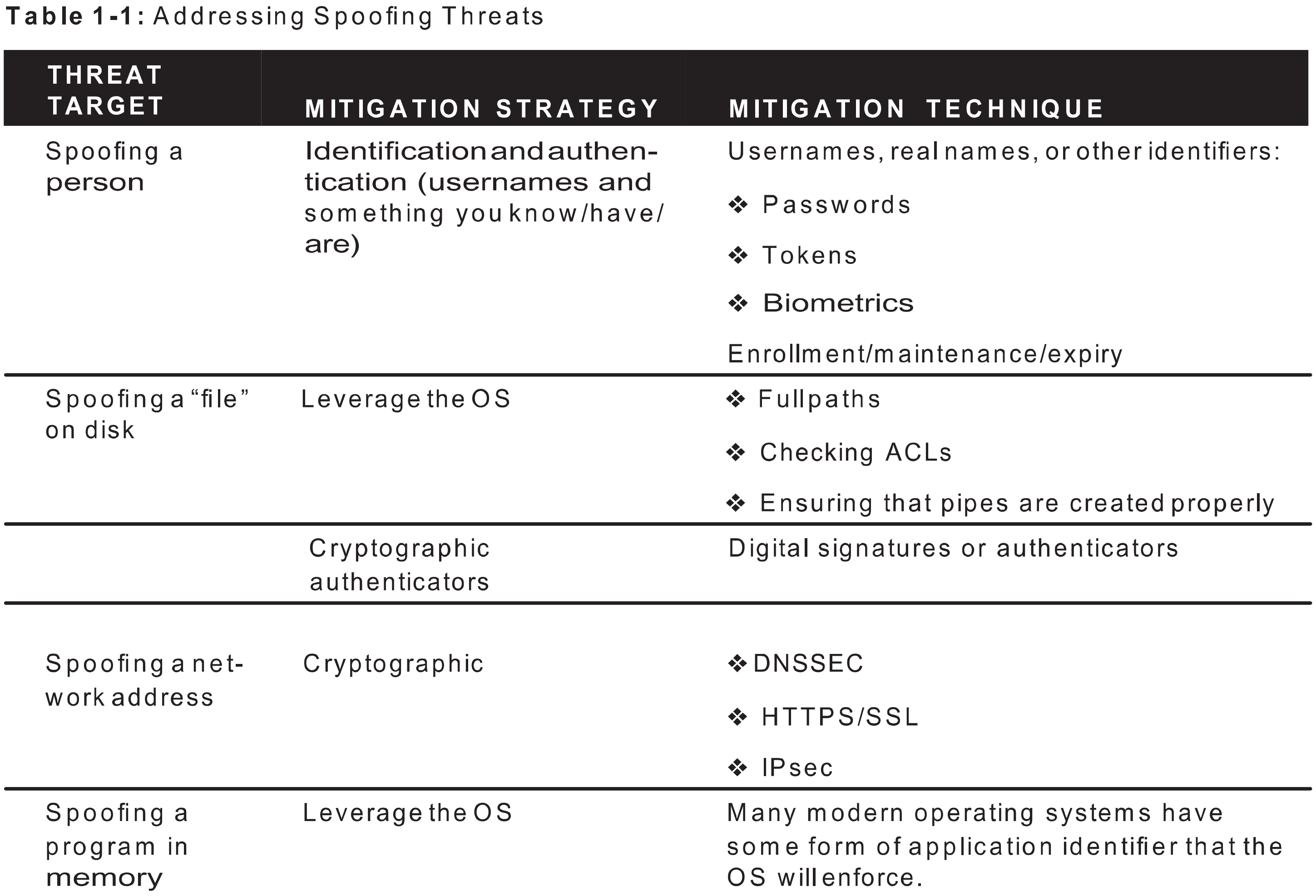


**Table 1-1:** Addressing Spoofing Threats

| THREAT TARGET | MITIGATION STRATEGY | MITIGATION TECHNIQUE |
|---|---|---|
| Spoofing a person | Identification and authentication (usernames and something you know/have/are) | Usernames, real names, or other identifiers:<br>❖ Passwords<br>❖ Tokens<br>❖ Biometrics<br>Enrollment/maintenance/expiry |
| Spoofing a "file" on disk | Leverage the OS | ❖ Full paths<br>❖ Checking ACLs<br>❖ Ensuring that pipes are created properly |
| | Cryptographic authenticators | Digital signatures or authenticators |
| Spoofing a network address | Cryptographic | ❖ DNSSEC<br>❖ HTTPS/SSL<br>❖ IPsec |
| Spoofing a program in memory | Leverage the OS | Many modern operating systems have some form of application identifier that the OS will enforce. |

# Addressing Tampering Threats

**Table 1-2:** Addressing Tampering Threats

| THREAT TARGET | MITIGATION STRATEGY | MITIGATION TECHNIQUE |
|---|---|---|
| Tampering with a file | Operating system | ACLs |
| | Cryptographic | ❖ Digital Signatures<br>❖ Keyed MAC |
| Racing to create a file (tampering with the file system) | Using a directory that's protected from arbitrary user tampering | ACLs<br>Using private directory structures<br>(Randomizing your file names just makes it annoying to execute the attack.) |
| Tampering with a network packet | Cryptographic | ❖ HTTPS/SSL<br>❖ IPsec |
| | Anti-pattern | Network isolation (See note on network isolation anti-pattern.) |

# Addressing Repudiation Threats

- Addressing repudiation focuses on ensuring that your system is designed to log and ensuring that those logs are preserved and protected.
- Table 1-3 and the list that follows show targets of repudiation, mitigation strategies that address repudiation, and techniques to implement those mitigations.

**Table 1-3:** Addressing Repudiation Threats

| THREAT TARGET | MITIGATION STRATEGY | MITIGATION TECHNIQUE |
|---|---|---|
| No logs means you can't prove anything. | Log | Be sure to log all the security-relevant information. |
| Logs come under attack | Protect your logs. | ❖ Send over the network.<br>❖ ACL |
| Logs as a channel for attack | Tightly specified logs | Documenting log design early in the development process |

# Addressing Information Disclosure Threats

**Table 1-4:** Addressing Information Disclosure Threats

| THREAT TARGET | MITIGATION STRATEGY | MITIGATION TECHNIQUE |
|---|---|---|
| Network monitoring | Encryption | ❖ HTTPS/SSL<br>❖ IPsec |
| Directory or filename (for example layoff-letters/adamshostack.docx) | Leverage the OS. | ACLs |
| File contents | Leverage the OS. | ACLS |
| | Cryptography | File encryption such as PGP, disk encryption (FileVault, BitLocker) |
| API information disclosure | Design | Careful design control<br>Consider pass by reference or value. |

# Addressing Denial of Service Threats

**Table 1-5:** Addressing Denial of Service Threats

| THREAT TARGET | MITIGATION STRATEGY | MITIGATION TECHNIQUE |
|---|---|---|
| Network flooding | Look for exhaustible resources. | ❖ Elastic resources<br>❖ Work to ensure attacker resource consumption is as high as or higher than yours. |
| | | Network ACLS |
| Program resources | Careful design | Elastic resource management, proof of work |
| | Avoid multipliers. | Look for places where attackers can multiply CPU consumption on your end with minimal effort on their end: Do something to require work or enable distinguishing attackers, such as client does crypto first or login before large work factors (of course, that can't mean that logins are unencrypted). |
| System resources | Leverage the OS. | Use OS settings. |

# Addressing Elevation of Service Threats -1

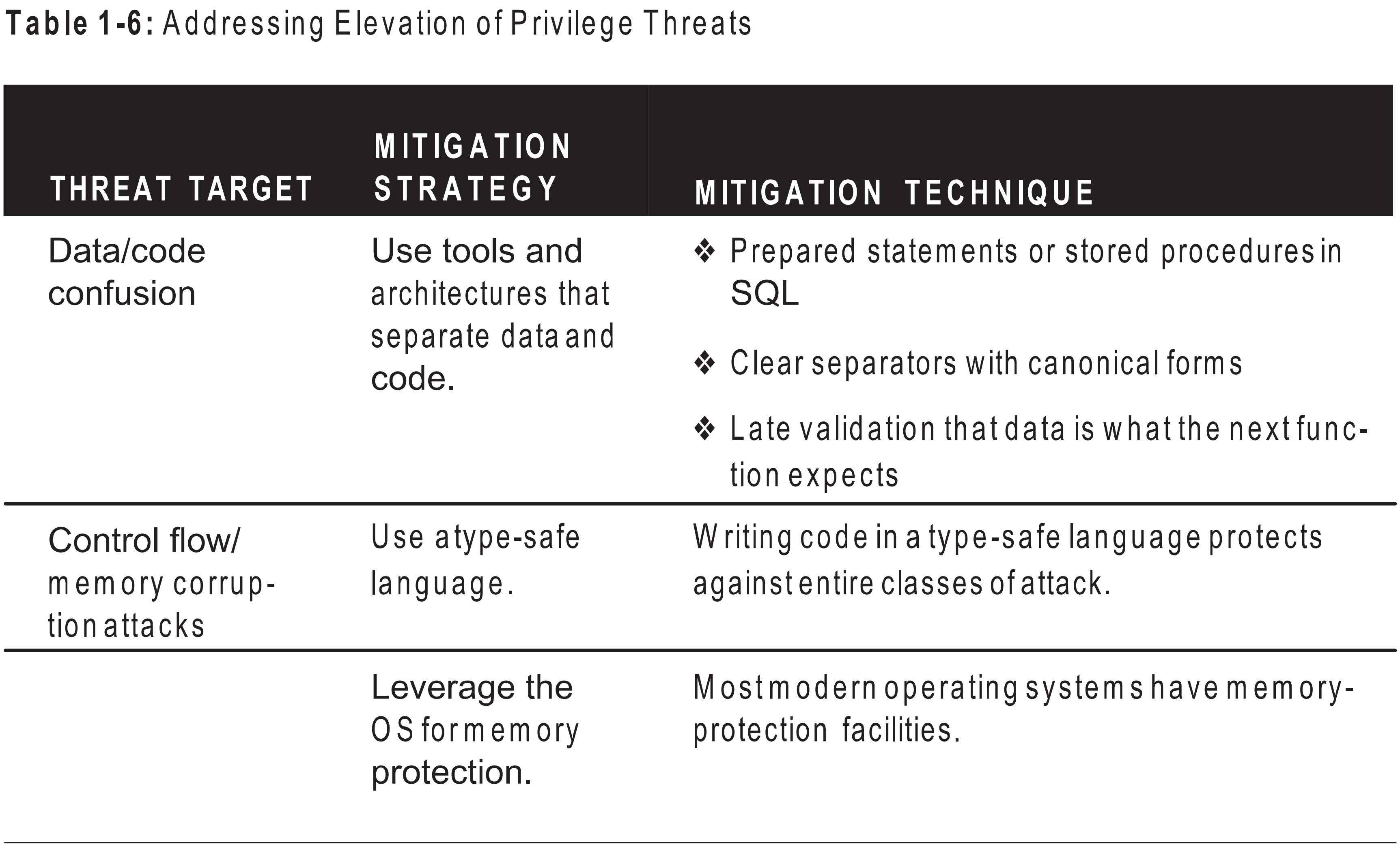


**Table 1-6:** Addressing Elevation of Privilege Threats

| THREAT TARGET | MITIGATION STRATEGY | MITIGATION TECHNIQUE |
|---|---|---|
| Data/code confusion | Use tools and architectures that separate data and code. | ❖ Prepared statements or stored procedures in SQL<br>❖ Clear separators with canonical forms<br>❖ Late validation that data is what the next function expects |
| Control flow/memory corruption attacks | Use a type-safe language. | Writing code in a type-safe language protects against entire classes of attack. |
| | Leverage the OS for memory protection. | Most modern operating systems have memory-protection facilities. |

# Addressing Elevation of Service Threats -2

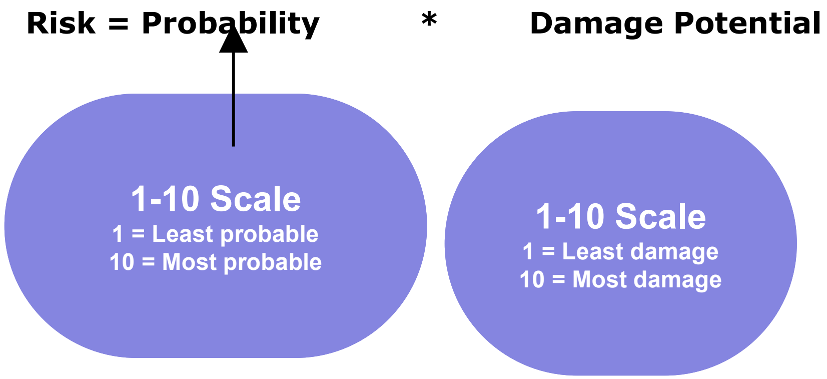
| THREAT TARGET | MITIGATION STRATEGY | MITIGATION TECHNIQUE |
|---|---|---|
|  | Use the sandbox. | ❖ Modern operating systems support sandboxing in various ways (AppArmor on Linux, AppContainer or the MOICE pattern on Windows, Sandboxlib on Mac OS).<br>❖ Don't run as the "nobody" account, create a new one for each app. Postfix and QMail are examples of the good pattern of one account per function. |
| Command injection attacks | Be careful. | ❖ Validate that your input is the size and form you expect.<br>❖ Don't sanitize. Log and then throw it away if it's weird. |

# Step 4. Validate the Threat model

- Validate the addressing mechanisms identified in step 3
- Simulation based approaches or actual testing will be used to perform validation

# Step 5. Rate

- Simple model:



Risk = Probability * Damage Potential

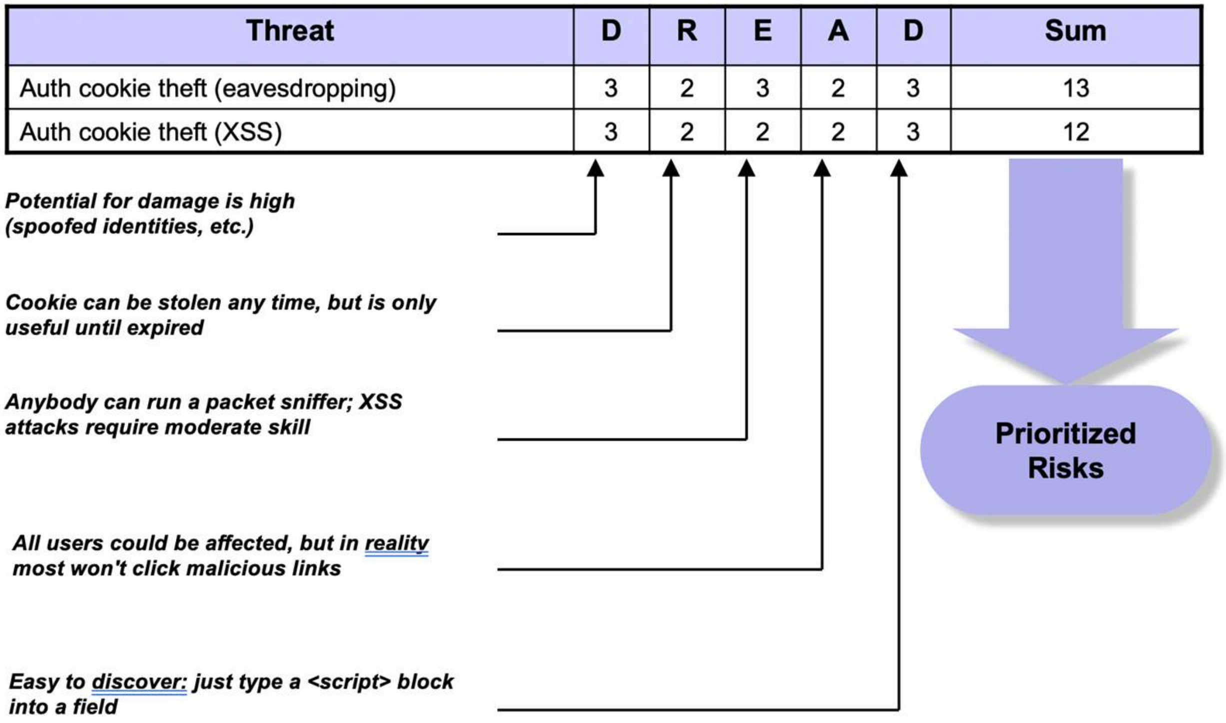
1-10 Scale
1 = Least probable
10 = Most probable

1-10 Scale
1 = Least damage
10 = Most damage

- Dread Model
  - Greater granularization of threat potential
  - Rates (prioritizes) each threat on scale of 1-15
  - Developed and widely used by Microsoft

# DREAD

- ***D*** **Damage potential**
- ***R*** **Reproducibility**
- ***E*** **Exploitability**
- ***A*** **Affected users**
- ***D*** **Discoverability**



| Threat | D | R | E | A | D | Sum |
|---|---|---|---|---|---|---|
| Auth cookie theft (eavesdropping) | 3 | 2 | 3 | 2 | 3 | 13 |
| Auth cookie theft (XSS) | 3 | 2 | 2 | 2 | 3 | 12 |

*Potential for damage is high (spoofed identities, etc.)*

*Cookie can be stolen any time, but is only useful until expired*

*Anybody can run a packet sniffer; XSS attacks require moderate skill*

*All users could be affected, but in reality most won't click malicious links*

*Easy to discover: just type a <script> block into a field*

# Practical example of Threat Modelling: Smart Speaker

- Smart Speakers are IoT devices that connect to Wi-Fi network to play music based on voiced commands

- Example: Google Home, Alexa

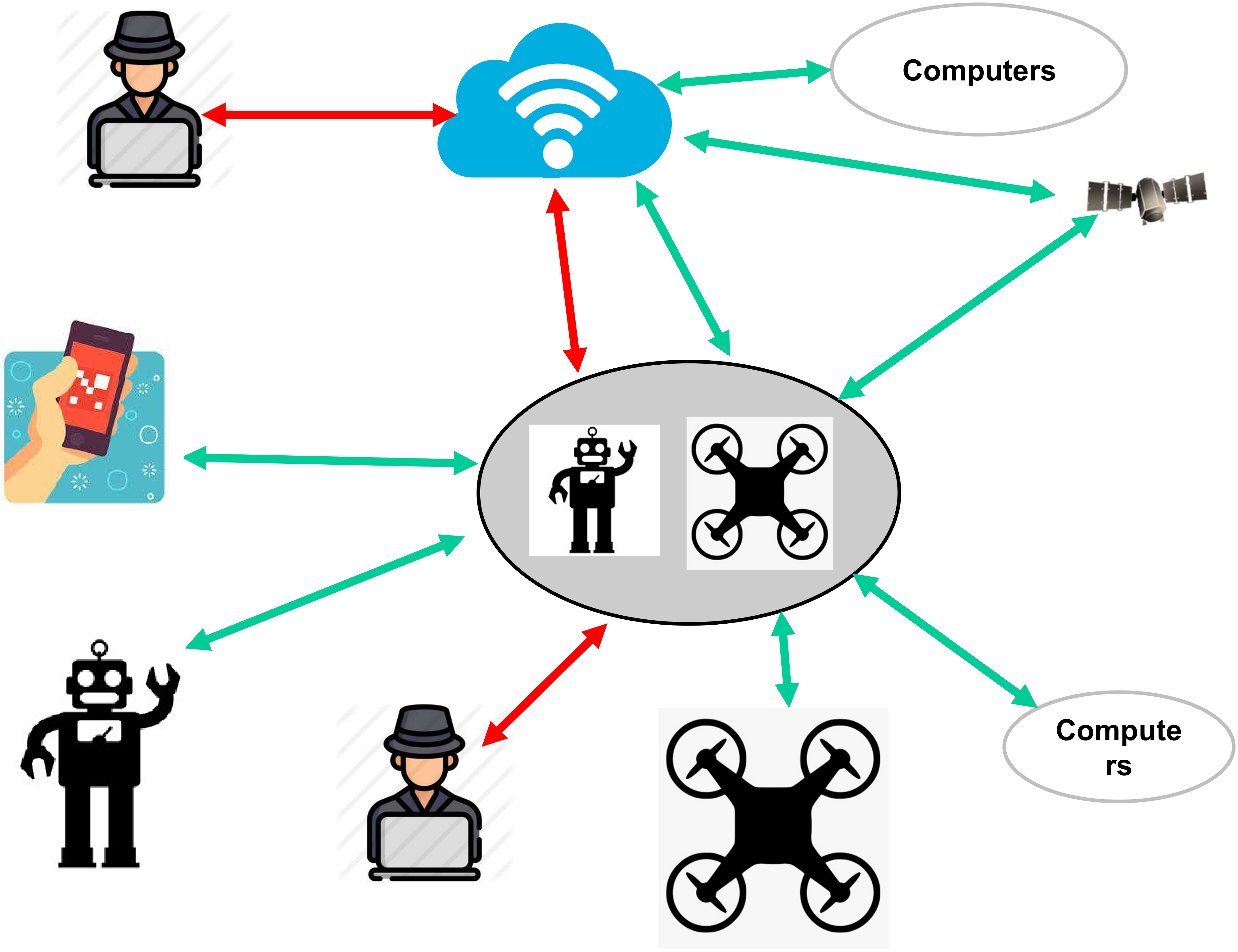
# Summary

- Without threat modelling, protecting yourself is like “shooting in the dark”
- You need expertise in understanding most common attacks – read security bulletins
- Developers must learn and use secure coding practices
  - Learn some crypto too
- Assume you are vulnerable, prove you are not

# References

- http://msdn.microsoft.com/security/securecode/threatmodeling/default.aspx
- http://sec.cs.kent.ac.uk/cms2004/Program/CMS2004final/p4a6.pdf
- http://cpd.ogi.edu/seminars04/hickmanthreatmodeling.pdf
- Reference for threat modeling tool:

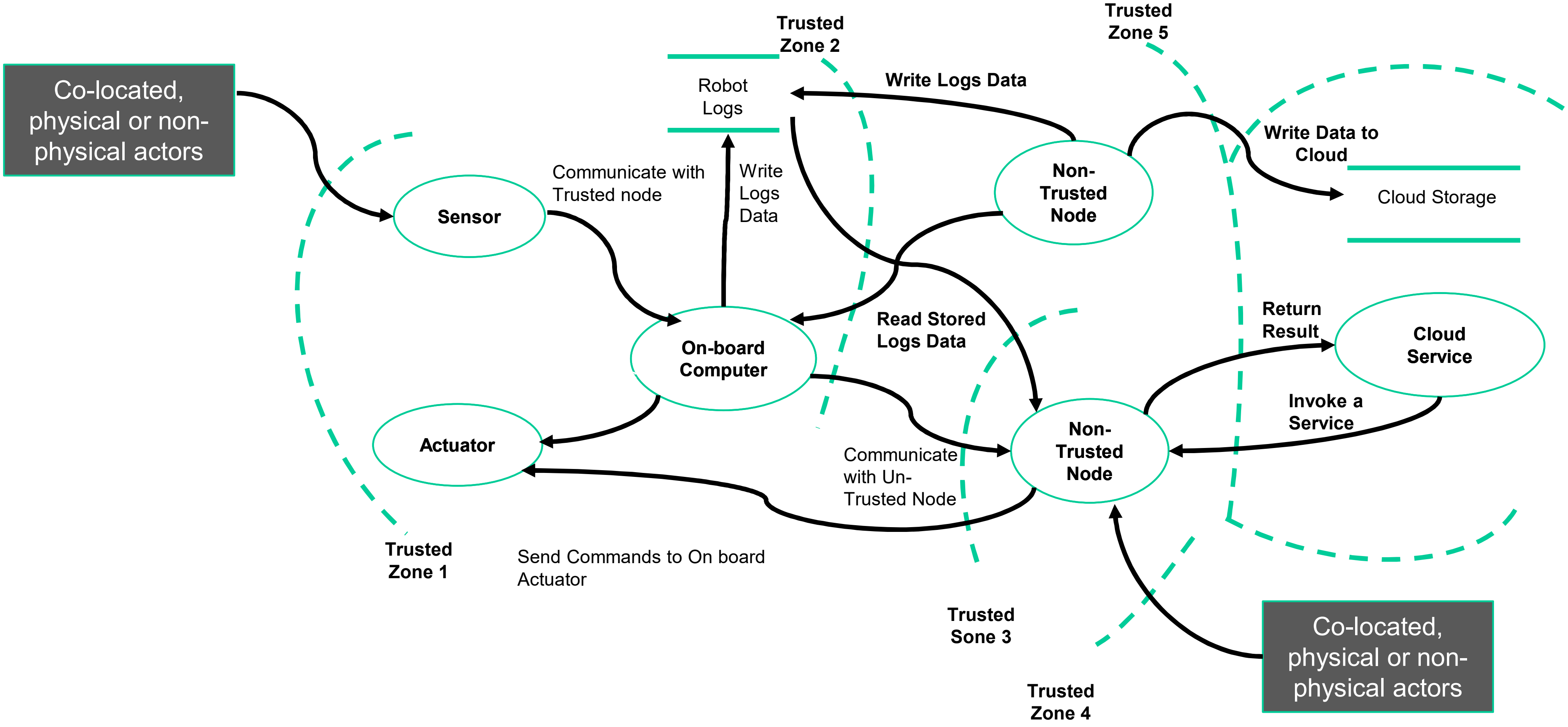
  http://thesource.ofallevil.com/downloads/details.aspx?FamilyID=28a7e041-8909-4084-8b05-06c3135e2a16&displaylang=en


Computers
Compute
rs

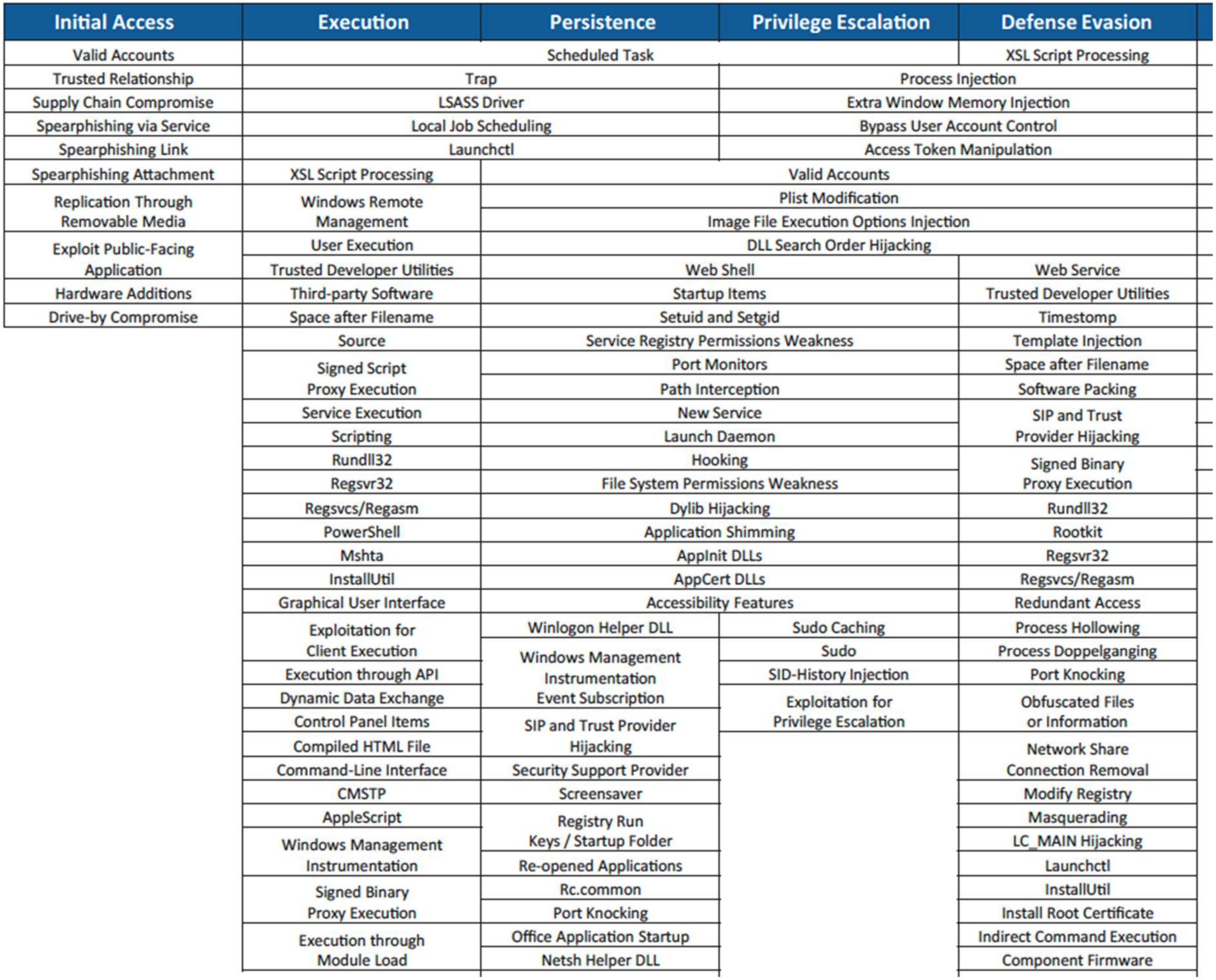
# Potential Services/Applications

- Inter-Component Communication
- Long-Range Communication (Cellular, Radio, Satellite, etc.)
- Remote Application Interface
- OS & Kernel
- Component Compromises
- Configuration Management
- Data Storage (File System)
- Data Logs
- Sensors
- Actuators
- Communications
- Client Application
- Cloud Integration
- Software Deployment
- Credentials, PKI and Secrets

# THREAT MODELING OF AUTONOMOUS VEHICLE

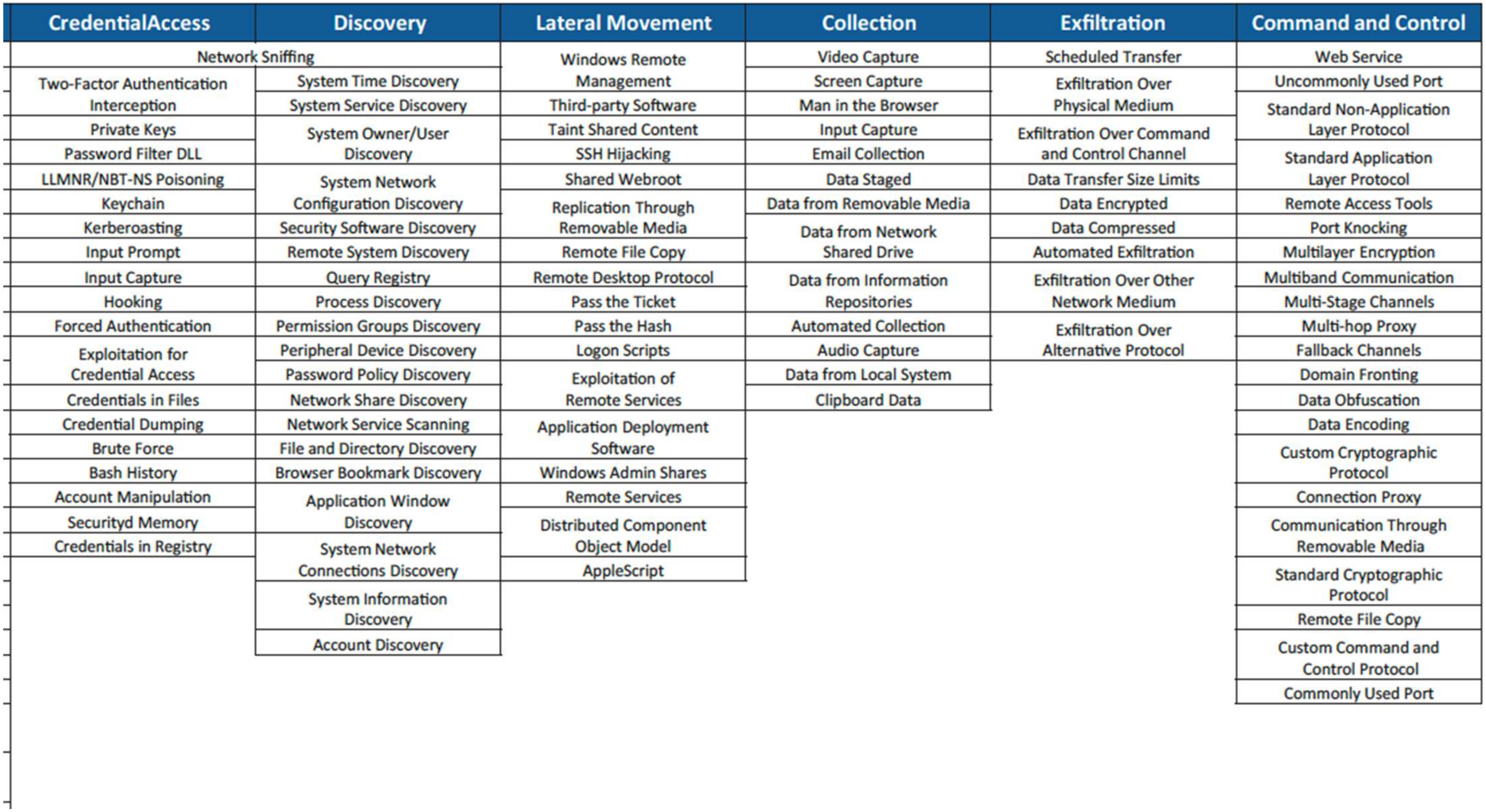


# MITRE Adversarial Tactics, Techniques and Common Knowledge (ATT&CK)



<table>
<tr><th>Initial Access</th><th>Execution</th><th>Persistence</th><th>Privilege Escalation</th><th>Defense Evasion</th></tr>
<tr><td>Valid Accounts</td><td colspan="3">Scheduled Task</td><td>XSL Script Processing</td></tr>
<tr><td>Trusted Relationship</td><td colspan="2">Trap</td><td colspan="2">Process Injection</td></tr>
<tr><td>Supply Chain Compromise</td><td colspan="2">LSASS Driver</td><td colspan="2">Extra Window Memory Injection</td></tr>
<tr><td>Spearphishing via Service</td><td colspan="2">Local Job Scheduling</td><td colspan="2">Bypass User Account Control</td></tr>
<tr><td>Spearphishing Link</td><td colspan="2">Launchctl</td><td colspan="2">Access Token Manipulation</td></tr>
<tr><td>Spearphishing Attachment</td><td>XSL Script Processing</td><td colspan="3">Valid Accounts</td></tr>
<tr><td rowspan="2">Replication Through Removable Media</td><td rowspan="2">Windows Remote Management</td><td colspan="3">Plist Modification</td></tr>
<tr><td colspan="3">Image File Execution Options Injection</td></tr>
<tr><td rowspan="2">Exploit Public-Facing Application</td><td>User Execution</td><td colspan="3">DLL Search Order Hijacking</td></tr>
<tr><td>Trusted Developer Utilities</td><td colspan="2">Web Shell</td><td>Web Service</td></tr>
<tr><td>Hardware Additions</td><td>Third-party Software</td><td colspan="2">Startup Items</td><td>Trusted Developer Utilities</td></tr>
<tr><td>Drive-by Compromise</td><td>Space after Filename</td><td colspan="2">Setuid and Setgid</td><td>Timestomp</td></tr>
<tr><td></td><td>Source</td><td colspan="2">Service Registry Permissions Weakness</td><td>Template Injection</td></tr>
<tr><td></td><td rowspan="2">Signed Script Proxy Execution</td><td colspan="2">Port Monitors</td><td>Space after Filename</td></tr>
<tr><td></td><td colspan="2">Path Interception</td><td>Software Packing</td></tr>
<tr><td></td><td>Service Execution</td><td colspan="2">New Service</td><td rowspan="2">SIP and Trust Provider Hijacking</td></tr>
<tr><td></td><td>Scripting</td><td colspan="2">Launch Daemon</td></tr>
<tr><td></td><td>Rundll32</td><td colspan="2">Hooking</td><td rowspan="2">Signed Binary Proxy Execution</td></tr>
<tr><td></td><td>Regsvr32</td><td colspan="2">File System Permissions Weakness</td></tr>
<tr><td></td><td>Regsvcs/Regasm</td><td colspan="2">Dylib Hijacking</td><td>Rundll32</td></tr>
<tr><td></td><td>PowerShell</td><td colspan="2">Application Shimming</td><td>Rootkit</td></tr>
<tr><td></td><td>Mshta</td><td colspan="2">AppInit DLLs</td><td>Regsvr32</td></tr>
<tr><td></td><td>InstallUtil</td><td colspan="2">AppCert DLLs</td><td>Regsvcs/Regasm</td></tr>
<tr><td></td><td>Graphical User Interface</td><td colspan="2">Accessibility Features</td><td>Redundant Access</td></tr>
<tr><td></td><td rowspan="2">Exploitation for Client Execution</td><td>Winlogon Helper DLL</td><td>Sudo Caching</td><td>Process Hollowing</td></tr>
<tr><td></td><td rowspan="3">Windows Management Instrumentation Event Subscription</td><td>Sudo</td><td>Process Doppelganging</td></tr>
<tr><td></td><td>Execution through API</td><td>SID-History Injection</td><td>Port Knocking</td></tr>
<tr><td></td><td>Dynamic Data Exchange</td><td rowspan="2">Exploitation for Privilege Escalation</td><td rowspan="2">Obfuscated Files or Information</td></tr>
<tr><td></td><td>Control Panel Items</td><td rowspan="2">SIP and Trust Provider Hijacking</td></tr>
<tr><td></td><td>Compiled HTML File</td><td></td><td rowspan="2">Network Share Connection Removal</td></tr>
<tr><td></td><td>Command-Line Interface</td><td>Security Support Provider</td><td></td></tr>
<tr><td></td><td>CMSTP</td><td>Screensaver</td><td></td><td>Modify Registry</td></tr>
<tr><td></td><td>AppleScript</td><td rowspan="2">Registry Run Keys / Startup Folder</td><td></td><td>Masquerading</td></tr>
<tr><td></td><td rowspan="2">Windows Management Instrumentation</td><td></td><td>LC_MAIN Hijacking</td></tr>
<tr><td></td><td>Re-opened Applications</td><td></td><td>Launchctl</td></tr>
<tr><td></td><td rowspan="2">Signed Binary Proxy Execution</td><td>Rc.common</td><td></td><td>InstallUtil</td></tr>
<tr><td></td><td>Port Knocking</td><td></td><td>Install Root Certificate</td></tr>
<tr><td></td><td rowspan="2">Execution through Module Load</td><td>Office Application Startup</td><td></td><td>Indirect Command Execution</td></tr>
<tr><td></td><td>Netsh Helper DLL</td><td></td><td>Component Firmware</td></tr>
</table>

# MITRE Adversarial Tactics, Techniques and Common Knowledge (ATT&CK)



| CredentialAccess | Discovery | Lateral Movement | Collection | Exfiltration | Command and Control |
|---|---|---|---|---|---|
| Network Sniffing | | Windows Remote Management | Video Capture | Scheduled Transfer | Web Service |
| Two-Factor Authentication Interception | System Time Discovery | Third-party Software | Screen Capture | Exfiltration Over Physical Medium | Uncommonly Used Port |
| Private Keys | System Service Discovery | Taint Shared Content | Man in the Browser | Exfiltration Over Command and Control Channel | Standard Non-Application Layer Protocol |
| Password Filter DLL | System Owner/User Discovery | SSH Hijacking | Input Capture | Data Transfer Size Limits | Standard Application Layer Protocol |
| LLMNR/NBT-NS Poisoning | System Network Configuration Discovery | Shared Webroot | Email Collection | Data Encrypted | Remote Access Tools |
| Keychain | Security Software Discovery | Replication Through Removable Media | Data Staged | Data Compressed | Port Knocking |
| Kerberoasting | Remote System Discovery | Remote File Copy | Data from Removable Media | Automated Exfiltration | Multilayer Encryption |
| Input Prompt | Query Registry | Remote Desktop Protocol | Data from Network Shared Drive | Exfiltration Over Other Network Medium | Multiband Communication |
| Input Capture | Process Discovery | Pass the Ticket | Data from Information Repositories | Exfiltration Over Alternative Protocol | Multi-Stage Channels |
| Hooking | Permission Groups Discovery | Pass the Hash | Automated Collection | | Multi-hop Proxy |
| Forced Authentication | Peripheral Device Discovery | Logon Scripts | Audio Capture | | Fallback Channels |
| Exploitation for Credential Access | Password Policy Discovery | Exploitation of Remote Services | Data from Local System | | Domain Fronting |
| Credentials in Files | Network Share Discovery | Application Deployment Software | Clipboard Data | | Data Obfuscation |
| Credential Dumping | Network Service Scanning | Windows Admin Shares | | | Data Encoding |
| Brute Force | File and Directory Discovery | Remote Services | | | Custom Cryptographic Protocol |
| Bash History | Browser Bookmark Discovery | Distributed Component Object Model | | | Connection Proxy |
| Account Manipulation | Application Window Discovery | AppleScript | | | Communication Through Removable Media |
| Securityd Memory | System Network Connections Discovery | | | | Standard Cryptographic Protocol |
| Credentials in Registry | System Information Discovery | | | | Remote File Copy |
| | Account Discovery | | | | Custom Command and Control Protocol |
| | | | | | Commonly Used Port |

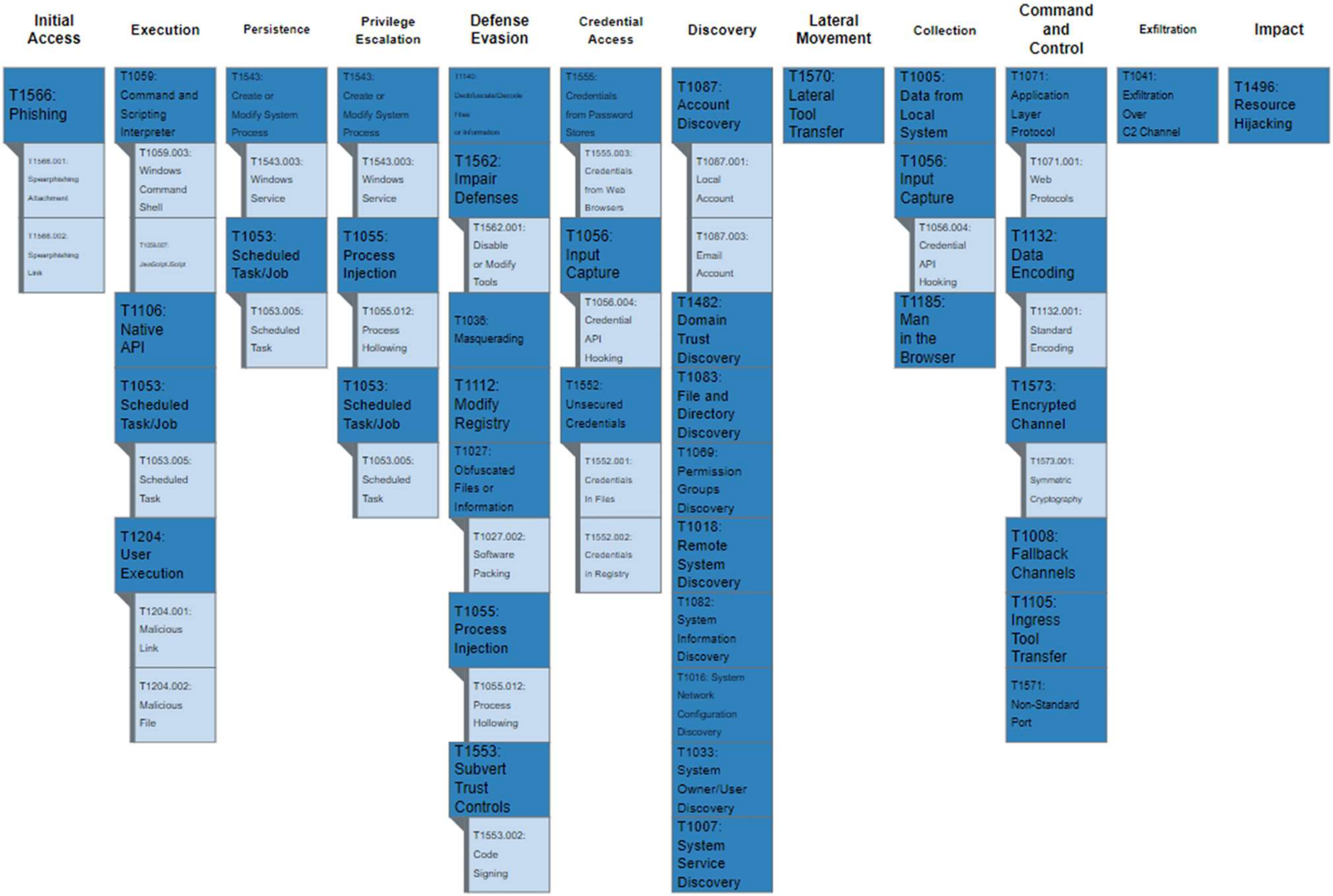
# TrickBot

TrickBot is an advanced Trojan that malicious actors spread primarily by spearphishing campaigns using tailored emails that contain malicious attachments or links, which—if enabled—execute malware (Phishing: Spearphishing Attachment [T1566.001], Phishing: Spearphishing Link [T1566.002]).

The phishing emails contain links that redirect to a website hosted on a compromised server that prompts the victim to click on photo proof of their traffic violation (User Execution: Malicious Link [T1204.001], User Execution: Malicious File [T1204.002]). In clicking the photo, the victim unknowingly downloads a malicious JavaScript file that, when opened, automatically communicates with the malicious actor’s command and control (C2) server to download TrickBot to the victim’s system

TrickBot is capable of data exfiltration over a hardcoded C2 server, cryptomining, and host enumeration (e.g., reconnaissance of Unified Extensible Firmware Interface or Basic Input/Output System [UEFI/BIOS] firmware) (Exfiltration Over C2 Channel [T1041], Resource Hijacking [T1496], System Information Discovery [T1082]).[2]

# Applying ATT&CK to TrickBot



**Initial Access**
- T1566: Phishing
  - T1566.001: Spearphishing Attachment
  - T1566.002: Spearphishing Link

**Execution**
- T1059: Command and Scripting Interpreter
  - T1059.003: Windows Command Shell
  - T1059.007: JavaScript/JScript
- T1106: Native API
- T1053: Scheduled Task/Job
  - T1053.005: Scheduled Task
- T1204: User Execution
  - T1204.001: Malicious Link
  - T1204.002: Malicious File

**Persistence**
- T1543: Create or Modify System Process
  - T1543.003: Windows Service
- T1053: Scheduled Task/Job
  - T1053.005: Scheduled Task

**Privilege Escalation**
- T1543: Create or Modify System Process
  - T1543.003: Windows Service
- T1055: Process Injection
  - T1055.012: Process Hollowing
- T1053: Scheduled Task/Job
  - T1053.005: Scheduled Task

**Defense Evasion**
- T1140: Deobfuscate/Decode Files or Information
- T1562: Impair Defenses
  - T1562.001: Disable or Modify Tools
- T1036: Masquerading
- T1112: Modify Registry
- T1027: Obfuscated Files or Information
  - T1027.002: Software Packing
- T1055: Process Injection
  - T1055.012: Process Hollowing
- T1553: Subvert Trust Controls
  - T1553.002: Code Signing

**Credential Access**
- T1555: Credentials from Password Stores
  - T1555.003: Credentials from Web Browsers
- T1056: Input Capture
  - T1056.004: Credential API Hooking
- T1552: Unsecured Credentials
  - T1552.001: Credentials In Files
  - T1552.002: Credentials in Registry

**Discovery**
- T1087: Account Discovery
  - T1087.001: Local Account
  - T1087.003: Email Account
- T1482: Domain Trust Discovery
- T1083: File and Directory Discovery
- T1069: Permission Groups Discovery
- T1018: Remote System Discovery
- T1082: System Information Discovery
- T1016: System Network Configuration Discovery
- T1033: System Owner/User Discovery
- T1007: System Service Discovery

**Lateral Movement**
- T1570: Lateral Tool Transfer

**Collection**
- T1005: Data from Local System
- T1056: Input Capture
  - T1056.004: Credential API Hooking
- T1185: Man in the Browser

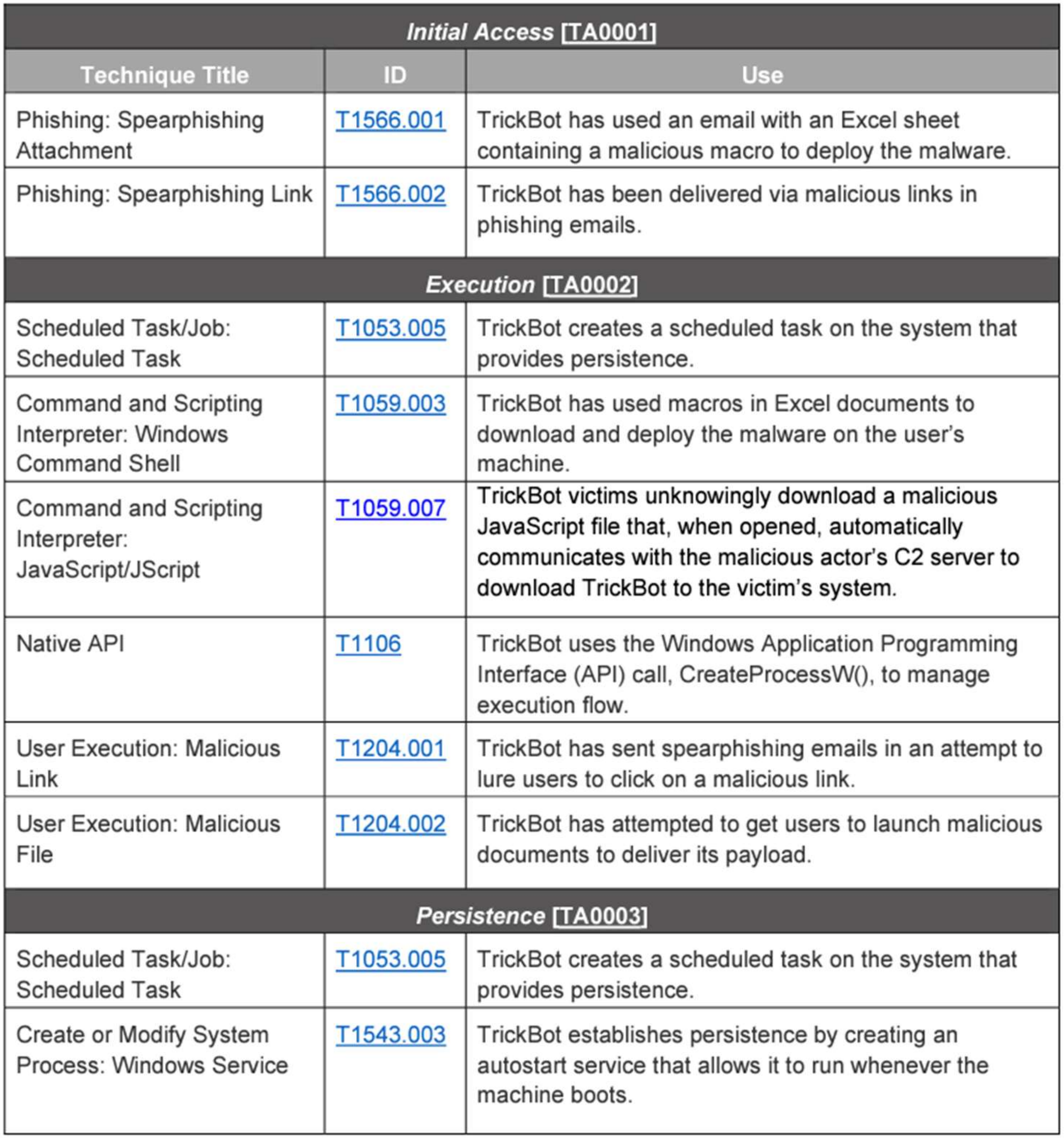
**Command and Control**
- T1071: Application Layer Protocol
  - T1071.001: Web Protocols
- T1132: Data Encoding
  - T1132.001: Standard Encoding
- T1573: Encrypted Channel
  - T1573.001: Symmetric Cryptography
- T1008: Fallback Channels
- T1105: Ingress Tool Transfer
- T1571: Non-Standard Port

**Exfiltration**
- T1041: Exfiltration Over C2 Channel

**Impact**
- T1496: Resource Hijacking

*Figure 1: ATT&CK Navigator visualization of enterprise techniques used by TrickBot*

# TrickBott ATT&CK Techniques



| Technique Title | ID | Use |
|---|---|---|
| ***Initial Access* [TA0001]** | | |
| Phishing: Spearphishing Attachment | T1566.001 | TrickBot has used an email with an Excel sheet containing a malicious macro to deploy the malware. |
| Phishing: Spearphishing Link | T1566.002 | TrickBot has been delivered via malicious links in phishing emails. |
| ***Execution* [TA0002]** | | |
| Scheduled Task/Job: Scheduled Task | T1053.005 | TrickBot creates a scheduled task on the system that provides persistence. |
| Command and Scripting Interpreter: Windows Command Shell | T1059.003 | TrickBot has used macros in Excel documents to download and deploy the malware on the user's machine. |
| Command and Scripting Interpreter: JavaScript/JScript | T1059.007 | TrickBot victims unknowingly download a malicious JavaScript file that, when opened, automatically communicates with the malicious actor's C2 server to download TrickBot to the victim's system. |
| Native API | T1106 | TrickBot uses the Windows Application Programming Interface (API) call, CreateProcessW(), to manage execution flow. |
| User Execution: Malicious Link | T1204.001 | TrickBot has sent spearphishing emails in an attempt to lure users to click on a malicious link. |
| User Execution: Malicious File | T1204.002 | TrickBot has attempted to get users to launch malicious documents to deliver its payload. |
| ***Persistence* [TA0003]** | | |
| Scheduled Task/Job: Scheduled Task | T1053.005 | TrickBot creates a scheduled task on the system that provides persistence. |
| Create or Modify System Process: Windows Service | T1543.003 | TrickBot establishes persistence by creating an autostart service that allows it to run whenever the machine boots. |