Cybersecurity Exam Study Notes

OWASP Risk Rating Methodology

The OWASP Risk Rating Methodology provides a framework for assessing security risks in web applications. The core formula is **Risk = Likelihood × Impact**. The methodology follows six steps: identifying a risk, estimating likelihood, estimating impact, determining severity, deciding what to fix, and customizing your risk rating model.

Likelihood Factors are divided into two categories. Threat Agent Factors include skill level (technical capability of attackers), motive (reward potential), opportunity (access/resources required), and size (how large the threat agent group is). Vulnerability Factors include ease of discovery, ease of exploit, awareness (how well-known the vulnerability is), and intrusion detection capability.

Impact Factors split into Technical Impact (loss of confidentiality, integrity, availability, and accountability) and Business Impact (financial damage, reputation damage, non-compliance, and privacy violation). Each factor is scored 0-9, then averaged within categories to produce likelihood and impact scores.

Severity Determination uses a matrix where scores 0-3 = Low, 3-6 = Medium, 6-9 = High. The final risk level combines likelihood and impact ratings. Business impact should take precedence over technical impact when business context is available. The methodology is meant to be customized—organizations should add factors relevant to their context, customize options, and weight factors based on what matters most to them.

CISSP Domains Overview

The Certified Information Systems Security Professional (CISSP) certification covers eight domains representing the Common Body of Knowledge (CBK) for information security professionals. Candidates need five years of experience in at least two domains.

Domain 1: Security and Risk Management (16%) is the largest domain, covering professional ethics, the CIA triad plus authenticity and non-repudiation, security governance aligned with business objectives, legal/regulatory compliance (GDPR, HIPAA, etc.), business continuity planning, personnel security, risk management concepts (threat identification, risk response, control types), threat modeling, supply chain risk management, and security awareness training.

Domain 2: Asset Security (10%) focuses on data classification, asset handling requirements, secure provisioning, data lifecycle management (collection through destruction), data retention policies, and data security controls including DRM, DLP, and CASB technologies.

Domain 3: Security Architecture and Engineering (13%) covers secure design principles (least privilege, defense in depth, zero trust, fail securely, separation of duties), security models (Bell-LaPadula, Biba), cryptographic solutions and attacks, physical security design, and system lifecycle management across various architectures including cloud, IoT, and containerized systems.

Domain 4: Communication and Network Security (13%) addresses network architecture (OSI/TCP-IP models), secure protocols (IPSec, SSH, TLS), network segmentation (VLANs, microsegmentation), wireless and cellular security, software-defined networking, and secure communication channels.

Domain 5: Identity and Access Management (13%) covers physical and logical access control, authentication strategies (MFA, passwordless, SSO), federated identity management, authorization models (RBAC, MAC, DAC, ABAC), and the identity provisioning lifecycle.

Domain 6: Security Assessment and Testing (12%) focuses on assessment strategies (internal, external, third-party), vulnerability assessment, penetration testing (red/blue/purple teams), code review, compliance checks, and security audit processes.

Domain 7: Security Operations (13%) encompasses investigations and digital forensics, logging and monitoring (SIEM, UEBA), configuration management, incident management lifecycle (detection through lessons learned), patch management, change management, disaster recovery, and business continuity.

Domain 8: Software Development Security (10%) covers secure SDLC integration (DevSecOps), security controls in development ecosystems (CI/CD, SAST/DAST), software security assessment, acquired software risks, and secure coding standards.

Botnet Fundamentals

Definitions and Core Concepts

A **bot** is a piece of computer code that performs tasks automatically. Bots are inherently neutral—they can play poker, search for prime numbers, or perform malicious activities. A **botnet** is a network of compromised machines that can be remotely controlled by an attacker. The **botmaster** is the person or group controlling the botnet.

Bots originated in Internet Relay Chat (IRC) networks (RFC 1459, 1993), initially performing benevolent tasks like channel administration and games. Early bots like Eggdrop were created to stop channel wars and primitive DoS attacks.

Four Key Botnet Characteristics

1. Networked: Bots communicate with command centers, sending reports, receiving orders, and getting updates. Researchers emphasize addressing 'not merely the numerous binaries but the network of attackers itself.'

2. Compromised Machines: Victims are unwilling participants. Compromises occur through vulnerability exploits, automated code exploits, web-based malware (phishing, drive-by downloads), or botnet takeovers.

3. Remote Control: Bots report to and receive orders from a Command and Control (C&C;) structure. This allows botmasters to leverage computing power of some or all bots as required.

4. Malicious Intent: Activities include DDoS attacks, spamming, network sniffing, keylogging, malware spreading, credit card theft, identity theft, and espionage. The 2007 Estonia DDoS attack demonstrated politically-motivated use.

Command and Control (C&C;) Architectures

IRC-based: Bots join channels for real-time instructions. Legacy but still exists (e.g., Hamweq botnet).

Web-based (HTTP/HTTPS): Bots connect to web servers via POST requests. Advantages: HTTP traffic blends with legitimate traffic making detection difficult; SSL encryption protects content and C&C; paths; can hide behind legitimate website facades. Used by Torpig and Rustock botnets.

Peer-to-Peer (P2P): Decentralized architecture where multiple bots share control roles. No single point of failure. Storm Worm used Kademlia DHT protocol.

Covert Channels: Communication through DNS requests or other stealth mechanisms.

Bot Tracking and Economics

Botmasters need accurate tracking to: assess computing power for DDoS attacks, account for diurnal patterns (bots active in their time zones), and monetize operations through bot rental. Botnet rental is common and inexpensive—hourly rates comparable to pub visits. The BBC demonstrated this by renting 22,000 compromised hosts.

Spam Economics

Despite extremely low response rates (below 0.00001%), spam remains profitable because: scale compensates for low conversion, targeted phishing dramatically increases response rates, it serves as an infection vector for botnet propagation, and it's versatile for multiple criminal purposes including identity theft and credit card fraud.

Bot Protection Techniques

Encryption: Evolved from simple decryptor routines to oligomorphic, polymorphic, and metamorphic schemes. RC4 was used by Rustock; SSL certificates provide simpler protection for web-based C&C.;

Obfuscation: Dead code insertion, subroutine reordering, instruction substitution create 'binary mutations' to evade AV products and slow researcher analysis.

Rootkits: Kernel-level hiding of processes from administrators. Redirects system calls to hide bot presence.

Domain Flux: Domain generation algorithms determine which C&C; to query in a given timeframe (used by Torpig).

Peer-to-Peer Botnets

Traditional botnets use centralized C&C; servers, creating a single point of failure. P2P botnets emerged to address this weakness by distributing control across the network.

P2P Botnet Architecture eliminates central servers by having all nodes function as both clients and servers. Communication uses content-based publish/subscribe mechanisms common in file-sharing systems like Gnutella, eMule, and BitTorrent. This provides high resilience against takedowns since there's no single point to attack.

Storm Worm (Trojan.Peacomm) was one of the most significant P2P botnets, emerging in 2007. It used OVERNET, a Kademlia-based distributed hash table (DHT) protocol. Later versions created their own network called 'Stormnet' using XOR encryption. The botnet employed a two-tier architecture: the first tier within the P2P network locates second-tier computers that send actual commands.

Kademlia DHT is the underlying protocol for many P2P botnets. It uses XOR distance metrics to organize peers and locate content. Each node has a 160-bit ID with O(log n) lookup complexity. The protocol's lack of authentication makes it vulnerable to exploitation by botnets.

Hybrid P2P Botnets combine centralized and decentralized elements. Some bots act as 'servents' (server+client) while others are simple clients. Ultrapeers handle routing while leaf nodes participate in attacks.

Mitigation Approaches: Infiltrating using honeypots, crawling P2P networks to enumerate infected machines, and pollution attacks injecting fake announcements. The unauthenticated nature of P2P protocols enables mitigation.

Botnet Mitigation Strategies

Technical Approaches

Patching: Many botnets propagate through worm-like behavior exploiting software vulnerabilities. Keeping systems well-patched is a prime mitigation strategy, though not sufficient alone.

Network Monitoring: Botnet traffic is anomalous. Creating network baselines helps detect suspicious hosts. Once identified, bots can be placed in network isolation ('walled gardens').

Combined HIDS + NIDS: Host and Network Intrusion Detection Systems provide complete visibility. Bots leave traces on compromised hosts. Open source options include OSSEC and Snort.

Security Suites: Combined firewall, malware detection, AV, spam and phishing protection minimize risk for individual users and small businesses.

Broader Ecosystem Perspective

Technical solutions alone are insufficient. 'As security comes at a cost, tolerating some level of insecurity is economically rational.' Key stakeholders must participate:

Software Vendors: Economic incentives favor speed over security; risk often gets dumped on consumers. Microsoft's 'Trustworthy Computing' initiative shows industry recognition of the problem.

ISPs: Increasingly taking ownership—reduced blacklisting risk, fewer support calls, brand protection. A small number of ISPs host a large number of 'troublesome' hosts; focusing on them yields significant progress.

Financial Industry: Proactive fraud management with PCI DSS. Reported losses around 0.06-1% show industries reaching equilibrium between security investment and acceptable loss levels.

International Cooperation: 'The impediment to fighting botnets is international law.' Botmasters often choose ISPs, registrars, and DNS providers in countries with weak enforcement.

Key Insight

Botnets cannot be fought in isolation. Every botnet will eventually be detected, but the hydra effect means new botnets continuously replace dismantled ones. Multi-disciplinary approaches (technical, social, policy) with broad ecosystem participation are essential. Securing cyberspace is recognized as one of the Grand Challenges for Engineering by the National Academy of Engineering.

Honeypots and Deception Technology

Honeypots are security mechanisms designed to detect, deflect, and counteract unauthorized access attempts. They appear as legitimate network resources but are actually isolated, monitored systems for capturing attacker behavior.

Classification by Interaction Level: Low-interaction honeypots emulate basic services (HTTP, FTP, Telnet) with limited functionality, detecting automated attacks with minimal risk. High-interaction honeypots provide full system environments capturing detailed tactics but requiring robust isolation. Pure honeypots are complete production-like systems monitored through bug taps on network links.

Classification by Purpose: Production honeypots divert attackers and provide early warning. Research honeypots gather intelligence about attack methods for academic or security research.

Honeynets are networks of multiple honeypots simulating interconnected environments. A 'honeywall' monitors and directs traffic. They track advanced threat actors and analyze lateral movement.

Honeytokens are fake data triggering alerts when accessed: canary files with embedded beacons, fake credentials in config files, bogus database records, fake AWS/API keys, DNS canaries, and planted email addresses.

Popular Tools: Cowrie (SSH/Telnet), Dionaea (SMB, HTTP, FTP, MSSQL), Conpot (ICS/SCADA), Glastopf (web applications), T-Pot (all-in-one platform with ELK integration).

Deployment: DMZ for external reconnaissance, internal networks for lateral movement, near high-value assets for targeted attacks, or as endpoint canaries for insider threats/ransomware. Always isolate from production.

Interdomain Routing and BGP Security

The Internet comprises approximately 60,000 autonomous systems (AS) interconnected by BGP. BGP is the 'glue' enabling routing between networks by exchanging reachability information.

BGP Vulnerabilities stem from trust-based design—no built-in mechanism to verify origin or path validity. BGP routers accept announcements based on mutual trust rather than cryptographic verification.

BGP Hijacking: Prefix hijacking announces someone else's prefix. Subprefix hijacking announces more specific routes (/24 instead of /23), preferred due to longest-prefix-match. Route leaks propagate information beyond intended scope.

RPKI provides cryptographic verification through Route Origin Authorizations (ROAs)—signed attestations specifying which AS can originate specific prefixes. Route Origin Validation (ROV) classifies routes as Valid, Invalid, or Not Found. As of 2024, ~53% of routes are covered by ROAs.

BGPsec extends RPKI to secure entire AS paths through cryptographic signatures. **ASPA** validates customer-provider relationships. Both have limited adoption due to complexity.

Best Practices: Deploy RPKI and create ROAs. Implement ROV. Use prefix filtering based on IRR data. Monitor BGP announcements. Implement maximum prefix limits. Comprehensive security requires broad participation.

Key Concepts Summary

Risk Management: Risk = Likelihood × Impact. Customize frameworks to organizational context. Balance technical and business impact assessments.

Defense in Depth: Layer multiple security controls. No single solution provides complete protection. Combine prevention, detection, and response capabilities.

Zero Trust: Never trust, always verify. Assume breach. Implement microsegmentation and continuous verification.

Resilience: Design systems to withstand attacks and failures. Eliminate single points of failure. Build redundancy and recovery capabilities.

Deception: Use honeypots and decoys to detect attackers early and gather intelligence. Integrate deception into broader security strategy.

Cryptographic Trust: PKI, RPKI, and certificate-based validation provide authentication and integrity. Adoption must be widespread to be effective.

Botnet Defense: Requires multi-stakeholder approach. Technical controls (patching, IDS, network monitoring) combined with ecosystem participation (ISPs, vendors, international cooperation). No single point solution exists.