

---

# **Introduction to Information Security (IIS) - Lecture Notes**

Matt Schlenker

## Contents

<b>1 The Security Mindset</b>	<b>25</b>
1.1 Why Cyber Security? . . . . .	25
1.2 Security Impact Quiz . . . . .	26
1.3 Security Impact Quiz Solution . . . . .	26
1.4 Cyber Assets at Risk . . . . .	26
1.5 Vulnerabilities and Attacks . . . . .	27
1.5.1 Trivial Example . . . . .	27
1.6 A Real World Example . . . . .	27
1.7 Black Market Prices Quiz . . . . .	28
1.8 Black Market Prices Quiz Solution . . . . .	29
1.9 Sony Pictures Quiz . . . . .	30
1.10 Sony Pictures Quiz Solution . . . . .	31
1.11 Revisiting Threats, Vulnerabilities and Risk . . . . .	31
1.12 What Should We Do in Cyber Security? . . . . .	32
1.13 Security Requirements Quiz . . . . .	33
1.14 Security Requirements Quiz Solution . . . . .	34
1.15 What Should the Good Guys Do? . . . . .	34
1.16 Losses Due to Cyber Crime Quiz . . . . .	35
1.17 Losses Due to Cyber Crime Quiz Solution . . . . .	36
1.18 How Do We Address Cyber Security? . . . . .	36
1.19 Security Mindset Quiz . . . . .	37
1.20 Security Mindset Quiz Solution . . . . .	38
<b>2 Software Security</b>	<b>38</b>
2.1 Software Vulnerabilities . . . . .	38
2.2 Vulnerable Program . . . . .	39
2.3 Stack Access Quiz . . . . .	40
2.4 Stack Access Quiz Solution . . . . .	41
2.5 Understanding the Stack . . . . .	41
2.6 Attacker Code Quiz . . . . .	42
2.7 Attacker Code Quiz Solution . . . . .	43
2.8 Attacker Code Execution Part 1 . . . . .	43
2.9 Attacker Code Execution Part 2 . . . . .	44
2.10 Attacker Code Execution Part 3 . . . . .	45
2.11 Buffer Overflow Quiz . . . . .	46
2.12 Buffer Overflow Quiz Solution . . . . .	47

2.13 Shellcode . . . . .	47
2.14 Shellcode Privileges . . . . .	48
2.15 NVD Quiz . . . . .	49
2.16 NVD Quiz Solution . . . . .	50
2.17 Return to libc . . . . .	50
2.18 Heap Overflows . . . . .	51
2.19 OpenSSL Heartbleed . . . . .	51
2.20 Defense Against Buffer Overflows . . . . .	51
2.21 Safe Languages . . . . .	52
2.22 Unsafe Languages . . . . .	52
2.23 Strongly Vs Weakly Typed Language Quiz . . . . .	53
2.24 Strongly Vs Weakly Typed Language Quiz Solution . . . . .	54
2.25 Analysis Tools . . . . .	54
2.26 Stack Canary . . . . .	54
2.27 ASLR . . . . .	55
2.27.1 Non-executable stack . . . . .	55
2.28 Buffer Overflow Attacks Quiz . . . . .	56
2.29 Buffer Overflow Attacks Quiz Solution . . . . .	57
<b>3 Operating System Security</b> . . . . .	<b>57</b>
3.1 Operating Systems Defined . . . . .	57
3.2 Operating Systems . . . . .	58
3.2.1 Abstraction of hardware . . . . .	58
3.2.2 Controlled hardware access . . . . .	59
3.2.3 Process Isolation . . . . .	59
3.3 Need for Trusting an Operating System . . . . .	60
3.3.1 Complete mediation . . . . .	60
3.3.2 Tamper-proof . . . . .	60
3.3.3 Correctness . . . . .	60
3.4 OS and Resource Protection . . . . .	60
3.5 Secure OS Quiz 1 . . . . .	61
3.6 Secure OS Quiz 1 Solution . . . . .	62
3.7 Secure OS Quiz 2 . . . . .	63
3.8 Secure OS Quiz 2 Solution . . . . .	64
3.9 Secure OS Quiz 3 . . . . .	65
3.10 Secure OS Quiz 3 Solution . . . . .	66
3.11 How Can We Trust an OS? . . . . .	66
3.12 System Calls From User to OS Code . . . . .	67

3.13 Untrusted User Code Isolation . . . . .	67
3.14 User Isolation Quiz . . . . .	68
3.15 User Isolation Quiz Solution . . . . .	69
3.16 Address Space . . . . .	69
3.17 Unit of Isolation . . . . .	70
3.18 Address Translation . . . . .	70
3.19 Code Protection . . . . .	72
3.20 Process Protection through Memory Management . . . . .	72
3.21 Revisiting Stack Overflow Quiz . . . . .	73
3.22 Revisiting Stack Overflow Quiz Solution . . . . .	74
3.23 Preventing Malicious Code Execution . . . . .	74
3.24 OS Isolation from Application Code Part 1 . . . . .	74
3.25 OS Isolation from Application Code Part 2 . . . . .	75
3.26 Kernel Memory Split . . . . .	76
3.27 Execution Privilege Level Quiz . . . . .	77
3.28 Execution Privilege Level Quiz Solution . . . . .	78
3.29 Complete Mediation: The TCB . . . . .	78
3.30 Complete Mediation: The User Code . . . . .	79
3.31 Complete Mediation: OS . . . . .	79
3.32 Virtualization . . . . .	79
3.33 Limiting the Damage of a Hacked OS . . . . .	80
3.34 Virtualization Security Layers . . . . .	81
3.35 Correctness: The Final TCB Requirement . . . . .	81
3.36 TCB Requirements Quiz . . . . .	82
3.37 TCB Requirements Quiz Solution . . . . .	83
3.38 Size of Security Code . . . . .	84
3.39 Size of Security Code Solution . . . . .	85
3.40 Hypervisor Code Size Quiz . . . . .	86
3.41 Hypervisor Code Size Quiz Solution . . . . .	87
<b>4 Authentication</b> . . . . .	<b>87</b>
4.1 What is Authentication? . . . . .	87
4.2 Authentication Goals . . . . .	89
4.3 Authentication Quiz . . . . .	89
4.4 Authentication Quiz Solution . . . . .	90
4.5 How is Authentication Implemented? . . . . .	90
4.6 Login Attacks Quiz . . . . .	92
4.7 Login Attacks Quiz Solution . . . . .	93

4.8 Implementation Quiz . . . . .	94
4.9 Implementation Quiz Solution . . . . .	95
4.10 Threat Modeling of the Password Method . . . . .	95
4.11 Importance of a Trusted Path . . . . .	96
4.12 Password Popularity Quiz . . . . .	96
4.13 Password Popularity Quiz Solution . . . . .	97
4.14 Implementing Password Authentication . . . . .	97
4.14.1 Method 1 . . . . .	97
4.14.2 Method 2 . . . . .	98
4.15 Hash Functions . . . . .	98
4.15.1 Hash Functions and Threats . . . . .	99
4.16 Password Quiz . . . . .	100
4.17 Password Quiz Solution . . . . .	101
4.18 Hashed Passwords Quiz . . . . .	102
4.19 Hashed Passwords Quiz Solution . . . . .	103
4.20 Hash Function Characteristics Quiz . . . . .	104
4.21 Hash Function Characteristics Quiz Solution . . . . .	105
4.22 Brute Force Guessing of Passwords . . . . .	105
4.23 Passwords are not Really Random . . . . .	106
4.24 Brute Force Figure . . . . .	107
4.25 Unique PINS Quiz . . . . .	108
4.26 Unique PINS Quiz Solution . . . . .	109
4.27 Brute Force Quiz . . . . .	110
4.28 Brute Force Quiz Solution . . . . .	111
4.29 Touch Screen Passwords Quiz . . . . .	112
4.30 Touch Screen Passwords Quiz Solution . . . . .	113
4.31 Problems with Passwords . . . . .	113
4.31.1 Best Practices for System Administrators . . . . .	114
4.31.2 Best Practices for Users . . . . .	114
4.32 Something You Have Authentication . . . . .	114
4.33 Something You Are Authentication . . . . .	115
4.33.1 Implementing Biometric Authentication . . . . .	116
4.34 Other Methods . . . . .	116
4.34.1 Multi-factor Authentication . . . . .	116
4.34.2 Authentication over a Network . . . . .	117
4.35 Multi-factor Authentication Quiz . . . . .	117
4.36 Multi-factor Authentication Quiz Solution . . . . .	118
4.37 Chip and Pin Authentication Quiz . . . . .	119

4.38 Chip and Pin Authentication Quiz Solution . . . . .	120
4.39 Biometric Authentication Quiz . . . . .	121
4.40 Biometric Authentication Quiz Solution . . . . .	122
<b>5 Access Control</b>	<b>122</b>
5.1 Controlling Accesses to Resources . . . . .	122
5.2 Access and Authentication . . . . .	123
5.3 Access Control Matrix Defined . . . . .	123
5.4 Access Control Matrix . . . . .	124
5.5 Data Confidentiality Quiz . . . . .	125
5.6 Data Confidentiality Quiz Solution . . . . .	126
5.7 Determining Access Quiz . . . . .	127
5.8 Determining Access Quiz Solution . . . . .	128
5.9 Discretionary Access Control Quiz . . . . .	129
5.10 Discretionary Access Control Quiz Solution . . . . .	130
5.11 Implementing Access Control . . . . .	130
5.12 Example Access Control Matrix . . . . .	131
5.13 ACL Implementation . . . . .	131
5.14 C Lists Implementation . . . . .	132
5.15 ACL and C Lists Implementation: ACL vs C list . . . . .	133
5.15.1 Efficiency . . . . .	133
5.15.2 Accountability . . . . .	133
5.15.3 Revocation . . . . .	133
5.16 ACE Quiz . . . . .	134
5.17 ACE Quiz Solution . . . . .	135
5.18 ACE Access Quiz . . . . .	136
5.19 ACE Access Quiz Solution . . . . .	137
5.20 Revocation of Rights Quiz . . . . .	138
5.21 Revocation of Rights Quiz Solution . . . . .	139
5.22 Access Control Implementation Part 1 . . . . .	139
5.23 Access Control Implementation SetUID . . . . .	140
5.24 Access Control Implementation Part 2 . . . . .	140
5.25 How does the OS Implement ACL? . . . . .	141
5.26 Time to Check vs Time to Use Quiz . . . . .	142
5.27 Time to Check vs Time to Use Quiz Solution . . . . .	143
5.28 Unix File Sharing Quiz . . . . .	144
5.29 Unix File Sharing Quiz Solution . . . . .	145
5.30 SetUID Bit Quiz . . . . .	146

5.31 SetUID Bit Quiz Solution . . . . .	147
5.32 Role Based Access Control . . . . .	147
5.32.1 RBAC Benefits . . . . .	148
5.32.2 Least Privilege . . . . .	148
5.33 RBAC Benefits Quiz . . . . .	149
5.34 RBAC Benefits Quiz Solution . . . . .	150
5.35 Access Control Policy Quiz . . . . .	151
5.36 Access Control Policy Quiz Solution . . . . .	152
<b>6 Mandatory Access Control</b>	<b>152</b>
6.1 Discretionary Access Control . . . . .	152
6.2 Two Problems with DAC . . . . .	152
6.3 DAC Quiz . . . . .	153
6.4 DAC Quiz Solution . . . . .	154
6.5 Mandatory Access Control (MAC) Models . . . . .	154
6.6 Implementing MAC . . . . .	155
6.7 Implementing MAC Example . . . . .	155
6.8 Health Data Quiz . . . . .	156
6.9 Health Data Quiz Solution . . . . .	157
6.10 Security Clearance Quiz . . . . .	158
6.11 Security Clearance Quiz Solution . . . . .	159
6.12 Comparing Labels . . . . .	159
6.13 Comparing Labels Example . . . . .	160
6.14 Ordering Among Labels . . . . .	160
6.15 Order Quiz . . . . .	161
6.16 Order Quiz Solution . . . . .	162
6.17 Label Domination Quiz . . . . .	163
6.18 Label Domination Quiz Solution . . . . .	164
6.19 Sensitive Data Quiz . . . . .	165
6.20 Sensitive Data Quiz Solution . . . . .	166
6.21 Using Labels for MAC: Confidentiality . . . . .	166
6.21.1 Read-Down Rule . . . . .	167
6.21.2 Write-Up Rule . . . . .	167
6.22 Preventing Information Flow with BLP . . . . .	167
6.23 Unclassified Documents Quiz . . . . .	169
6.24 Unclassified Documents Quiz Solution . . . . .	170
6.25 Classified Data Quiz . . . . .	171
6.26 Classified Data Quiz Solution . . . . .	172

6.27 BLP Model Quiz . . . . .	173
6.28 BLP Model Quiz Solution . . . . .	174
6.29 Other MAC Models . . . . .	174
6.30 Policies for Commercial Environments Part 1 . . . . .	175
6.31 Policies for Commercial Environments Part 2 . . . . .	175
6.31.1 Clark-Wilson Policy . . . . .	175
6.31.2 Chinese Wall Policy . . . . .	176
6.32 Clark Wilson Quiz . . . . .	177
6.33 Clark Wilson Quiz Solution . . . . .	178
6.34 COI Quiz . . . . .	179
6.35 COI Quiz Solution . . . . .	180
6.36 RBAC Quiz . . . . .	181
6.37 RBAC Quiz Solution . . . . .	182
6.38 MAC Support Quiz . . . . .	183
6.39 MAC Support Quiz Solution . . . . .	184
6.40 Revisiting TCB . . . . .	184
6.41 Trusting Software . . . . .	185
6.42 TCB Design Principles . . . . .	185
6.43 Least Privilege Quiz . . . . .	186
6.44 Least Privilege Quiz Solution . . . . .	187
6.45 TCB High Assurance Quiz . . . . .	188
6.46 TCB High Assurance Quiz Solution . . . . .	189
6.47 Design Principle Quiz . . . . .	190
6.48 Design Principle Quiz Solution . . . . .	191
6.49 Support Key Security Features . . . . .	191
6.50 Data Protection . . . . .	192
6.51 Trusted Paths . . . . .	192
6.51.1 Auditing . . . . .	193
6.52 Kernel Design . . . . .	193
6.53 Kernel Design and TCB . . . . .	193
6.54 Revisiting Assurance . . . . .	194
6.55 Assurance Challenges . . . . .	194
6.55.1 Functional Testing . . . . .	194
6.55.2 Penetration Testing . . . . .	195
6.55.3 Formal Verification . . . . .	195
6.56 Reducing TCB Size Quiz . . . . .	196
6.57 Reducing TCB Size Quiz Solution . . . . .	197
6.58 Testing TCB Quiz . . . . .	198

6.59 Testing TCB Quiz Solution . . . . .	199
6.60 Model Checking Quiz . . . . .	200
6.61 Model Checking Quiz Solution . . . . .	201
6.62 Security Evaluations: The Orange Book . . . . .	201
6.63 Security Evaluations: Common Criteria . . . . .	202
6.64 TCSEC Divisions Quiz . . . . .	202
6.65 TCSEC Divisions Quiz Solution . . . . .	203
6.66 Earning an EAL4 Certification Quiz . . . . .	204
6.67 Earning an EAL4 Certification Quiz Solution . . . . .	205
6.68 Cost Benefit Certification Tradeoffs Quiz . . . . .	206
6.69 Cost Benefit Certification Tradeoffs Quiz Solution . . . . .	207
<b>7 Database Security</b>	<b>207</b>
7.1 Importance of Database Security . . . . .	207
7.2 Database Threats Quiz . . . . .	208
7.3 Database Threats Quiz Solution . . . . .	209
7.4 Database Hacking Quiz . . . . .	210
7.5 Database Hacking Quiz Solution . . . . .	211
7.6 Relational Database Systems (RDBS) Part 1 . . . . .	211
7.7 Relational Database Systems (RDBS) Part 2 . . . . .	212
7.8 Relational Database Systems (RDBS) Part 3 . . . . .	213
7.9 Key Value Quiz . . . . .	214
7.10 Key Value Quiz Solution . . . . .	215
7.11 Database Views Quiz . . . . .	216
7.12 Database Views Quiz Solution . . . . .	217
7.13 Database Access Control . . . . .	217
7.13.1 Grant . . . . .	217
7.13.2 Revoke . . . . .	218
7.14 Database Access Control Quiz . . . . .	219
7.15 Database Access Control Quiz Solution . . . . .	220
7.16 Cascading Authorizations Quiz . . . . .	221
7.17 Cascading Authorizations Quiz Solution . . . . .	222
7.18 DAC or MAC Quiz . . . . .	223
7.19 DAC or MAC Quiz Solution . . . . .	224
7.20 Attacks on Databases: SQL Injections Part 1 . . . . .	224
7.21 Attacks on Databases: SQL Injections Part 2 . . . . .	225
7.22 SQL Injection Defenses . . . . .	225
7.23 SQL Login Quiz . . . . .	226

7.24 SQL Login Quiz Solution . . . . .	227
7.25 SQL Login Quiz 2 . . . . .	228
7.26 SQL Login Quiz 2 Solution . . . . .	229
7.27 Inference Attacks on Databases Part 1 . . . . .	229
7.28 Inference Attacks on Databases Part 2 . . . . .	229
7.29 Inference Attacks on Databases Part 3 . . . . .	230
7.30 Defenses Against Inference Attacks . . . . .	230
7.31 Defenses Against Inference Attacks Process . . . . .	231
7.32 SQL Inference Attack Quiz . . . . .	231
7.33 SQL Inference Attack Quiz Solution . . . . .	232
7.34 SQL Inference Attack Quiz 2 . . . . .	233
7.35 SQL Inference Attack Quiz 2 Solution . . . . .	234
<b>8 Malicious Code . . . . .</b>	<b>235</b>
8.1 What is Malware Quiz . . . . .	235
8.2 What is Malware Quiz Solution . . . . .	236
8.3 Types of Malware . . . . .	236
8.4 Trap Doors . . . . .	237
8.5 Logic Bombs . . . . .	237
8.6 Trojan Horses . . . . .	237
8.7 Viruses . . . . .	238
8.8 Host Required Malware Quiz 1 . . . . .	239
8.9 Host Required Malware Quiz 1 Solution . . . . .	240
8.10 Host Required Malware Quiz 2 . . . . .	241
8.11 Host Required Malware Quiz 2 Solution . . . . .	242
8.12 Virus Structure . . . . .	242
8.13 Types of Viruses . . . . .	244
8.14 Boot Sector Virus . . . . .	244
8.15 Macro Viruses . . . . .	245
8.16 Types of Viruses Quiz . . . . .	246
8.17 Types of Viruses Quiz Solution . . . . .	247
8.18 Rootkit . . . . .	247
8.19 Rootkit Quiz . . . . .	250
8.20 Rootkit Quiz Solution . . . . .	251
8.21 Truth and Misconceptions Quiz . . . . .	252
8.22 Truth and Misconceptions Quiz Solution . . . . .	253
8.23 Worms . . . . .	253

8.24	The Internet Worm . . . . .	253
8.24.1	Programming Error . . . . .	254
8.24.2	Implementation . . . . .	254
8.24.3	What We Learned . . . . .	255
8.25	Worm Quiz . . . . .	255
8.26	Worm Quiz Solution . . . . .	256
8.27	Malware Prevention & Detection Approaches . . . . .	256
8.27.1	Simple Scanners . . . . .	257
8.27.2	Heuristic Scanners . . . . .	257
8.27.3	Activity Traps . . . . .	257
8.27.4	Full-Featured Analysis . . . . .	257
8.28	Malware Prevention & Detection Quiz . . . . .	258
8.29	Malware Prevention & Detection Quiz Solution . . . . .	259
8.30	Most Expensive Worm Quiz . . . . .	260
8.31	Most Expensive Worm Quiz Solution . . . . .	261
<b>9</b>	<b>Modern Malware</b>	<b>261</b>
9.1	Past Malware . . . . .	261
9.2	Modern Malware . . . . .	261
9.3	Botnet . . . . .	262
9.4	Bot Quiz . . . . .	263
9.5	Bot Quiz Solution . . . . .	264
9.6	Attacks and Frauds by Botnets . . . . .	264
9.7	DDoS Using Botnets . . . . .	265
9.8	Amplification Distributed Reflective Attacks . . . . .	265
9.9	DDoS Quiz . . . . .	266
9.10	DDoS Quiz Solution . . . . .	267
9.11	Botnet Command and Control . . . . .	267
9.12	Botnet C&C Problem . . . . .	267
9.13	Botnet C&C: Naive Approach . . . . .	268
9.14	Botnet C&C Design . . . . .	268
9.15	C&C Design Quiz . . . . .	269
9.16	C&C Design Quiz Solution . . . . .	270
9.17	DNS Based Botnet C&C . . . . .	270
9.18	Botnet C&C Quiz . . . . .	272
9.19	Botnet C&C Quiz Solution . . . . .	273
9.20	Advanced Persistent Threat . . . . .	273
9.20.1	Advanced . . . . .	273

9.20.2 Persistent . . . . .	274
9.20.3 Threat . . . . .	274
9.21 APT Lifecycle . . . . .	274
9.22 APT Characteristics . . . . .	275
9.23 APT Quiz . . . . .	276
9.24 APT Quiz Solution . . . . .	277
9.25 APT Example . . . . .	277
9.26 Malware Analysis . . . . .	278
9.27 Malware Obfuscation . . . . .	278
9.28 Unpacking . . . . .	279
9.29 Malware Analysis Quiz . . . . .	280
9.30 Malware Analysis Quiz Solution . . . . .	281
<b>10 Firewalls</b>	<b>281</b>
10.1 Defense in Depth . . . . .	281
10.2 What is a Firewall? . . . . .	282
10.3 Firewalls Quiz . . . . .	283
10.4 Firewalls Quiz Solution . . . . .	284
10.5 Firewall Design Goals . . . . .	284
10.6 Firewall Access Policy . . . . .	284
10.7 Firewall Limitations . . . . .	285
10.8 Additional Convenient Firewall Features . . . . .	285
10.9 Firewall Features Quiz . . . . .	286
10.10 Firewall Features Quiz Solution . . . . .	287
10.11 Firewalls and Filtering . . . . .	287
10.12 Filtering Types . . . . .	287
10.13 Packet Filtering . . . . .	287
10.14 Packet Filtering Firewall . . . . .	288
10.15 Firewall Filtering Quiz . . . . .	289
10.16 Firewall Filtering Quiz Solution . . . . .	290
10.17 Typical Firewall Configuration . . . . .	290
10.18 Packet Filtering Examples . . . . .	291
10.19 Modifying the Rules on Source Ports . . . . .	292
10.20 Packet Filtering Advantages . . . . .	293
10.21 Packet Filtering Weaknesses . . . . .	293
10.22 Packet Filtering Firewall Countermeasures . . . . .	294
10.23 Packet Filtering Quiz . . . . .	295
10.24 Packet Filtering Quiz Solution . . . . .	296

10.25 Stateful Inspection Firewall . . . . .	296
10.26 Connection State Table . . . . .	296
10.27 Application Level Gateway . . . . .	297
10.28 Filtering Quiz . . . . .	299
10.29 Filtering Quiz Solution . . . . .	300
10.30 Bastion Hosts . . . . .	300
10.31 Host Based Firewalls . . . . .	301
10.32 Host Based Firewall Advantages . . . . .	301
10.33 Personal Firewalls . . . . .	301
10.34 Advanced Firewall Protection . . . . .	302
10.35 Personal Firewalls Quiz . . . . .	303
10.36 Personal Firewalls Quiz Solution . . . . .	304
10.37 Deploying Firewalls . . . . .	304
10.38 Internal Firewalls . . . . .	305
10.39 Distributed Firewall Deployment . . . . .	306
10.40 Firewall Deployment Quiz . . . . .	307
10.41 Firewall Deployment Quiz Solution . . . . .	308
10.42 Stand Alone Firewall Quiz . . . . .	309
10.43 Stand Alone Firewall Quiz Solution . . . . .	310
10.44 Firewall Topologies . . . . .	311

<b>11 Intrusion Detection</b>	<b>311</b>
11.1 Defense in Depth . . . . .	311
11.2 Intrusion Examples . . . . .	311
11.3 Intrusion Detection Quiz . . . . .	312
11.4 Intrusion Detection Quiz Solution . . . . .	313
11.5 Intrusion Detection System (IDS) . . . . .	313
11.6 Intruder Behavior . . . . .	314
11.7 Intruder Quiz . . . . .	316
11.8 Intruder Quiz Solution . . . . .	317
11.9 Types of Backdoors Quiz . . . . .	318
11.10 Types of Backdoors Quiz Solution . . . . .	319
11.11 Elements of Intrusion Detection . . . . .	319
11.12 Components of Intrusion Detection Systems . . . . .	320
11.13 Intrusion Detection Approaches . . . . .	320
11.14 Analysis Approaches . . . . .	321
11.15 Analysis Detection Quiz . . . . .	321
11.16 Analysis Detection Quiz Solution . . . . .	322

11.17 Signature Detection Quiz . . . . .	323
11.18 Signature Detection Quiz Solution . . . . .	324
11.19 A Variety of Classification Approaches . . . . .	324
11.20 Anomaly Quiz . . . . .	325
11.21 Anomaly Quiz Solution . . . . .	326
11.22 Statistical Approaches . . . . .	326
11.23 Knowledge Based Approach . . . . .	327
11.24 Statistical & Knowledge Based Approaches Quiz . . . . .	327
11.25 Statistical & Knowledge Based Approaches Quiz Solution . . . . .	328
11.26 Machine Learning Approaches . . . . .	328
11.27 Machine Learning Intruder Detection Approaches . . . . .	328
11.28 Machine Learning Quiz . . . . .	329
11.29 Machine Learning Quiz Solution . . . . .	330
11.30 Limitations of Anomaly Detection . . . . .	330
11.31 Anomaly Detection Example . . . . .	330
11.32 Misuse or Signature Detection . . . . .	331
11.33 Anomalous Behavior Quiz . . . . .	332
11.34 Anomalous Behavior Quiz Solution . . . . .	333
11.35 Signature Approaches . . . . .	333
11.36 Signature Approach Advantages & Disadvantages . . . . .	333
11.37 Zero Day Market Place Quiz . . . . .	334
11.38 Zero Day Market Place Quiz Solution . . . . .	335
11.39 Rule Based Detection . . . . .	335
11.40 Misuse Signature Intruder Detection . . . . .	335
11.41 Attacks Quiz . . . . .	337
11.42 Attacks Quiz Solution . . . . .	338
11.43 Monitoring Networks and Hosts . . . . .	338
11.44 Network IDS . . . . .	338
11.45 Network Based IDS (NIDS) . . . . .	339
11.46 Host IDS . . . . .	339
11.47 NIDS Quiz . . . . .	340
11.48 NIDS Quiz Solution . . . . .	341
11.49 Inline Sensors . . . . .	341
11.50 Passive Sensors . . . . .	341
11.51 Firewall Versus Network IDS . . . . .	342
11.52 IDS Quiz . . . . .	343
11.53 IDS Quiz Solution . . . . .	344
11.54 NIDS Sensor Deployment . . . . .	344

11.55NIDS Sensor Deployment Quiz . . . . .	348
11.56NIDS Sensor Deployment Quiz Solution . . . . .	349
11.57Snort . . . . .	349
11.58Snort Components . . . . .	349
11.59Snort Configuration . . . . .	350
11.60Snort Rules . . . . .	350
11.61Snort Rules Options . . . . .	351
11.62Snort Rule Actions . . . . .	351
11.63Snort Rule Example . . . . .	353
11.64Snort Quiz . . . . .	354
11.65Snort Quiz Solution . . . . .	355
11.66Honeypots . . . . .	355
11.67Honeypots Classification . . . . .	356
11.68Honeypot Deployment . . . . .	356
11.69Honeypot Quiz . . . . .	358
11.70Honeypot Quiz Solution . . . . .	359
11.71Evaluating IDS . . . . .	359
11.72Bayesian Detection Rate . . . . .	360
11.73Architecture of Network IDS . . . . .	362
11.74IDS Quiz . . . . .	363
11.75IDS Quiz Solution . . . . .	364
11.76Eluding Network IDS . . . . .	364
11.77Insertion Attack . . . . .	365
11.78Evasion Attack . . . . .	365
11.79DOS Attacks on Network IDS . . . . .	366
11.80Intrusion Prevention Systems (IPS) . . . . .	367
11.81IDS Attack Quiz . . . . .	367
11.82IDS Attack Quiz Solution . . . . .	368
<b>12 Introduction to Cryptography</b> . . . . .	<b>368</b>
12.1 Decryption . . . . .	368
12.2 Encryption Basics . . . . .	369
12.3 Attacks on Encryption . . . . .	370
12.3.1 Brute-force . . . . .	370
12.3.2 Cryptanalysis . . . . .	370
12.3.3 Implementation attacks . . . . .	370
12.3.4 Social-engineering attacks . . . . .	371
12.4 Encryption Attack Quiz . . . . .	371

12.5 Encryption Attack Quiz Solution . . . . .	372
12.6 Simple Ciphers Quiz . . . . .	373
12.7 Simple Ciphers Quiz Solution . . . . .	374
12.8 Simple Ciphers . . . . .	374
12.9 Letter Frequency of Ciphers . . . . .	375
12.10 Monoalphabetic Cipher Quiz . . . . .	376
12.11 Monoalphabetic Cipher Quiz Solution . . . . .	377
12.12 Vigenere Cipher . . . . .	377
12.13 Vigenere Cipher Quiz . . . . .	379
12.14 Vigenere Cipher Quiz Solution . . . . .	380
12.15 What Should be Kept Secret? . . . . .	380
12.16 Types of Cryptography . . . . .	380
12.17 Hash Functions . . . . .	381
12.17.1 Efficiency . . . . .	381
12.17.2 One-way function . . . . .	381
12.17.3 Weak collision resistance . . . . .	382
12.17.4 Strong collision-resistant . . . . .	382
12.18 Hash Functions for Passwords . . . . .	382
12.19 Hash Function Quiz . . . . .	384
12.20 Hash Function Quiz Solution . . . . .	385
12.21 Symmetric Encryption . . . . .	385
12.22 Comparison of Encryption Algorithms . . . . .	386
12.23 Symmetric Encryption Quiz . . . . .	389
12.24 Symmetric Encryption Quiz Solution . . . . .	390
12.25 Asymmetric Encryption . . . . .	390
12.26 Asymmetric Encryption Quiz . . . . .	391
12.27 Asymmetric Encryption Quiz Solution . . . . .	392
12.28 Digital Signatures . . . . .	392
12.29 Digital Envelopes . . . . .	393
12.30 Encryption Quiz . . . . .	395
12.31 Encryption Quiz Solution . . . . .	396
<b>13 Symmetric Encryption</b> . . . . .	<b>396</b>
13.1 Block Cipher Scheme . . . . .	396
13.2 Block Cipher Primitives . . . . .	397
13.3 Block Cipher Quiz . . . . .	398
13.4 Block Cipher Quiz Solution . . . . .	399
13.5 Data Encryption Standard . . . . .	399

13.6 Decryption . . . . .	402
13.7 XOR Quiz . . . . .	403
13.8 XOR Quiz Solution . . . . .	404
13.9 Mangler Function . . . . .	404
13.10S Box . . . . .	405
13.11S Box Quiz . . . . .	406
13.12S Box Quiz Solution . . . . .	407
13.13Security of DES . . . . .	407
13.14Triple DES . . . . .	407
13.15DES Quiz . . . . .	409
13.16DES Quiz Solution . . . . .	410
13.17Advanced Encryption Standard . . . . .	410
13.18AES Round . . . . .	411
13.19AES Encryption Quiz . . . . .	413
13.20AES Encryption Quiz Solution . . . . .	414
13.21Encrypting a Large Message . . . . .	414
13.22ECB Problem #1 . . . . .	415
13.23ECB Problem #2 . . . . .	416
13.24Cipher Block Chaining . . . . .	416
13.25Protecting Message Integrity . . . . .	417
13.26Protecting Message Confidentiality and Integrity . . . . .	418
13.27CBC Quiz . . . . .	418
13.28CBC Quiz Solution . . . . .	419
<b>14 Public-Key Cryptography</b> . . . . .	<b>419</b>
14.1 Modular Arithmetic . . . . .	419
14.2 Additive Inverse Quiz . . . . .	420
14.3 Additive Inverse Quiz Solution . . . . .	421
14.4 Modular Multiplication . . . . .	421
14.5 Modular Multiplication Quiz . . . . .	422
14.6 Modular Multiplication Quiz Solution . . . . .	423
14.7 Totient Function . . . . .	423
14.8 Totient Quiz . . . . .	424
14.9 Totient Quiz Solution . . . . .	425
14.10Modular Exponentiation . . . . .	425
14.11Modular Exponentiation Quiz . . . . .	426
14.12Modular Exponentiation Quiz Solution . . . . .	427
14.13RSA (Rivest, Shamir, Adleman) . . . . .	427

14.14 Why Does RSA Work . . . . .	429
14.15 RSA Quiz . . . . .	431
14.16 RSA Quiz Solution . . . . .	432
14.17 RSA Encryption Quiz . . . . .	433
14.18 RSA Encryption Quiz Solution . . . . .	434
14.19 Why is RSA Secure? . . . . .	434
14.20 Issues with Schoolbook RSA . . . . .	435
14.21 RSA in Practice Quiz . . . . .	435
14.22 RSA in Practice Quiz Solution . . . . .	436
14.23 Diffie and Hellman Key Exchange . . . . .	436
14.24 Diffie-Hellman Example . . . . .	437
14.25 Diffie-Hellman Quiz . . . . .	439
14.26 Diffie-Hellman Quiz Solution . . . . .	440
14.27 Diffie-Hellman Security . . . . .	440
14.28 Diffie-Hellman Limitations . . . . .	441
14.29 Bucket Brigade Attack, Man in the Middle (MIM) . . . . .	441
14.30 Other Public Key Algorithms . . . . .	442
14.31 RSA, Diffie-Hellman Quiz . . . . .	444
14.32 RSA, Diffie-Hellman Quiz Solution . . . . .	445

## **15 Hashes** 445

15.1 Hash Functions . . . . .	445
15.1.1 One-Way Function . . . . .	445
15.1.2 Weak Collision-Resistant Property . . . . .	446
15.1.3 Strong Collision-Resistant Property . . . . .	447
15.2 Hash Function Weaknesses . . . . .	448
15.2.1 Pigeonhole Principle . . . . .	448
15.2.2 Birthday Paradox . . . . .	448
15.2.3 Back to Hash Functions . . . . .	449
15.3 Determining Hash Length . . . . .	450
15.4 Hash Size Quiz . . . . .	450
15.5 Hash Size Quiz Solution . . . . .	451
15.6 Secure Hash Algorithm . . . . .	451
15.7 Message Processing . . . . .	452
15.8 Hash Based Message Authentication . . . . .	454
15.9 HMAC Security . . . . .	455
15.10 Hash Function Quiz . . . . .	456
15.11 Hash Function Quiz Solution . . . . .	457

<b>16 Security Protocols</b>	<b>457</b>
16.1 Why Security Protocols . . . . .	457
16.2 Mutual Authentication: Shared Secret . . . . .	458
16.3 Mutual Authentication Quiz . . . . .	460
16.4 Mutual Authentication Quiz Solution . . . . .	461
16.5 Mutual Authentication: Simplified . . . . .	461
16.5.1 Preventing Reflection Attacks . . . . .	463
16.6 Mutual Authentication Public Keys . . . . .	464
16.7 Security Protocols Quiz . . . . .	465
16.8 Security Protocols Quiz Solution . . . . .	466
16.9 Session Keys . . . . .	466
16.10 Key Distribution Center (KDC) . . . . .	467
16.11 Exchanging Public Key Certificates . . . . .	469
16.12 Session Key Quiz . . . . .	470
16.13 Session Key Quiz Solution . . . . .	471
16.14 Kerberos . . . . .	471
16.15 Accessing the Printer . . . . .	472
16.16 Kerberos Quiz . . . . .	475
16.17 Kerberos Quiz Solution . . . . .	476
<b>17 IPSec and TLS</b>	<b>476</b>
17.1 Goals of IPSec . . . . .	476
17.2 Spoofing Quiz . . . . .	477
17.3 Spoofing Quiz Solution . . . . .	478
17.4 IPSec Modes . . . . .	478
17.5 Tunnel Mode . . . . .	479
17.6 IPSec Quiz . . . . .	482
17.7 IPSec Quiz Solution . . . . .	483
17.8 IPSec Architecture . . . . .	483
17.9 Encapsulated Security Payload (ESP) . . . . .	484
17.10 ESP Modes Quiz . . . . .	485
17.11 ESP Modes Quiz Solution . . . . .	486
17.12 ESP in Transport Mode . . . . .	486
17.13 ESP Tunnel Mode . . . . .	487
17.14 Authentication Header . . . . .	488
17.15 ESP and AH Quiz . . . . .	490
17.16 ESP and AH Quiz Solution . . . . .	491
17.17 Internet Key Exchange . . . . .	491

17.18 Security Association . . . . .	491
17.19 SPD and SADB Fit Together . . . . .	492
17.20 SPD and SADB Example . . . . .	493
17.20.1 Transport Mode . . . . .	493
17.20.2 Tunnel Mode . . . . .	494
17.21 Outbound Processing . . . . .	494
17.22 Inbound Processing . . . . .	495
17.23 Anti Replay . . . . .	496
17.24 IPSec Quiz . . . . .	497
17.25 IPSec Quiz Solution . . . . .	498
17.26 Internet Key Exchange . . . . .	498
17.27 IKE Phase I . . . . .	498
17.28 IKE Phase I Example . . . . .	499
17.29 Diffie Hellman Quiz . . . . .	500
17.30 Diffie Hellman Quiz Solution . . . . .	501
17.31 Diffie-Hellman and Pre-shared Secret . . . . .	501
17.32 Authentication of the Key Exchange . . . . .	502
17.33 IKE Phase II Keys . . . . .	503
17.34 IPSec Summary . . . . .	503
17.35 IKE Quiz . . . . .	505
17.36 IKE Quiz Solution . . . . .	506
17.37 SSL and TLS . . . . .	506
17.38 TLS Concepts . . . . .	507
17.39 SSL Record Protocol . . . . .	507
17.40 The Handshake Protocol . . . . .	508
17.40.1 Phase One . . . . .	509
17.40.2 Phase Two . . . . .	510
17.40.3 Phase Three . . . . .	511
17.40.4 Phase Four . . . . .	512
17.41 TLS and SSL Quiz . . . . .	514
17.42 TLS and SSL Quiz Solution . . . . .	515
<b>18 Wireless and Mobile Security</b> . . . . .	<b>515</b>
18.1 Introduction to Wifi . . . . .	515
18.2 Wifi Quiz . . . . .	516
18.3 Wifi Quiz Solution . . . . .	517
18.4 Overview of Wifi Security . . . . .	517
18.5 Overview of 802.11i . . . . .	517

18.6 Wifi Security Standards Quiz . . . . .	518
18.7 Wifi Security Standards Quiz Solution . . . . .	519
18.8 Overview of Smartphone Security . . . . .	519
18.9 Overview of iOS Security . . . . .	520
18.10 Operating System Vulnerabilities Quiz . . . . .	522
18.11 Operating System Vulnerabilities Quiz Solution . . . . .	523
18.12 Hardware Security Feature . . . . .	523
18.13 iOS Trusted Bootchain . . . . .	524
18.14 File Data Encryption . . . . .	524
18.15 Security Quiz . . . . .	526
18.16 Security Quiz Solution . . . . .	527
18.17 Mandatory Code Signing . . . . .	527
18.18 Restricted App Distribution Model . . . . .	528
18.19 App Store Security Quiz . . . . .	528
18.20 App Store Security Quiz Solution . . . . .	529
18.21 Sandboxing . . . . .	529
18.22 Address Space Layout Randomization . . . . .	530
18.23 iOS Security Quiz . . . . .	531
18.24 iOS Security Quiz Solution . . . . .	532
18.25 Data Execution Prevention . . . . .	532
18.26 Passcodes and Touch ID . . . . .	533
18.27 iOS Quiz . . . . .	533
18.28 iOS Quiz Solution . . . . .	534
18.29 Android Security Overview . . . . .	534
18.30 Application Sandbox . . . . .	536
18.31 Android Sandbox vs iOS Sandbox . . . . .	536
18.32 Code Signing . . . . .	537
18.33 Android Apps Quiz . . . . .	538
18.34 Android Apps Quiz Solution . . . . .	539
<b>19 Web Security</b> . . . . .	<b>539</b>
19.1 How the Web Works . . . . .	539
19.2 Cookies . . . . .	540
19.3 Cookie Quiz . . . . .	542
19.4 Cookie Quiz Solution . . . . .	543
19.5 The Web and Security . . . . .	543
19.6 Web Security Quiz . . . . .	544
19.7 Web Security Quiz Solution . . . . .	545

19.8 Cross-Site Scripting (XSS) . . . . .	545
19.9 XSS Example . . . . .	545
19.10 XSS Quiz . . . . .	547
19.11 XSS Quiz Solution . . . . .	548
19.12 XSRF: Cross-Site Request Forgery . . . . .	548
19.13 XSRF Example . . . . .	549
19.14 XSRF vs XSS . . . . .	550
19.15 XSRF Quiz . . . . .	551
19.16 XSRF Quiz Solution . . . . .	552
19.17 Structured Query Language (SQL) . . . . .	552
19.18 Sample PHP Code . . . . .	552
19.19 Example Login . . . . .	553
19.20 Malicious User Input . . . . .	555
19.21 SQL Injection Quiz . . . . .	557
19.22 SQL Injection Quiz Solution . . . . .	558
<b>20 Cyber Security</b>	<b>558</b>
20.1 Managing Security . . . . .	558
20.2 Key Challenges . . . . .	559
20.3 Network Use Policy Quiz . . . . .	560
20.4 Network Use Policy Quiz Solution . . . . .	561
20.5 Botnet Quiz . . . . .	562
20.6 Botnet Quiz Solution . . . . .	563
20.7 Security Planning . . . . .	563
20.8 Assets and Threats . . . . .	563
20.9 Security Audit Quiz . . . . .	564
20.10 Security Audit Quiz Solution . . . . .	565
20.11 CISQ Quiz . . . . .	566
20.12 CISQ Quiz Solution . . . . .	567
20.13 Security Planning: Controls . . . . .	567
20.14 Security Planning: Security Policy . . . . .	568
20.15 Georgia Tech Computer and Network Use Policy . . . . .	569
20.16 Computer Use Policy Quiz . . . . .	570
20.17 Computer Use Policy Quiz Solution . . . . .	571
20.18 Student Privacy Quiz . . . . .	572
20.19 Student Privacy Quiz Solution . . . . .	573
20.20 Anthem Breach Quiz . . . . .	574
20.21 Anthem Breach Quiz Solution . . . . .	575

20.22 Cyber Risk Assessment . . . . .	575
20.23 Quantifying Cyber Risk . . . . .	576
20.24 Managing Cyber Risk . . . . .	576
20.25 Security Breach Quiz . . . . .	577
20.26 Security Breach Quiz Solution . . . . .	578
20.27 Reducing Exposure Quiz . . . . .	579
20.28 Reducing Exposure Quiz Solution . . . . .	580
20.29 Cyber Insurance Quiz . . . . .	581
20.30 Cyber Insurance Quiz Solution . . . . .	582
20.31 Enterprise Cyber Security Posture Part 1 . . . . .	582
20.31.1 Reactive . . . . .	582
20.31.2 Proactive . . . . .	583
20.32 Enterprise Cyber Security Posture Part 2 . . . . .	583
20.33 Security Planning and Management . . . . .	584
20.34 Cyber Security Budgets Quiz . . . . .	585
20.35 Cyber Security Budgets Quiz Solution . . . . .	586
20.36 Proactive Security Quiz . . . . .	587
20.37 Proactive Security Quiz Solution . . . . .	588
<b>21 Law, Ethics, and Privacy . . . . .</b>	<b>588</b>
21.1 US Laws Related to Online Abuse . . . . .	588
21.2 Legal Deterrents Quiz . . . . .	589
21.3 Legal Deterrents Quiz Solution . . . . .	590
21.4 Cost of Cybercrime Quiz . . . . .	591
21.5 Cost of Cybercrime Quiz Solution . . . . .	592
21.6 US Computer Fraud and Abuse Act (CFAA) . . . . .	592
21.7 Digital Millennium Copyright Act Part 1 . . . . .	593
21.8 Digital Millennium Copyright Act Part 2 . . . . .	593
21.9 Computer Abuse Laws Enforcement . . . . .	593
21.10 Melissa Virus Quiz . . . . .	594
21.11 Melissa Virus Quiz Solution . . . . .	595
21.12 Unauthorized Access Quiz . . . . .	596
21.13 Unauthorized Access Quiz Solution . . . . .	597
21.14 DMCA Exclusions Quiz . . . . .	598
21.15 DMCA Exclusions Quiz Solution . . . . .	599
21.16 Ethical Issues . . . . .	599
21.17 Computer Ethics Quiz . . . . .	600
21.18 Computer Ethics Quiz Solution . . . . .	601

21.19 Responsible Disclosure Quiz . . . . .	602
21.20 Responsible Disclosure Quiz Solution . . . . .	603
21.21 Privacy Definition . . . . .	603
21.22 Privacy is Not a New Problem . . . . .	603
21.23 What is Private . . . . .	604
21.24 What is Not Private . . . . .	605
21.25 Institutional Privacy . . . . .	605
21.26 Privacy Quiz . . . . .	606
21.27 Privacy Quiz Solution . . . . .	607
21.28 Right to Be Forgotten Quiz . . . . .	608
21.29 Right to Be Forgotten Quiz Solution . . . . .	609
21.30 Threats to Privacy Part 1 . . . . .	609
21.31 Threats to Privacy Part 2 . . . . .	610
21.32 Privacy Threats to Online Tracking Info . . . . .	610
21.33 Example: Google Privacy Policy . . . . .	611
21.34 Google: How is Info Used . . . . .	611
21.35 Google: Who is it Shared With . . . . .	612
21.36 Google: Information Security . . . . .	612
21.37 EFF Quiz . . . . .	613
21.38 EFF Quiz Solution . . . . .	614
21.39 Google Privacy Policy Quiz . . . . .	615
21.40 Google Privacy Policy Quiz Solution . . . . .	616
21.41 Legal Deterrents Quiz . . . . .	617
21.42 Legal Deterrents Quiz Solution . . . . .	618
21.43 Facebook Privacy Policies . . . . .	618
21.44 Privacy Enhancing Technologies . . . . .	619
21.45 TOR . . . . .	620
21.46 Controlling Tracking on the Internet . . . . .	621
21.47 Fandango Quiz . . . . .	621
21.48 Fandango Quiz Solution . . . . .	622
21.49 Tracking Quiz . . . . .	623
21.50 Tracking Quiz Solution . . . . .	624

## 1 The Security Mindset

### 1.1 Why Cyber Security?

Before considering why we care about cyber security, it's important to understand why we would care about security in general.

Essentially, we worry about security when we have something of value which is at risk of being harmed.

In order to understand the importance of cyber security, we need to consider the digital assets that we maintain and the threats those assets face.

For example, individuals store a lot of sensitive data online, such as financial information and medical information. If criminals get their hands on this data, they can monetize it and profit from it.

More broadly, communities rely on the internet for critical resources, making them attractive targets for adversaries.

Smart grids, which control the electrical power generation and distribution, are one such community asset. If computers control the smart grid, then whoever controls those computer controls an extremely important resource.

At a more general level, businesses and governments use the internet to carry out their daily activities, many of which involve storing proprietary and/or classified information. If hackers or adversaries gain access to this information, the consequences could be economically or politically disastrous.

## 1.2 Security Impact Quiz



### Security Impact Quiz

Each of these organizations suffered data breaches of more than 30,000 records. Check the companies that you have patronized:

<input type="checkbox"/> Home Depot	<input type="checkbox"/> Anthem
<input type="checkbox"/> Facebook	<input type="checkbox"/> Target
<input type="checkbox"/> Ebay	<input type="checkbox"/> Twitter
<input type="checkbox"/> Apple	<input type="checkbox"/> UPS
<input type="checkbox"/> JP Morgan Chase	<input type="checkbox"/> Mozilla
<input type="checkbox"/> Snapchat	<input type="checkbox"/> Nintendo



## 1.3 Security Impact Quiz Solution

No one right solution. I think the important thing to understand here is that no company is safe from a breach, and many companies that we interact with on a daily basis have suffered security breaches.

## 1.4 Cyber Assets at Risk

In order to define the security mindset, we need to consider:

- Threats (who are the bad actors?)
- Vulnerabilities (what weaknesses can they exploit?)
- Attack (how will the bad actors exploit the weaknesses?)

A **threat source** refers to an individual or entity that wishes to do us harm in our online lives.

**Cybercriminals** want to profit from our sensitive data for financial gain.

**Hacktivists** mount attacks in accordance with some political or philosophical agenda (think Edward Snowden).

**Nation states** thwart cyber security in order to gain political advantage through espionage.

## 1.5 Vulnerabilities and Attacks

Threat actors exploit vulnerabilities to launch attacks.

An example of a vulnerability is a weak password. A threat actor can likely guess a weak password, and use that password to *compromise* your account.

If the threat actor is able to compromise an entire digital system instead of just a single account - by gaining access to a centralized server or database, for example - we refer to this compromise as a *security breach*.

An **attack** is a successful exploitation of a vulnerability by a threat source, resulting in a system that has been compromised.

Unfortunately, vulnerabilities are very hard to get rid of completely. They are found in software, networks and, frequently, humans.

### 1.5.1 Trivial Example

Many of us ride our bikes.

Our bikes are an important asset to us: they cost money and they help us get around. In addition, there are threats against bikes. People steal bikes.

If you just leave a bike lying around, it is seriously vulnerable. Someone can just pick it up and walk away.

One way we can chose to protect our bike is by locking it - by the wheel - to a sturdy or heavy object.

In this case, the thief is not going to fight the security you have in place (the lock), but rather will unscrew the lock from the frame and walk away with the bike minus the wheel.

The vulnerability we have here is that while we have secured the wheel, we have left the rest of the bike (which is still valuable) unsecured.

We never realized that we had to protect more than just the wheel.

## 1.6 A Real World Example

The [Target data breach](#) of 2013 is one of the most widely publicized security breaches.

When thinking about security breaches there are different questions to ask:

- What is of value?
- What is the threat source?

- What vulnerability was exploited?

In this attack, the information of value was credit card data that was available on the point-of-sale systems present in Target stores.

The threat source was cybercriminals wishing to profit off of this information.

In this case, the vulnerability was human.

The attack began with a [phishing](#) message sent to a member of the HVAC company that was contracting for Target. Through the phishing attacks, the criminals were able to get credentials to get onto Target's network, at which point they were able to install malware on the point-of-sale systems in order to siphon off the credit card numbers.

## 1.7 Black Market Prices Quiz

### Black Market Prices Quiz



What is your hacked/stolen data worth on the Black Market (as of March 2015)? Enter dollar amounts in the boxes next to the data.

3 digit security code on your credit card  
 Credit card information  
 PayPal/Ebay account  
 Health information



## 1.8 Black Market Prices Quiz Solution



### Black Market Prices Quiz

What is your hacked/stolen data worth on the Black Market (as of March 2015)? Enter dollar amounts in the boxes next to the data.

\$2	3 digit security code on your credit card
\$5-\$45	Credit card information
\$27	PayPal/Ebay account
\$10	Health information

I think the point here is not to remember the exact numbers, but rather to understand that this information can be purchased relatively cheaply. This makes sense given that millions of records can be retrieved in a single breach.

### 1.9 Sony Pictures Quiz



#### Sony Pictures Quiz

With regards to the *The Interview* (2014) related hack, answer the following questions. Put the number of the correct answer in the box next to the question:

- The threat source was:   
[1] cybercriminals, [2] Hacktivists, or [3] Nation-State
- Goal of the attack was:   
[1] Monetize stolen information, [2] Stop Sony from releasing the movie, [3] Extort money from Sony
- What did the attack accomplish:   
[1] Disclosed sensitive data, [2] Destroyed Sony computers



## 1.10 Sony Pictures Quiz Solution



### Sony Pictures Quiz

With regards to the [The Interview \(2014\)](#) related hack, answer the following questions. Put the number of the correct answer in the box next to the question:

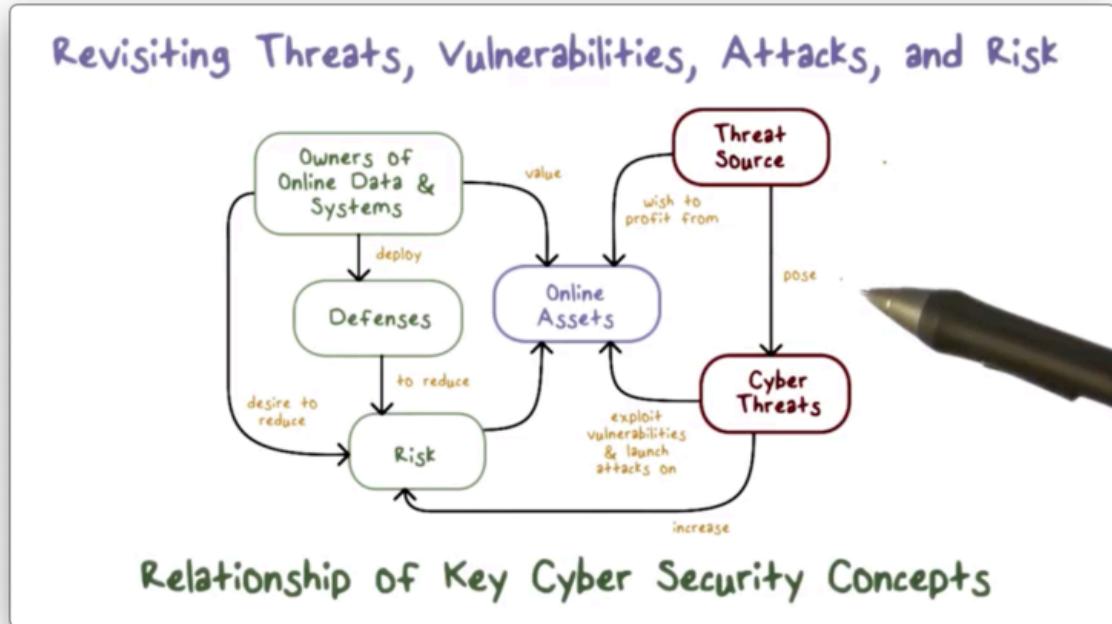
- The threat source was:  [1] cybercriminals, [2] Hacktivists, or [3] Nation-State
- Goal of the attack was:  [1] Monetize stolen information, [2] Stop Sony from releasing the movie, [3] Extort money from Sony
- What did the attack accomplish:  [1] Disclosed sensitive data, [2] Destroyed Sony computers



Read more [here](#)

## 1.11 Revisiting Threats, Vulnerabilities and Risk

The following diagram succinctly illustrates the relationship between assets, vulnerabilities, threats, and risk.



## 1.12 What Should We Do in Cyber Security?

There are different things that we can hope to achieve in the field of cyber security.

First, we can try to make threats go away. This is not an easy feat to achieve, but we can try to discourage criminal activity by introducing computer abuse laws.

We can reduce vulnerabilities, but we are never going to have zero vulnerabilities. Complex systems are error-prone, and some of those errors will expose vulnerabilities that can be exploited.

Alternatively, we can strive to meet various security requirements of sensitive information. The most common requirements are confidentiality, integrity, and availability.

If the data is sensitive in the sense that it cannot be disclosed to unauthorized parties, then a **confidentiality** requirement is present.

If no one should be able to modify or corrupt the data, the data is said to have an **integrity** requirement.

If the data is critical in the sense that we must always have access to it - your bank account data, for instance - then the data has an **availability** requirement.

These three requirements comprise the **CIA triad**.

While these three requirements are primarily concerned with the data aspects of cyber security, cyber attacks can also have physical consequences. The most well-known case of this is [Stuxnet](#).

### 1.13 Security Requirements Quiz



#### Security Requirements Quiz

Data breaches violate which of the following security requirements?

- Integrity
- Availability
- Confidentiality



### 1.14 Security Requirements Quiz Solution



## Security Requirements Quiz

Data breaches violate which of the following security requirements?

- Integrity
- Availability
- Confidentiality



Since data breaches involve the disclosure of information to unauthorized parties, these breaches violate confidentiality.

### 1.15 What Should the Good Guys Do?

The good guys have many different jobs.

We have to worry about *prevention*, which means keeping the bad guys out of our systems.

Since prevention is not guaranteed, we want to *detect* the inevitable compromise as quickly as possible.

After detection, we must make sure that we *respond* to the compromise, which may include *recovery* of lost or corrupted data.

Finally, we must provide some *remediation* to ensure that the same attack doesn't happen again.

Cybersecurity consists of *policies*, which describe what needs to be done on each of these fronts, and *mechanisms*, which describe how the policies will be carried out.

### 1.16 Losses Due to Cyber Crime Quiz



#### Cybercrime Losses Quiz

What is the estimated value of world-wide losses due to cybercrime?

- Less than \$10 Billion (US)
- Close to \$500 Billion (US)
- Trillions of US Dollars



### 1.17 Losses Due to Cyber Crime Quiz Solution

 **Cybercrime Losses Quiz**

What is the estimated value of world-wide losses due to cybercrime?

Less than \$10 Billion (US)

Close to \$500 Billion (US)

Trillions of US Dollars



### 1.18 How Do We Address Cyber Security?

One way to reduce vulnerability is to follow design principles that are good for security.

**Economy of mechanism** means the design of security measures built into the system should be as simple and small as possible.

**Fail-safe default** means that access decisions should be explicitly granted rather than explicitly denied. The default situation is lack of access, and the security controls identify conditions under which access is granted.

**Complete mediation** says that every access to a resource must be checked against the access controls. No access should proceed unmonitored.

**Open design** means the design of a security mechanism - for example, encryption algorithms - should be open rather than secret. Security by obscurity is a false promise.

**Least privilege** means that every process or user of the system should operate with the least set of privileges required to perform a task. This ensures that damage inflicted by a corrupted user or process is contained.

**Psychological acceptability** means that security mechanisms should not interfere unduly with the work of users, while at the same time meet the needs of those who authorize access. Security mechanisms that excessively hinder the usability or accessibility of resources are likely to be turned off.

These principles come from the paper [Design Principles for Secure Systems](#)

### 1.19 Security Mindset Quiz



#### Security Mindset Quiz

What security weakness was exploited to enable Stuxnet malware to compromise Iran's nuclear plant networks?

- Out of date anti-virus system
- Disloyal employees or poor judgement by humans
- Weak security controls, such as easy to guess passwords

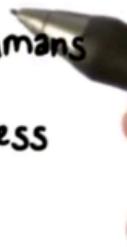
## 1.20 Security Mindset Quiz Solution



### Security Mindset Quiz

What security weakness was exploited to enable Stuxnet malware to compromise Iran's nuclear plant networks?

- Out of date anti-virus system
- Disloyal employees or poor judgement by humans
- Weak security controls, such as easy to guess passwords



## 2 Software Security

### 2.1 Software Vulnerabilities

A common vulnerability that we are going to discuss is a **buffer overflow**.

A buffer overflow occurs when the amount of memory allocated for a piece of expected data is insufficient (too small) to hold the actual received data. As a result, the received data “runs over” into adjacent memory, often corrupting the values present there.

Specifically, **stack buffer overflows** are buffer overflows that exploit data in the **call stack**.

During program execution, a **stack** data structure, known as the call stack, is maintained. The call stack is made up of **stack frames**.

When a function is called, a stack frame is pushed onto the stack. When the function returns, the stack frame is popped off of the stack.

The stack frame contains the allocation of memory for the local variables defined by the function and the parameters passed into the function.

A function call involves a transfer of control from the calling function to the called function. Once the called function has completed its work, it needs to pass control back to the calling function. It does this by holding a reference to the **return address**, also present in the stack frame.

Stack buffer overflows can be exploited through normal system entry points that are called legitimately by non-malicious users of the system. By passing in carefully crafted data, however, an attacker can trigger a stack buffer overflow, and potentially gain control over the system's execution.

## 2.2 Vulnerable Program

The following program - which roughly resembles a standard password checking program - is vulnerable.

```
1 #include <stdio.h>
2 #include <strings.h>
3
4 int main(int argc, char *argv[]) {
5     int allow_login = 0;
6     char pwdstr[12];
7     char targetpwd[12] = "MyPwd123";
8
9     gets(pwdstr);
10    if(strncmp(pwdstr, targtpwd, 12) == 0)
11        allow_login = 1;
12
13    if(allow_login == 0)
14        printf("Login request rejected");
15    else
16        printf("Login request allowed");
17 }
```

We have allocated space for `int` named `allow_login` that is initially set to 0.

In addition, we have allocated space for a user-submitted password (`pwdstr`) and a target password (`targetpwd`).

We then ask the user for their password (`gets`). Their response gets read into `pwdstr` and if `pwdstr` matches `targetpwd` (via `strncmp`), we set `allow_login` to 1.

Finally, if `allow_login` is 0, we print “Login request rejected”. Otherwise, we print “Login request allowed”.

## 2.3 Stack Access Quiz



### Stack Access Quiz

Check the lines of code, when executed, accesses addresses in the stack frame for main():

```
int main(int argc, char *argv[]) {
    int allow_login = 0;
    char pwdstr[12];
     char targetpwd[12] = "MyPwd123";
     gets(pwdstr);
     if (strncmp(pwdstr,targetpwd, 12) == 0)
         allow_login = 1;

     if (allow_login == 0)
         printf("Login request rejected");
     else
         printf("Login request allowed");
}
```



## 2.4 Stack Access Quiz Solution

The slide features a logo of two crossed swords on the left. In the center, the title "Stack Access Quiz" is written in blue. Below it, handwritten text reads: "Check the lines of code, when executed, accesses addresses in the stack frame for main():". To the right of the text is a photograph of a person's hand holding a black pen.

```
int main(int argc, char *argv[]) {
    int allow_login = 0;
    char pwdstr[12];
 char targetpwd[12] = "MyPwd123";
 gets(pwdstr);
 if (strcmp(pwdstr,targetpwd, 12) == 0)
     allow_login = 1;

 if (allow_login == 0)
     printf("Login request rejected");
 else
     printf("Login request allowed");
}
```

Since `allow_login`, `pwdstr` and `targetpwd` are all local variables to `main`, any access of them will access memory locations inside the stack frame for `main`.

The only lines of code that don't access the stack frame for `main` are the calls to `printf`, (which create a new stack frame), and `else`.

## 2.5 Understanding the Stack

There are two things you can do with a stack: push and pop.

The stack grows when something is pushed onto it, and shrinks when something is popped off of it.

The current “top” of the stack is maintained by a stack pointer, which points to different memory locations as the stack grows and shrinks.

We can assume that the stack grows from high (numerically larger) addresses to low (numerically smaller) addresses.

This means that the stack pointer points to the highest memory address at the beginning of program execution, and decreases as frames are pushed onto the stack.

## 2.6 Attacker Code Quiz



### Attacker Code Quiz

What type of password string could defeat the password check code? (Check all that apply)

```
#include <stdio.h>
#include <strings.h>

int main(int argc, char *argv[]) {
    int allow_login = 0;
    char pwdstr[12];
    char targetpwd[12] = "MyPwd123";
    gets(pwdstr);
    if (strncmp(pwdstr,targetpwd, 12) == 0)
        allow_login = 1;

    if (allow_login == 0)
        printf("Login request rejected");
    else
        printf("Login request allowed");
}
```

- Any password of length greater than 12 bytes that ends in '123'
- Any password of length greater than 16 bytes that begins with 'MyPwd123'
- Any password of length greater than 8 bytes



## 2.7 Attacker Code Quiz Solution



### Attacker Code Quiz

What type of password string could defeat the password check code? (Check all that apply)

```
#include <stdio.h>
#include <strings.h>

int main(int argc, char *argv[])
{
    int allow_login = 0;
    char pwdstr[12];
    char targetpwd[12] = "MyPwd123";
    gets(pwdstr);
    if (strncmp(pwdstr,targetpwd, 12) == 0)
        allow_login = 1;

    if (allow_login == 0)
        printf("Login request rejected");
    else
        printf("Login request allowed");
}
```

Any password of length greater than 12 bytes that ends in '123'

Any password of length greater than 16 bytes that begins with 'MyPwd123'

Any password of length greater than 8 bytes

Remember that the stack pointer moves down in memory as space is allocated. This means that `allow_login` will receive memory starting at the highest feasible address, and `pwdstr` will receive memory starting at the next highest feasible address.

Suppose both `int` and `char` occupy 1 byte. `allow_login` may be allocated 1 byte of space starting at memory address 1000. `pwdstr` may be allocated 12 bytes of space starting at memory address 988.

If the user enters a password longer than 12 bytes, the remaining bytes will overflow into the memory allocated to `allow_login`, effectively overwriting its value.

Since login will succeed if `allow_login` is anything but 0 (i.e. not a fail-safe default), this overflow will almost certainly lead to access being granted.

## 2.8 Attacker Code Execution Part 1

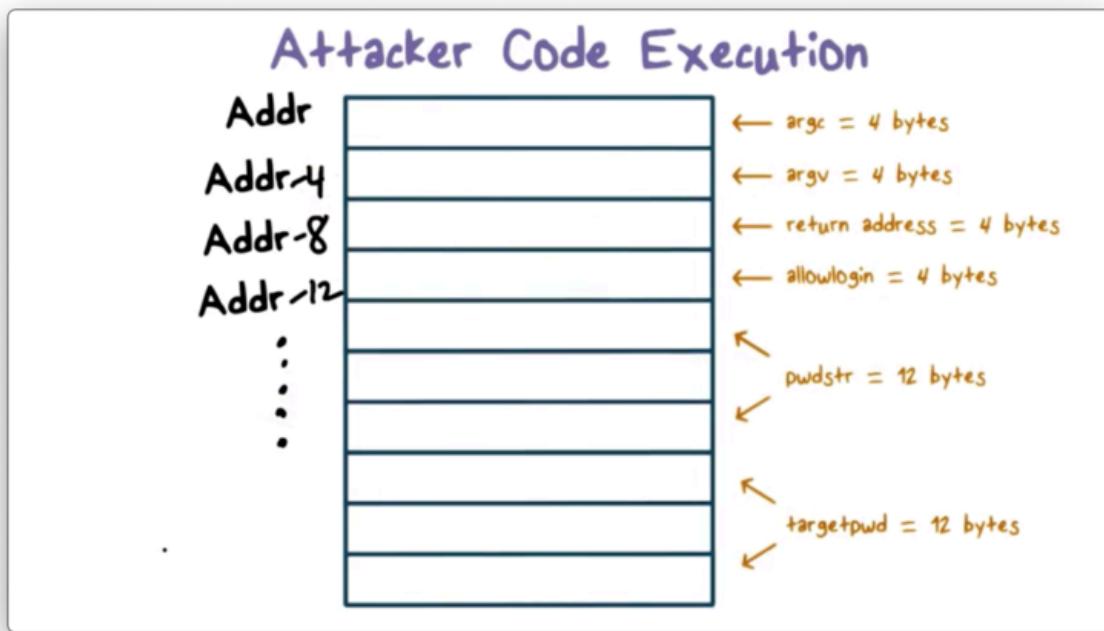
If the attacker guesses the correct password and types that as input to the program, login will be allowed.

If the attacker guesses the wrong password - which fits into the allocated buffer - there will be no overflow and login will be rejected.

These are the two basic outcomes for a naive attack: either the attacker guesses correctly and access is granted or the attacker guesses incorrectly and access is denied.

## 2.9 Attacker Code Execution Part 2

In order to understand how an attacker can use buffer overflow to gain control of this program, we first need to look at how the data associated with this program is laid out on the stack.



We know that the stack grows from higher memory addresses to lower memory address.

When we make the function call to `main`, we push the arguments `argc` (4 bytes) and `argv` (4 bytes) onto the stack.

Assuming the top of the stack is located at memory address `addr`, the stack pointer points to `addr - 8` after pushing these argument onto the stack.

Next, we have to push the return address (4 bytes) onto the stack. Every time we make a function call, we have to push the return address onto the stack so the program knows where to continue execution within the calling function once the called function completes.

After pushing the return address, the stack pointer points to `addr - 12`.

Finally, we allocate space for `allowLogin` (4 bytes), `pwdstr` (12 bytes) and `targetpwd` (12 bytes).

If `pwdstr` is within 12 bytes, it will occupy only the memory allocated to it. If `pwdstr` is longer than 12 bytes, it will exhaust the 12 bytes allocated to it, and will overflow into the space allocated for `allowLogin`.

The reason `pwdstr` overflows into `allowLogin` and not `targetPwd` is because occupation of memory occurs sequentially, from lower memory addresses to higher memory address. *Note: this is the opposite of the direction in which the stack grows.*

If the supplied value for `pwdstr` is greater than 16 bytes, `pwdstr` will also overwrite the return address.

## 2.10 Attacker Code Execution Part 3

As an attacker, we want to direct program control to some location where the attacker can craft some code.

If the attacker writes more than 16 bytes to `pwdstr`, the buffer allocated to `pwdstr` will overflow and will overwrite the return address.

If we know the address of the code that we want to execute, we can craft our input carefully, such that the existing return address gets overwritten with the address we want.

If we do this, what will happen?

Remember, the point of the return address is to give the function a location to transfer control to when it is done executing. If we overwrite that address, the function will “return” to the address we supply and begin executing instructions from that address.

## 2.11 Buffer Overflow Quiz



### Buffer Overflow Quiz

Which of these **vulnerabilities** applies to the code:

- The target password was too short, this made it easy to overflow the buffer.
- The code did not check the input and reject password strings longer than 12 bytes.
- The code did not add extra, unused variables.  
If this is done then when the user inputs a long password, it won't overflow into the return address.



## 2.12 Buffer Overflow Quiz Solution



### Buffer Overflow Quiz

Which of these **vulnerabilities** applies to the code:

- The target password was too short, this made it easy to overflow the buffer.
- The code did not check the input and reject password strings longer than 12 bytes.
- The code did not add extra, unused variables.  
If this is done then when the user inputs a long password, it won't overflow into the return address.

The first answer is wrong. The target password can be as long as you'd like, but if the attacker submits a longer password, the overflow will still happen.

The third answer is also wrong. Besides the fact that you shouldn't ever really add useless variables, these variables will only provide a finite amount of distance between the user-filled buffer and the return address. With a long enough password, the attacker can still overwrite the return address.

Only the second answer is correct. The overflow happens precisely because input larger than the space allocated for that input is not rejected by the program.

## 2.13 Shellcode

The code that the attacker typically wants to craft is code that is going to launch a **command shell**. This type of code is called **shellcode**.

The execution of the shellcode creates a shell which allows the attacker to execute arbitrary commands.

You can write the shellcode in C, like this:

```
1 int main (int argc, char *argv[]) {
```

```
2     char *sh;    char *args[2];
3
4     sh = "/bin/sh"; args[0] = sh;    args[1] = NULL;
5     execve(sh, args, NULL);
6 }
```

The “magic” here is `execve`, which replaces the currently running program with the invoked program - in this case, the shell at `/bin/sh`.

While the code can be written in C, it must be supplied to the vulnerable program as compiled machine code, because it is going to be stored in memory as actual machine instructions that will be executed once control is transferred.

## 2.14 Shellcode Privileges

The vulnerable program is running with some set of privileges before transfer is controlled to the shellcode.

When control is transferred, what privileges will be used?

The shellcode will have the same privileges as the host program.

This can be a set of privileges associated with a certain user and/or group. Alternatively, if the host program is a system service, the shellcode may end up with root privileges, essentially being handed the “keys to the kingdom”.

This is the best case scenario for the attacker, and the worst case scenario for the host.

## 2.15 NVD Quiz



### National Vulnerability Database (NVD) Quiz

- How many CVE (Common Vulnerability and Exposure) vulnerabilities do you think NVD will have?   
[1] Close to 500, [2] A few thousand, [3] Close to 70000
- If you search the NVD, how many buffer overflow vulnerabilities will be reported from the last three months?   
[1] less than 10, [2] Several hundred, [3] Close to one hundred
- How many buffer overflow vulnerabilities in the last 3 years?   
[1] Over a thousand, [2] fifty thousand, [3] five hundred

## 2.16 NVD Quiz Solution



### National Vulnerability Database (NVD) Quiz

- How many CVE (Common Vulnerability and Exposure) vulnerabilities do you think NVD will have? [3]  
[1] Close to 500, [2] A few thousand, [3] Close to 70000
- If you search the NVD, how many buffer overflow vulnerabilities will be reported from the last three months? [3]  
[1] less than 10, [2] Several hundred, [3] Close to one hundred
- How many buffer overflow vulnerabilities in the last 3 years? [1]  
[1] Over a thousand, [2] fifty thousand, [3] five hundred

## 2.17 Return to libc

So far we have talked about stack buffer overflows. There are other variations of buffer overflows.

The first variation is called **return-to-libc**.

When we talked about shellcode, the goal was to overflow the return address to point to the location of our shellcode, but we don't need to return to code that we have explicitly written.

In **return-to-libc**, the return address will be modified to point to a standard library function. Of course, this assumes that you will be able to figure out the address of the library function.

If you return to the right kind of library function and you are able to set up the arguments for it on the stack, then you can execute any library function any parameters.

For example, if you point to the address of the `system` library function, and pass something like `/bin/sh`, you should be able to open a command shell.

The main idea with return-to-libc is that we have driven our exploit through instructions already present on the system, as opposed to supplying our own.

## 2.18 Heap Overflows

An overflow doesn't have to occur to memory associated with the stack. A **heap overflow** describes buffer overflows that occur in the heap.

One crucial difference between the heap and the stack is that the heap does not have a return address, so the traditional stack overflow / return-to-libc mechanism won't work.

What we have in the heap are function pointers, which can be overwritten to point to functions that we want to execute.

Heap overflows require more sophistication and more work than stack overflows.

## 2.19 OpenSSL Heartbleed

So far, when we have talked about buffer overflow, we have talked about writing data; specially, inputting data into some part of memory and overflowing the memory that was allocated to us.

Overflows don't just have to be associated with writing data. For example, if a variable has 12 bytes, but we ask to read 100 bytes, the read will continue past the original 12 bytes and return data in subsequent memory locations.

The [OpenSSL Heartbleed vulnerability](#) did just this. It read past an assumed boundary (due to insufficient bounds checking) and was exploited to steal some important information - like encryption keys - that resided in adjacent memory.

## 2.20 Defense Against Buffer Overflows

Naturally, we shouldn't write code with buffer overflow vulnerabilities, but if such code is out there deployed on systems, we need to find ways to defend against attacks that exploit these vulnerabilities.

For instance, choice of programming language is crucial. There are languages where buffer overflows are not possible.

These languages:

- are strongly typed
- perform automatic bounds checking
- perform automatic memory management

Languages that have these features are referred to as "safe" languages and include languages like Java and C++.

## 2.21 Safe Languages

If we choose a “safe” language, buffer overflows become impossible due to the checks the language performs at runtime.

For example, instead of having to perform bounds checking explicitly, programmers can rest assured knowing that the language runtime will perform the check for them.

So, why don’t we use these languages for everything?

One drawback for these languages is performance degradation. The extra runtime checks slow down the execution of your program.

## 2.22 Unsafe Languages

When using “unsafe” languages, the programmer takes on the responsibility of preventing potential buffer overflow scenarios.

One way to do that is by checking all input to ensure that it conforms to expectations. Assume that all input is evil.

Another strategy to reduce the possibility of exploitation is to use safer functions that perform bounds checking for you. One such list of safe replacements for common library functions in C can be found [here](#).

A third strategy is to use automated tools that analyze a program and flag any code that looks vulnerable.

These tools look for code patterns or unsafe functions and warn you which code fragments may be vulnerable for exploitation.

One issue with automated analysis tools is that they may have many false positives (flagging something that is not an issue), and may even have false negatives (not flagging something that is an issue). No tool should replace thoughtful programming.

There is no excuse for writing code that is insecure!

## 2.23 Strongly Vs Weakly Typed Language Quiz



### Strongly vs. Weakly Typed Language Quiz

Strongly typed languages help reduce software vulnerabilities. Determine which of the following options apply to strongly typed languages and which are for weakly typed. (Use 's' or 'w').

- Any attempt to pass data of incompatible type is caught at compile time or generates an error at runtime.
- An array index operation  $b[k]$  may be allowed even though  $k$  is outside the range of the array.
- It is impossible to do "pointer arithmetic" to access arbitrary area of memory.

## 2.24 Strongly Vs Weakly Typed Language Quiz Solution



### Strongly vs. Weakly Typed Language Quiz

Strongly typed languages help reduce software vulnerabilities. Determine which of the following options apply to strongly typed languages and which are for weakly typed. (Use 's' or 'w').

- S Any attempt to pass data of incompatible type is caught at compile time or generates an error at runtime.
- W An array index operation b[k] may be allowed even though k is outside the range of the array.
- S It is impossible to do "pointer arithmetic" to access arbitrary area of memory.

## 2.25 Analysis Tools

A number of [source code analysis](#) tools are available. These tools analyze the source code of your application, and can flag potentially unsafe constructs and/or function usage.

Companies will often incorporate the use of these tools into their software development lifecycle to ensure that all code headed for production is audited before being released.

If you are attempting to analyze code that you didn't write, you may not have the source code available, at which point source code analysis tools obviously won't be helpful.

## 2.26 Stack Canary

One of the tricks that hackers use is to override the return address on the stack to point to some other code they want to execute.

During the execution of a function, however, there is no reason for the return address to be modified; that is, there is no reason a function should change where it returns during the middle of its execution.

As a result, if we can detect that the return address has been modified, we can show that a buffer overflow is being exploited and handle execution appropriately, likely with process termination.

How can we detect if the return address has been modified? We can use a **stack canary**, or a value that we write to an address just before the return address in a stack frame. If an overflow is exploited to overwrite the return address, the canary value will be overwritten with it.

All the runtime has to do, then, is to check if the canary value has changed when a function completes execution. If so, it can be sure that there is a problem.

What is nice about this approach is that the programmer doesn't have to do anything: the compiler inserts these checks. Of course, this means that the code may have to be recompiled with a compiler that has these features, a step which may come with its own issues.

## 2.27 ASLR

There are also OS-/hardware-based solutions which can help to thwart the exploitation of buffer overflow vulnerabilities.

The first technique that many operating systems use is **address space layout randomization** (ASLR).

Remember that one key job of the attacker is to be able to understand/approximate how memory is laid out within the stack or, in the case of return-to-libc, within a process's address space.

ASLR randomizes how memory is laid out within a process to make it very hard for an attacker to predict, even roughly, where certain key data structures and/or libraries reside.

Many modern operating systems provide [ASLR](#) support.

### 2.27.1 Non-executable stack

In the classic stack buffer overflow attack, the attacker writes shellcode to the stack and then overwrites the return address to point to that shellcode, which is then executed.

There is no legitimate reason for programs to execute instructions that are stored on the stack. One way to block executing shellcode off the stack is to make the stack non-executable.

Many modern operating systems implement such [executable-space protection](#).

## 2.28 Buffer Overflow Attacks Quiz



### Buffer Overflow Attacks Quiz

- Do stack canaries prevent return-to-libc buffer overflow attacks?  Yes  No
- Does ASLR protect against read-only buffer overflow attacks?  Yes  No
- Can the OpenSSL heartbleed vulnerability be avoided with non-executable stack?  Yes  No

## 2.29 Buffer Overflow Attacks Quiz Solution



### Buffer Overflow Attacks Quiz

- Do stack canaries prevent return-to-libc buffer overflow attacks?  Yes  No
- Does ASLR protect against read-only buffer overflow attacks?  Yes  No
- Can the OpenSSL heartbleed vulnerability be avoided with non-executable stack?  Yes  No

Stack canaries do prevent return-to-libc buffer overflow attacks, because stack canaries prevent return address overwriting. Without overwriting the return address, a function can only return to the function that called it.

ASLR does not protect against read-only buffer overflow exploits. ASLR only makes it harder to supply key addresses in write-based buffer overflow exploits.

Heartbleed cannot be avoided by using a non-executable stack. Heartbleed is a read-based buffer overflow exploit, and the attack did not involve injecting any machine instructions onto the stack.

## 3 Operating System Security

### 3.1 Operating Systems Defined

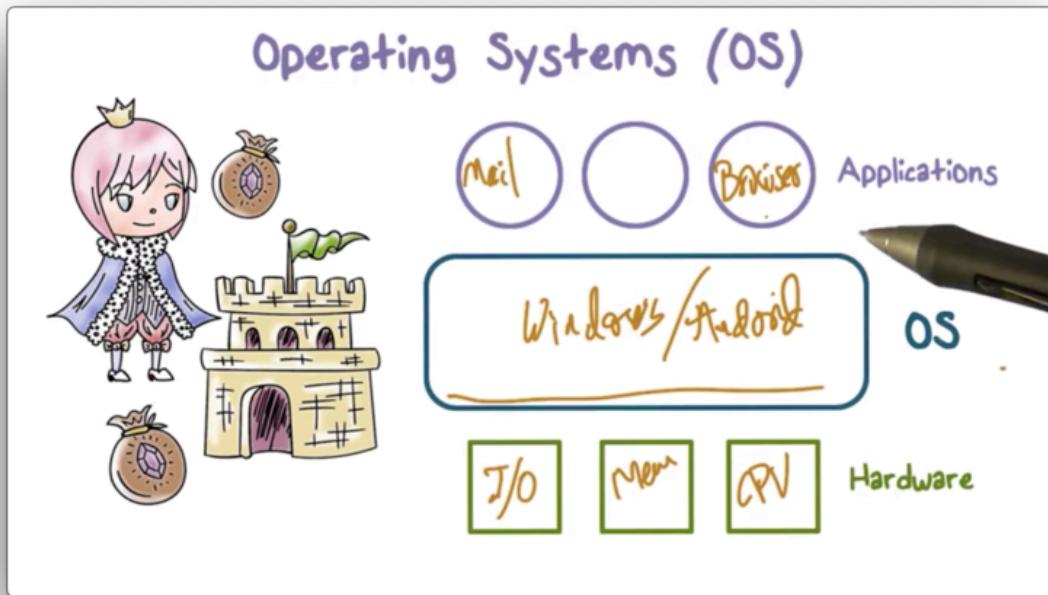
Operating systems play a really important role in computer systems. When we talk about computers, we often refer to the operating system when naming a device; for example, a Windows machine or an iOS device.

Operating systems play a critical role when it comes to protecting and securing resources present in our computer systems.

When we are looking at the arrangement of a computer system, we first start at the level of hardware. At this layer, we have the CPU, physical memory, and other I/O devices.

Direct use of hardware is really difficult. Instead of managing the hardware explicitly, we run a “program” called an **operating system** - such as Linux, Windows, or macOS - that handles the access and management of the low-level hardware resources.

The applications that we as users directly deal with - browsers, word processors and the like - sit on top of the operating system and interact with the physical resources through this intermediary.



As a mediator between applications and the real, hardware resources, the operating system unsurprisingly plays an important role in the security of computer systems.

## 3.2 Operating Systems

Operating systems provides key functionality to the applications which rely on them.

### 3.2.1 Abstraction of hardware

The hardware is not easy to use directly. If you want to program your applications to work with hardware directly, development becomes infinitely more complex.

Thankfully, the operating system abstracts the low-level hardware APIs into simpler higher-level entities.

For example, persistent data in a computer system is stored on the disk in disk blocks. Instead of having to worry about disk management and the location/traversal of blocks, the operating system provides us with the convenient *file* abstraction.

These high-level abstractions allow us to interact with the underlying resources in ways that are simpler to understand and reason about, which in turn makes it easier to build applications on top of the operating system.

### **3.2.2 Controlled hardware access**

The abstractions that the operating system provides to the higher applications are implemented using the real physical resources provided by the hardware.

The physical resources are shared across the application and, as a result, must be accessed in a controlled fashion.

Without controlled access, it would be all too easy for one application to overwrite data currently in use by another application. Alternatively, one application could hog the CPU and starve the other applications of execution.

Controlled access ensures that the applications across the system run cohesively.

### **3.2.3 Process Isolation**

All of the different applications are running on the same system and sharing the same physical resources.

Yet, from the perspective of an individual application or process, it seems as though the process has complete and exclusive access to the entire hardware underneath.

The operating system isolates processes from one another, so each application has the feeling that it is the only one running.

Processes need not necessarily be aware of other processes.

This is important because processes may not trust each other. The operating systems isolates processes from one another while providing a trusted interface (itself) with which a process can interact with the underlying physical system.

### 3.3 Need for Trusting an Operating System

The operating system has direct control over basically everything in the system. It can manipulate processes running on top of it, and access hardware beneath it.

The operating system has the “keys to the kingdom”, so to speak, so it is important that the operating system function as a **trusted computing base** (TCB) for our computer systems.

The following requirements should be met by a trusted computing base.

#### 3.3.1 Complete mediation

A TCB must mediate all access to the underlying protected resources. All requests for access to hardware must pass through the operating system. Hardware use cannot circumvent the OS.

#### 3.3.2 Tamper-proof

A TCB must also be tamper-proof. If untrusted code executing on top of the OS can tamper with the OS then we can no longer trust the OS.

#### 3.3.3 Correctness

Finally, a trusted system is a correct system. If you are going to rely on the OS to ensure that your protected resources get used in a proper way, it is imperative that this access control is implemented correctly.

### 3.4 OS and Resource Protection

How does the operating system mediate requests for protected resources?

A request always has two main components:

- the entity making the request
- the target resource being requested

Identifying the entity making the request is known as **authentication**. Authentication establishes on whose behalf a current application or process is running.

Once we authenticate the requesting entity, we employ **authorization** to grant or deny access to the requested resource.

Note that the operating system doesn't autonomously decide who accesses a given resource or not. System administrators must define **policies** that describe who can access what, and under which circumstances.

The operating system implements a policy by performing access checks and either granting or denying access in accordance with the policy.

### 3.5 Secure OS Quiz 1



### Secure OS Quiz #1

A computer vendor ad claimed that its computers (including the OS they ran) were more secure. This claim could be based on one or more of the following:

- This vendor's more secure OS met TCB requirements while the others did not.
- The two OS were similar as far as security was concerned but one was not as big a target.
- The more secure OS could be much simpler than the other one.



### Mac vs PC Security

### 3.6 Secure OS Quiz 1 Solution



#### Secure OS Quiz #1

A computer vendor ad claimed that its computers (including the OS they ran) were more secure. This claim could be based on one or more of the following:

- This vendor's more secure OS met TCB requirements while the others did not.
- The two OS were similar as far as security was concerned but one was not as big a target.
- The more secure OS could be much simpler than the other one.



### 3.7 Secure OS Quiz 2



#### Secure OS Quiz #2

A system call allows application code to gain access to functionality implemented by the OS. A system call is often called a protected procedure call.



Is the cost of a system call:

- the same as a regular call
- higher than a regular call



### 3.8 Secure OS Quiz 2 Solution

 **Secure OS Quiz #2**

A system call allows application code to gain access to functionality implemented by the OS. A system call is often called a protected procedure call.



Is the cost of a system call:

- the same as a regular call
- higher than a regular call



A system call requires control transfer from the calling process into the OS, which then must perform authentication/authorization checks before granting access and transferring control back.

This is more costly than a regular call, which incurs none of this overhead.

### 3.9 Secure OS Quiz 3



#### Secure OS Quiz #3

Complete mediation ensures that the OS cannot be bypassed when accessing a protected resource. How does the OS know who is making the request for the resource?



- Process runs on behalf of a user who must have previously logged in
- Requested resource allows us to find out who must be requesting it

### 3.10 Secure OS Quiz 3 Solution

 **Secure OS Quiz #3**

Complete mediation ensures that the OS cannot be bypassed when accessing a protected resource. How does the OS know who is making the request for the resource?

Process runs on behalf of a user who must have previously logged in

Requested resource allows us to find out who must be requesting it

Processes run on behalf of users. Users must login to the system to run applications/processes.

### 3.11 How Can We Trust an OS?

How is an operating system tamper-proof? In other words, how does an operating system maintain its isolation and integrity despite untrustworthy applications and users executing on top of it?

The operating system needs help from the hardware. Particularly, the hardware needs to protect the memory where the operating system resides, and block unprivileged attempts from interacting with memory at those locations.

The CPU executes in different **execution modes** (also known as **execution rings**), which serve as execution environments with varying privileges. The most privileged ring is ring 0, and privileges are revoked as you move up to higher rings.

Many architectures support more than two (user and system) execution rings.

Thanks to these rings, the processor is aware of which kind of code it is executing - be it untrusted user code or trusted system code - and knows that it shouldn't access code or data belonging to a higher privilege level.

Certain privileged hardware instructions can only be executed when the processor is executing in ring 0. These instructions are usually ones that provide direct access to the hardware: manipulating memory, interacting with devices, and so forth.

### 3.12 System Calls From User to OS Code

System calls are used to transfer control between user and system code.

These calls cannot be arbitrary. The operating system exposes an API which enumerates the specific ways in which applications can interact with it. For example, here are the [system calls](#) for Linux.

These calls must enter the operating system in a controlled fashion. They come through **call gates**, which are accompanied by a processor privilege ring change on system entry and exit.

Crossing this boundary means that the operating system will have to change some data structures that keep track of memory mappings, because we will be able to access memory now that we couldn't access before. In addition, certain registers will have to be saved, while others have to be loaded.

This used to happen via a hardware interrupt/[trap](#) into the system from user mode, but in [x86 architectures](#), we have explicit instructions to cross the system boundary: [syseenter](#) to enter the operating system during the system call and [sysexit](#) to return from the system call.

The extra work of adjusting privilege rings, changing memory mappings, and loading/saving additional registers makes these calls more costly.

### 3.13 Untrusted User Code Isolation

Untrusted user code has to be isolated from the system code. How can we achieve this isolation?

All the processor does is fetch the next instruction and execute it. Some of these instructions may be read/write or load/store requests. If the processor receives one of these requests supplied with a memory address belonging to the operating system, what stops it from executing that instruction?

We can rely on the hardware to protect memory. If you are running in user mode, the hardware will block an attempt to generate an address and complete a read or write in a memory location that belongs to the operating system.

This hardware protection also applies to processes trying to access memory belonging to other processes. The hardware enforces that processes only access memory that has been made available to it, blocking access to any other memory locations.

### 3.14 User Isolation Quiz



#### User Isolation Quiz

Which of these methods have been shown to allow hacker access to 'secure' memory belonging to the OS?

- Modification of firmware by Thunderstrike malware via malicious devices that connect via Mac's Thunderbolt interface
- Exploiting the 'refresh' mechanism of a Dynamic RAM for privilege escalation
- Exploiting OS code buffer overflow vulnerability

### 3.15 User Isolation Quiz Solution



## User Isolation Quiz

Which of these methods have been shown to allow hacker access to 'secure' memory belonging to the OS?

- Modification of firmware by Thunderstrike malware via malicious devices that connect via Mac's Thunderbolt interface
- Exploiting the 'refresh' mechanism of a Dynamic RAM for privilege escalation
- Exploiting OS code buffer overflow vulnerability

### 3.16 Address Space

From a process's point of view, it has the entire computer to itself. It isn't aware that it shares physical memory with other processes.

This is made possible by the **address space** abstraction that the operating system creates for a process upon creating the process itself.

An address space contains a sequence of contiguous memory locations (from zero to some max) that a process can address and utilize for its own data and instructions.

*The address space serves as the unit of isolation between processes sharing the same physical hardware resources.*

The memory that a process views may be larger than the actual physical memory present on the system, a feat made possible by **virtual memory**.

The address space usually contains  $2^{32}$  addresses (for a 32-bit architecture) or  $2^{64}$  addresses (for a 64-bit architecture).

### 3.17 Unit of Isolation

Physical addresses point to locations in physical memory, but addresses in an address space don't directly reference physical locations. Instead, these **logical addresses** must be translated into physical addresses by the operating system.

For example, the operating system might maintain a map that connects logical address 2000 in process A's address space to physical address 5000.

When process A is running, its address space maps to certain physical addresses in memory. When process B is running, its address space maps to different physical address in memory.

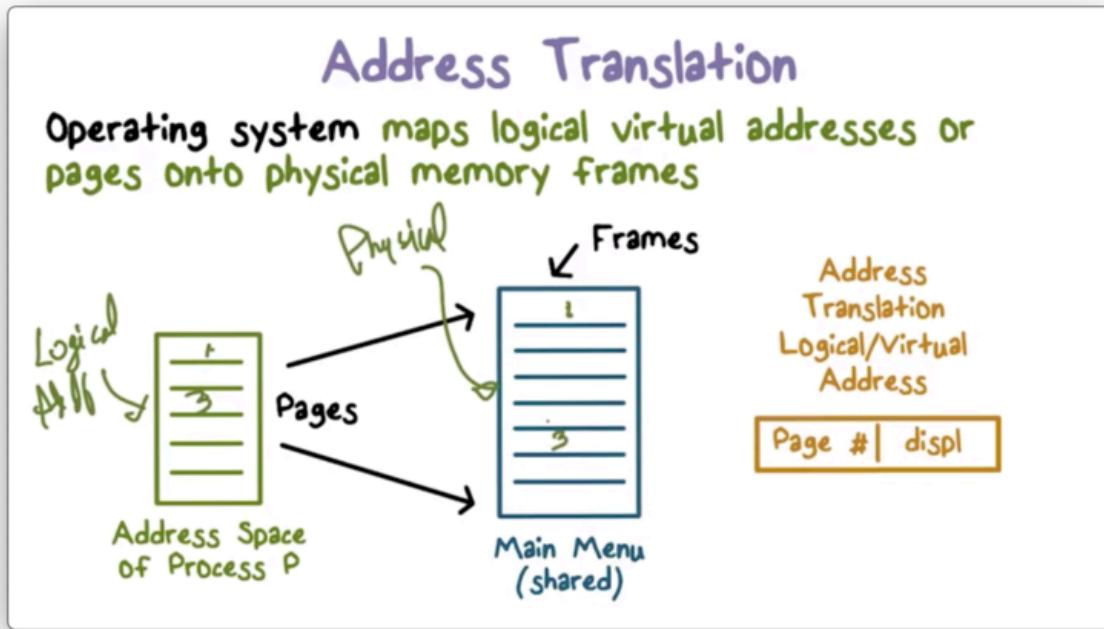
Even though the physical addresses corresponding to the address spaces of two processes are often interspersed, each process views its address space as contiguous and exclusive.

### 3.18 Address Translation

The reason that translation has to happen is that the address space that the process is provided is just an abstraction of the physical memory layout underneath.

For example, the process thinks it has access to a chunk of contiguous memory - which it doesn't - and in some cases the process may think that it has access to more memory than is currently physically available.

Naturally, the operating system needs to maintain a system to map between the abstraction and the reality.



This mapping process does not occur byte for byte. The number of mappings we would have to maintain if this were the case would not be scalable.

Instead, we divide the address space into larger chunks, called **pages**. A common page size is 4kb.

Thus, we can think about representing any address within the address space as a page number and an offset within that page.

A **page table** maintains the mapping between logical pages and physical pages. The operating system is responsible for building, maintaining, and protecting these tables.

When a process needs to retrieve data from an address in its address space, the operating system must first map that logical page to the correct physical page and carry the offset.

For example, the page table might have an entry that maps logical page 3 to physical page 8. This means that if a process is requesting some byte at offset  $d$  from the start of page 3, the operating system will fetch the byte at offset  $d$  from the start of physical page 8.

By using pages, we can reduce the number of entries in our lookup table from the number of addresses in the address space to that number divided by the page size.

For example, with 4kb pages and 32-bit addresses, we can reduce the number of entries in the page table from  $2^{32}$  to  $2^{20}$  ( $2^{32} / 2^{12}$ )

Virtual address translation - as maintained by page tables - ensures that a process can only access physical memory for which a corresponding logical address mapping exists in its page table.

### 3.19 Code Protection

The operating system will not map a virtual page of process A to a physical page that has been given to process B, unless the two processes wish to share memory.

Put another way: two page tables, each for a different process, will not contain a mapping to the same physical page at the same time.

What this means is that it is impossible for process A to access physical memory belonging to process B, because process A has no way to address that memory.

### 3.20 Process Protection through Memory Management

Protecting processes from each other and protecting the operating system from untrusted process code follow the same mechanism.

Page table lookups prevent a process from accessing any physical pages that belong to the operating system, since the operating system has not explicitly mapped those locations into the process's address space.

This translation process is so important that we typically have a piece of hardware called a **memory management unit** (MMU), that helps the OS perform this translation efficiently, using memory caches such as [translation look-aside buffers](#).

In addition to resolving a physical address from a virtual address, a page table entry can provide information about different types of memory access available for that page.

For example, an entry may store some information as to whether a given page is available for reading, writing, execution, or some combination of the three. These **RWX bits** can limit the type of access that a process can perform on addressable memory.

As a result, interacting with memory really has two components: first, understanding which memory locations are even available to a process, and; second, understanding which types of accesses are permitted on those locations.

### 3.21 Revisiting Stack Overflow Quiz



#### Revisiting Stack Overflow Quiz

The stack can be exploited through:

- Overflowing the buffer to change the return address to alter program execution
- Pushing data onto the stack to overflow the stack into the heap
- Popping data off the stack to gain access to application code.



### 3.22 Revisiting Stack Overflow Quiz Solution



## Revisiting Stack Overflow Quiz

The stack can be exploited through: *on Stack*.

- Overflowing the buffer to change the return address to alter program execution *Write into a local*
- Pushing data onto the stack to overflow the stack into the heap
- Popping data off the stack to gain access to application code.

### 3.23 Preventing Malicious Code Execution

One of the strategies that we have talked about for protecting against a stack buffer overflow attack is to use a non-executable stack.

While a malicious user might still be able to inject malicious instructions onto the stack, those instructions will not actually be executed.

The operating system can achieve this by not writing an executable bit for any page table entry associated with the virtual addresses within the process's stack.

Windows, OS X and Linux all make the stack non-executable.

### 3.24 OS Isolation from Application Code Part 1

Protecting processes from each other and protecting the operating system from untrusted process code follow the same mechanism.

The operating system (kernel) resides in a portion of each process's address space.

The address space in which a process executes now has two sections: the user code/data and the kernel code/data. This partitioning exists for every process that we have in the system.

Whenever a process wants to access a portion of the address space that contains kernel data or code, the process must make a system call to traverse that boundary.

Page tables, protection bits, and execution rings all play a role in establishing and enforcing this segregation within the address space.

### **3.25 OS Isolation from Application Code Part 2**

For 32-bit Linux systems, the address space is 4GB. Of that 4GB, the lower 3GB is used for user code/data, and the top 1GB is used for the kernel.

Access to different portions of address space (user vs kernel) are governed by the current privilege ring in which a process is currently executing.

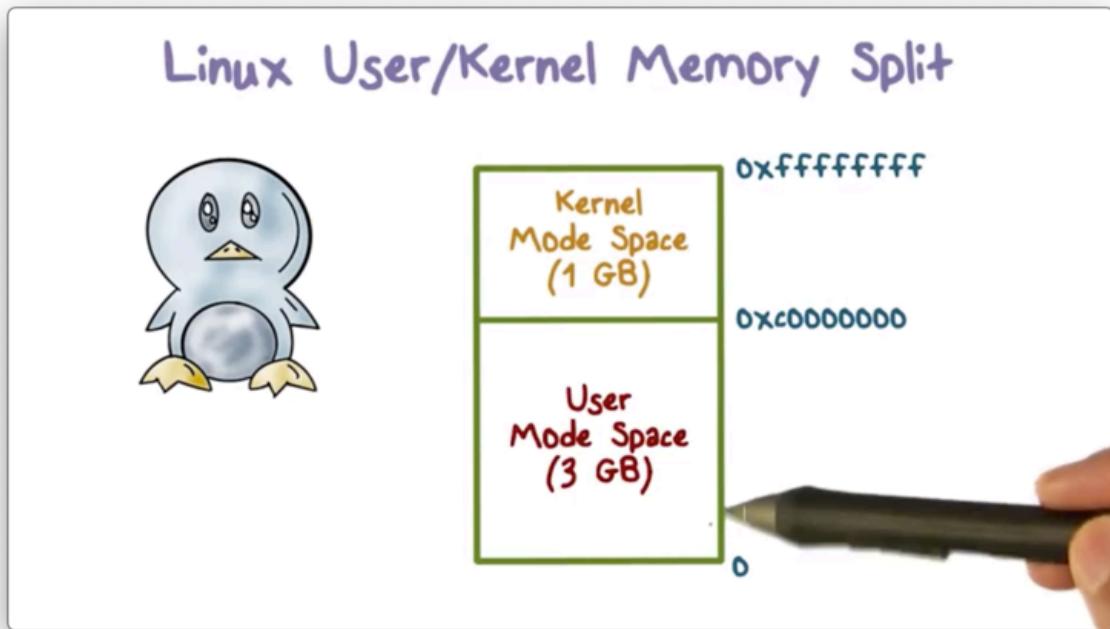
In x86 systems, a process must be in ring 0 to access kernel code. If it is operating in user mode (ring 3), it can't access the kernel portion of the address space.

The 1GB points to the same kernel code in each process (i.e. we aren't copying code for each process that is created), but the 3GB section is unique for each process.

While modern operating systems create this fence between untrusted code and trusted code, older operating systems didn't do this.

For example, MS-DOS does not have this separation. This means that any process could alter operating system code. Clearly, this is a vulnerability of this early OS.

### 3.26 Kernel Memory Split



### 3.27 Execution Privilege Level Quiz



#### Execution Privilege Level Quiz

For the following described functions, Should it be executed in the operating system or if it can be executed in application code running in user mode?

OS: User:

Switching CPU from one process to another when a process blocks.

Page fault handling

Changing who can access a protected resource such as a file

Setting up a new stack frame when an application program calls one of its functions



### 3.28 Execution Privilege Level Quiz Solution



## Execution Privilege Level Quiz

For the following described functions, Should it be executed in the operating system or if it can be executed in application code running in user mode?

OS: User:

- Switching CPU from one process to another when a process blocks.
- Page fault handling
- Changing who can access a protected resource such as a file
- Setting up a new stack frame when an application program calls one of its functions

### 3.29 Complete Mediation: The TCB

The second requirement of the a TCB is **complete mediation**.

Complete mediation means that the TCB must be aware of any and all accesses to resources it protects.

How do we implement complete mediation in operating systems?

We have to make sure that no protected resource - for example, a memory page or a file - can be accessed without going through the kernel.

We can achieve this by making the kernel act as a *reference monitor* that cannot be bypassed. This means that all references to underlying resources must be resolved by the operating systems in order to be valid.

We just saw an example of this with address spaces. A process can only access memory once the operating system translates its virtual reference into a real, physical memory location.

### 3.30 Complete Mediation: The User Code

Protected resources are implemented by the operating system, and the data structures that are utilized to access those resources - page tables, for example - live in the kernel portion of the address space.

While a process is executing user code, it can't access that portion of your address space. The process will need to switch to the system mode through a system call, and this transfer of control will be mediated by the operating system itself.

In addition, user code cannot access physical resources directly because interacting with those resources often requires access to privileged instructions that can only be executed in system mode.

### 3.31 Complete Mediation: OS

At user level, a process has access to virtual resources and the operating system exposes APIs for how those virtualized resources can be used.

For example, a process doesn't access a disk block directly. Instead, the operating system performs the disk I/O, and abstracts the disk controller operations into the *file* interface, which a process can use to read from and write to files.

Since virtual resources are just abstractions of physical resources, there must be a translation process whereby the virtual reference resolves into a physical reference.

If you want to access a disk block or a page of memory for example, you have to start with a file descriptor or a virtual address, and let the OS map these references into the appropriate physical resources.

This level of indirection added by the operating system makes it impossible for a process to directly reference physical resources and allows the OS achieve complete mediation.

### 3.32 Virtualization

We don't have to stop at virtualizing individual physical resources. The concept of translating virtual resources into physical resources can apply to the computer system as a whole, giving rise to [virtual machines](#).

Virtual machines provide functionality needed to execute entire operating systems, and an intermediary known as a hypervisor manages the hardware on behalf of one (or more!) operating systems running on top of it.

The main motivation for virtualization comes from the fact that operating systems are large and complex, and different applications may require different underlying operating systems.

One use of virtualization commercially is to free users from being limited to using one operating system and be able to use multiple operating systems on the same physical system.

### 3.33 Limiting the Damage of a Hacked OS

Remember that if an operating system is compromised, all applications running on that system will be compromised because they all share and trust the OS.

Virtualization can help limit the damage caused by a compromised/hacked operating system.

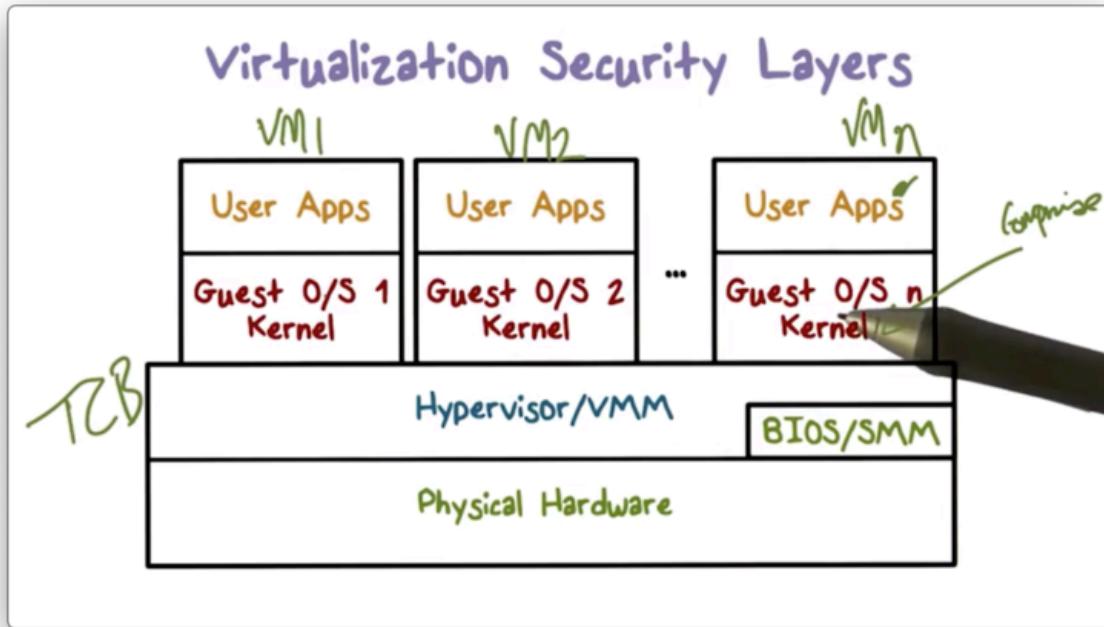
In virtualized systems, we introduce a **hypervisor** between the hardware and the operating systems. On top of the hypervisor, we support virtual machines, which have their own operating system - called the guest operating system - which itself can support a number of applications.

Compromise of an OS within one virtual machine only impacts the applications running inside of that virtual machine. The applications that are running in an adjacent virtual machine atop the same hypervisor are unaffected.

With virtualization we can achieve isolation between virtual machines - each with their operating system - where earlier we only had isolation between processes.

In this case, the TCB is now the hypervisor, which manages the hardware resources on behalf of the virtual machines.

### 3.34 Virtualization Security Layers



### 3.35 Correctness: The Final TCB Requirement

Correctness is important because compromise of the OS means that an attacker has access to all the physical resources: every memory page, every block on disc, etc.

Meeting the correctness requirement for operating systems is really hard. Operating systems are very complex, and we've already discussed how added complexity can make security much more difficult.

Virtualization can help with correctness by reducing complexity. The hypervisor can be much smaller and simpler than the operating system since all it has to do is partition physical resources among virtual machines.

### 3.36 TCB Requirements Quiz



#### TCB Requirements Quiz

An attack that exploits a vulnerability in an operating system turns off the check that is performed before access to a protected resource is granted.

What TCB requirement is violated as a result of this attack?

- Complete mediation
- Correctness
- Tamper-proof



### 3.37 TCB Requirements Quiz Solution



## TCB Requirements Quiz

An attack that exploits a vulnerability in an operating system turns off the check that is performed before access to a protected resource is granted.

What TCB requirement is violated as a result of this attack?

- Complete mediation
- Correctness
- Tamper-proof



In this case, we have tampered with the TCB by turning off the check. The access still proceeds through the operating system, and is still technically correct (i.e. the access wasn't permitted because of a bug).

### 3.38 Size of Security Code



#### Size of Security Code Quiz

Going from MS DOS to recent Windows operating systems, what is a rough estimate for the multiplier for the lines of code (e.g. multiplier is  $x$  if recent Windows OS is  $x$  times the number of lines of code in DOS)?

- Windows OS is 100x larger than MS DOS
- Windows OS is 500x larger than MS DOS
- Windows OS is 10,000x larger than MS DOS



### 3.39 Size of Security Code Solution



#### Size of Security Code Quiz

Going from MS DOS to recent Windows operating systems, what is a rough estimate for the multiplier for the lines of code (e.g. multiplier is  $x$  if recent Windows OS is  $x$  times the number of lines of code in DOS)?

- Windows OS is 100x larger than MS DOS
- Windows OS is 500x larger than MS DOS
- Windows OS is 10,000x larger than MS DOS



I think the point being made here is that the increase in complexity may be accompanied by an increase in vulnerability.

### 3.40 Hypervisor Code Size Quiz



### Hypervisor Code Size Quiz

The number of lines of code in a hypervisor is expected to be smaller. Xen is an open source hypervisor.

What is a rough estimate for the lines of code for the Xen hypervisor?



- 10,000
- 150,000
- 1,000,000



### 3.41 Hypervisor Code Size Quiz Solution

 **Hypervisor Code Size Quiz**

The number of lines of code in a hypervisor is expected to be smaller. Xen is an open source hypervisor.

What is a rough estimate for the lines of code for the Xen hypervisor?



<input type="checkbox"/>	10,000
<input checked="" type="checkbox"/>	150,000
<input type="checkbox"/>	1,000,000

A hand holding a pen is pointing towards the checked option.

Again, the argument being made here is that using a hypervisor as a TCB, with fewer lines of code than a full-fledged operating system, might be a more secure choice.

## 4 Authentication

### 4.1 What is Authentication?

We can understand the importance of authentication by going back to the conversation about trusted computing bases.

We have resources that need to be protected, and we can achieve this protection by using a trusted computing base that serves as a reference monitor.

Every request for every resource must be monitored, and the trusted computing base has to determine whether to accept or deny each request.

In order to make this decision, the TCB has to know the source of such requests, and **authentication** is the mechanism by which these sources are identified.

The first step in authentication is identification. Of course, a requestor cannot just claim whatever identity they'd like. As a result, the second step in authentication is identity verification.

For a concrete example, we identify ourselves when we supply our username to a login form, and we verify that identity when we supply our (secret) password.

Once we establish the source of a request, the next step is **authorization**. In authorization, the operating system decides whether the source of the request has the permission to access the resource they are requesting.

Once authentication and authorization are complete, then the operating system can allow access to the protected hardware resources.



The operating system needs to know who is making a request for a protected resource.

In a computer system, a request comes from a process, and each process runs on behalf of a certain user (also referred to as a *subject* or *principal*) within that computer system.

Therefore, authentication helps us answer the question: on whose behalf is the requesting process running?

We can answer this question with authentication. Before a user is allowed to interact with a computer system, they must first authenticate themselves - by logging in, for example.

Once authenticated, the OS is able to link any subsequent process or application resource requests back to the authenticated user.

## 4.2 Authentication Goals

When a legitimate user tries to authenticate herself, the system can demand some evidence, but when the right evidence is provided the system should allow the login to complete successfully.

This is called **availability**. The system is available to the user who is able to provide the sufficient evidence to support their claim to their identity.

In other words, we don't want to have any *false negatives*. This means that we don't want to have the situation where we deny access to a user who has provided sufficient evidence that they are who they claim to be.

On the other hand, we don't want to have a system that allows a malicious user to successfully impersonate a valid user.

In other words, we don't want to have any *false positives*. This means that we don't want to have a situation where we grant access to someone who is not who they claim to be.

Authentication systems that have low false positive rates are said to have **authenticity**. That is, these systems ensure that they only authenticate users with authentic claims to their identity.

## 4.3 Authentication Quiz



### Authentication Quiz

Check the correct answer from the choices.

We now have personal devices that are not shared across multiple users. What threats motivate the use of authentication in such devices?

Malware infection that may exfiltrate sensitive data

Loss of theft of the device

#### 4.4 Authentication Quiz Solution



#### Authentication Quiz

Check the correct answer from the choices.

We now have personal devices that are not shared across multiple users. What threats motivate the use of authentication in such devices?

- Malware infection that may exfiltrate sensitive data
- Loss of theft of the device

If someone steals your phone, you will be thankful for your lock screen/passcode.

#### 4.5 How is Authentication Implemented?

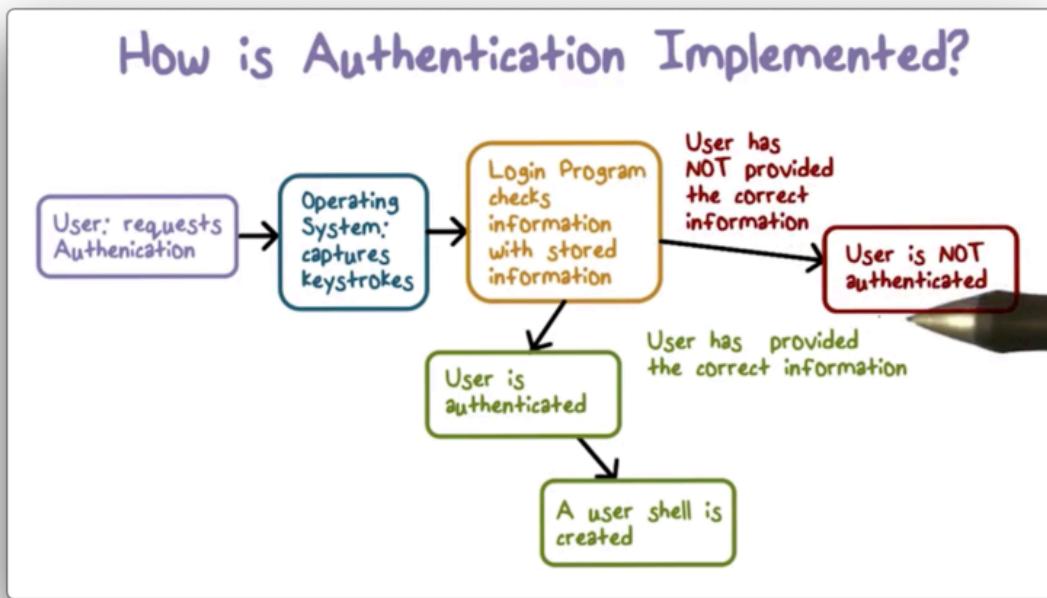
There are three basic methods for implementing authentication.

Authentication can be implemented using something a user knows. This could be a secret shared by the user and the operating system - a password, for example. The fact that a user is able to produce this secret during authentication is evidence that they are who they claim to be.

Authentication can also be implemented using something that the user has. For example, possession of a token or a smart card may be sufficient as a means of identity verification for some systems.

Finally, authentication can be implemented using something that the user is. For example, a system may be able to authenticate a user by digitizing and processing his or her fingerprint or voice. Such biometric data must be unique to each user for this scheme to be useful.

Here is a diagram illustrating the basic steps in an authentication interaction between a user and a system.



The user first comes to the system and requests access to it. In this request they claim an identity and provide a means of verifying that identity.

The operating system can take the supplied authentication information check it against stored information for the claimed identity.

If there is a match between what the system knows and what a user provides, then the user is authenticated, and a user shell is created.

If the check fails, then the system believes that the user has not provided the correct information. If so, the identity being claimed probably doesn't belong to the user who is asking for authentication.

In this case, the system doesn't have the right match and authentication fails.

## 4.6 Login Attacks Quiz



### Login Attacks Quiz

Check the correct answer from the choices.

An attacker correctly guesses Alice's password and logs in as her. Is this a case of...



- False positive
- True negative

#### 4.7 Login Attacks Quiz Solution



## Login Attacks Quiz

Check the correct answer from the choices.

An attacker correctly guesses Alice's password and logs in as her. Is this a case of...



False positive

True negative



Remember, the positive event is gaining access to the system. A false positive is gaining access erroneously. An attacker authenticating as someone else is a false positive.

#### 4.8 Implementation Quiz



#### Implementation Quiz

Check the correct answer from the choices.

A number of online banking systems send a limited lifetime PIN to your smartphone for you to be able to authenticate yourself to the bank. Is this an example of...

- Something you have  
Something you are



## 4.9 Implementation Quiz Solution



### Implementation Quiz

Check the correct answer from the choices.

A number of online banking systems send a limited lifetime PIN to your smartphone for you to be able to authenticate yourself to the bank. Is this an example of...

Something you have  
 Something you are



## 4.10 Threat Modeling of the Password Method

How can we attack password-based authentication?

A password is a secret shared between a legitimate user and the system, but that doesn't mean an attacker can't try to guess it. As obvious as it may seem, using common or weak passwords presents a real vulnerability in authentication systems.

When you authenticate with a computer system, how do you know that you are really talking to the system instead of a program that is impersonating the system? A hacker can create a fake login program and then steal your credentials as you type them in.

Finally, an attacker may install malicious software called a [keylogger](#) onto your machine, which will record all of the keyboard keys that you press. Keyloggers can be used to steal your password without have to impersonate the login program.

#### 4.11 Importance of a Trusted Path

When using a trusted path, a user can have confidence that there is no application between the user and the operating system.

The hardware and the operating system work in tandem to provide a trusted path.

For example, to login to windows systems you have to press the key sequence CTRL-ALT-DEL. This sequence can't be trapped by any other program besides the operating system. Thus, you can be confident that you are communicating directly with the operating system after pressing this sequence.

In addition, the display and keyboard must be connected to the CPU on which the operating system is running, in a way in which there is no one in between. As I/O devices, they need to be using their own trusted path to the OS.

Sometimes, visual feedback can be provided to assure the user that they are using the trusted path, such as a certain type of display on a computer monitor or a light on a keyboard.

#### 4.12 Password Popularity Quiz



### Password Popularity Quiz

Check which passwords made the top 10 most common passwords for 2014:

<input type="checkbox"/>	123456	<input type="checkbox"/>	696969
<input type="checkbox"/>	password	<input type="checkbox"/>	123123
<input type="checkbox"/>	letmein	<input type="checkbox"/>	batman
<input type="checkbox"/>	abc123	<input type="checkbox"/>	qwerty
<input type="checkbox"/>	111111	<input type="checkbox"/>	123456789



### 4.13 Password Popularity Quiz Solution



**Password Popularity Quiz**  
Check which passwords made the top 10 most common passwords for 2014:

<input checked="" type="checkbox"/>	123456	<input type="checkbox"/>	696969
<input checked="" type="checkbox"/>	password	<input type="checkbox"/>	123123
<input type="checkbox"/>	letmein	<input type="checkbox"/>	batman
<input type="checkbox"/>	abc123	<input checked="" type="checkbox"/>	qwerty
<input type="checkbox"/>	111111	<input checked="" type="checkbox"/>	123456789



If we are attacking systems, we might get the best bang for our buck trying these passwords.

### 4.14 Implementing Password Authentication

When a user authenticates with a computer system, they must supply their identity and some verification of that identity. In password-based systems, this verification is the user's password.

When a user supplies their password to a computer system, the system then has the task of verifying that this is the correct password for this user.

How does the system know the user's password? One solution is to ask users to share their password with the system when they create their account. This is known as **enrollment**.

After enrollment, how does the system store the password?

#### 4.14.1 Method 1

The system can persist user passwords in plaintext in a system file. All the system has to do is compare the supplied password against the stored password.

As a basic security measure, the file should only be readable by the root/admin user, which is the user on whose behalf the login program runs.

What if permissions are set incorrectly? If so, someone else may be able to read that file and learn everyone's password.

Even if the permissions are set correctly, why should the admin know the passwords? If an attacker can impersonate the root user, they can learn all the passwords.

It's clear that something about the secret has to be shared with the system, but storing those secrets in a file as plaintext - even with access control - is not a good idea.

#### 4.14.2 Method 2

Alternatively, the system don't store the passwords themselves, but rather stores something derived from them.

We can pass the password through a **one-way hash function** to create this derived value.

Hash functions are often used to take an arbitrary length input and produce a fixed-size output that is fairly unique to the input.

A one-way hash function means that it is easy to compute the hash value given the input, but it is very difficult to recreate the input given the hashed value.

During enrollment, the system applies the hash function to the supplied password and writes this value to the file. When a user later comes to authenticate, the system hashes the supplied password and checks that value against the persisted hash value.

The benefit of this approach is that if an attacker steals this file they cannot compromise any user accounts because they can't recreate the original passwords.

Even so, these password files should still only be readable by the root user.

### 4.15 Hash Functions

A hash function takes a variable length password and outputs a fixed-length hash value.

A hash function is a one-way function, which means that it is very difficult to invert. In other words, it's easy to compute a hash given an input, but it's very difficult to recreate an input given a hash.

#### 4.15.1 Hash Functions and Threats

We assume that the one-way property of hash functions holds, so hash function inversion is not a threat.

One vulnerability present in authentication systems that use hash functions is users using common passwords.

Since the hash functions being used are well-known (remember security by obscurity is not a thing), an attacker can easily compute the hash of well-known password values.

If the attacker can get ahold of the system password file, they can hash common passwords and check the hashes against the file.

For example, if the password file contains the hash value ‘abcdef’ for Bob’s account, and the attacker determine that passing ‘123456’ through the hash function yields ‘abcdef’, then the attacker knows that Bob’s password is ‘123456’.

In a **dictionary attack**, a malicious user passes a large dictionary of common passwords - with some common mutations - to a series of hash functions in order to match against a stolen password file.

Dictionary attacks are examples of an **offline attack**, whereby the malicious user is not interacting directly with the system. When the user interacts with the system - a so-called online attack - the system can block subsequent login attempts after some number of failed attempts.

In offline attacks, the attacker has plenty of time to test a wide value of common passwords and hash functions against a stolen password file.

#### 4.16 Password Quiz



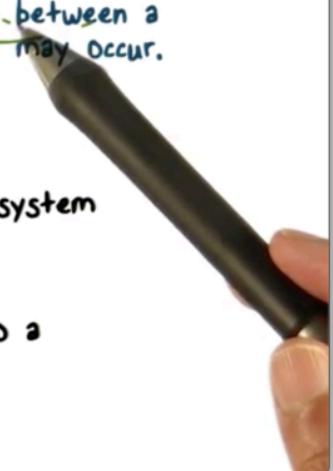
#### Password Quiz

If we do not have a trusted path between a user and the system, what problem may occur. Check the correct answer(s):

- User is not able to log into the system
- User may provide the password to a malicious program



#### 4.17 Password Quiz Solution



**Password Quiz**

If we do not have a trusted path between a user and the system, what problem may occur. Check the correct answer(s):

- User is not able to log into the system
- User may provide the password to a malicious program

A trusted path ensures that there is no application between the user and the operating system. Without this path, malicious programs may intercept login credentials.

#### 4.18 Hashed Passwords Quiz



#### Hashed Passwords Quiz

In the past, hashed passwords were stored in a publicly readable file /etc/passwd. Why were shadow password files added instead of making /etc/passwd file readable only to privileged users?

- Shadow files are more efficient to access
- There is other public information in /etc/passwd file that various utilities need



#### 4.19 Hashed Passwords Quiz Solution



#### Hashed Passwords Quiz

In the past, hashed passwords were stored in a publicly readable file /etc/passwd. Why were shadow password files added instead of making /etc/passwd file readable only to privileged users?



Shadow files are more efficient to access



There is other public information in /etc/passwd file that various utilities need



#### 4.20 Hash Function Characteristics Quiz



#### Hash Function Characteristics Quiz

The hash function used for computing hashed password values should meet the following requirements. Check the correct answer(s):



- Provide more efficient storage of password related information
- Produce different hashed values for distinct passwords
- Its inverse should be very hard to compute

## 4.21 Hash Function Characteristics Quiz Solution



### Hash Function Characteristics Quiz

The hash function used for computing hashed password values should meet the following requirements. Check the correct answer(s):

- Provide more efficient storage of password related information
- Produce different hashed values for distinct passwords
- Its inverse should be very hard to compute



## 4.22 Brute Force Guessing of Passwords

If an attacker has a system password file, how hard is it for them to brute force hash every single possible password?

There is publicly available software that can compute  $10^8$  MD5 hashes per second using a [graphical processing unit](#), a piece of hardware which is particularly well-suited for performed hashing calculations.

If the password is only six random alphanumeric characters (A-Z, a-z, 0-9), then the space of possible passwords is  $62^6$ . There are 62 ( $26 + 26 + 10$ ) possible characters for each position in the password, and there are 6 positions.

Using the above software, it will take  $62^6 / 10^8$  seconds - about ten minutes - to compute hashes for all possible passwords in that space.

If we increase the size of the password to eight random characters, the space grows from  $62^6$  to  $62^8$ , increasing the computation time from ten minutes to about twenty-five days.

Regardless, this is not an unfeasible amount of time for a hacker to wait while a program exhaustively searches a password space.

As a fun aside: increasing the password length to nine random characters increases the search time to 4.29 years, and increasing it to ten random characters increases the search time to 266 years.

### 4.23 Passwords are not Really Random

Passwords are not really random, so when an attacker is trying a brute force attack, they don't have to search for things in a completely random manner.

A smart attacker will try popular passwords first - for example, "password" and "123456" - in order to reduce the amount of work that they have to do.

An attacker can create a **rainbow table** mapping potential passwords to their hash values. With this table, the attacker doesn't even need to perform the hashing directly. Instead they can just lookup hash values from the password file and see what password maps to those hash values.

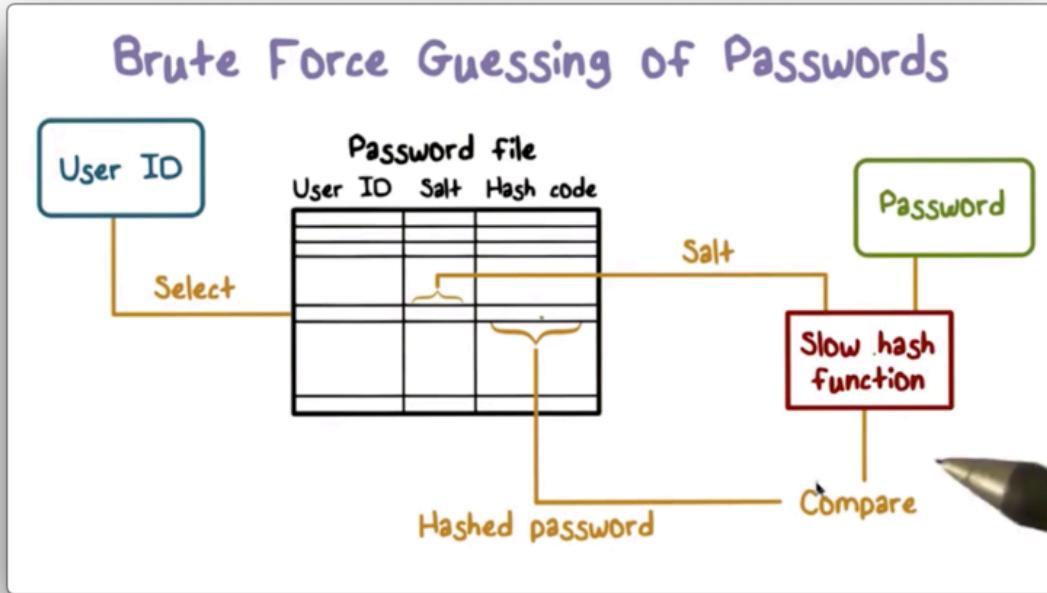
What if two users pick the same password? This is not an unlikely occurrence, given the popularity of a relatively small set of common passwords.

If multiple users use the same password, the hash values will be equal. This means that if an attacker compromises one of these accounts, they instantly compromise all of them.

To avoid this situation, we can add a **salt** - a random number - to each password before we hash it. Since the salt is randomly selected during enrollment, the salt and therefore the hash for users with the same password will be different.

Since the hashed value is derived from the salt and the password, the salt must be stored in the password file with the hashed value.

#### 4.24 Brute Force Figure



An enrolled user attempts to authenticate with the system by supplying a user id and a password.

The system indexes into the password file using the supplied user id in order to get the correct entry. This entry contains the user id, the salt, and the hash value.

The system then takes the supplied password, and the saved salt, and runs them both through the hash function. If the resulting hash value matches the hash value in the entry then access to the system is granted.

It is important to note that the hash function used by the system is intentionally slow. This is done to ensure that brute force attacks take much longer.

#### 4.25 Unique PINS Quiz



### Unique PINS Quiz

How many unique four digits PINs are possible? Check the correct answer:

- 1,000
- 100,000
- 10,000
- 1,000,000



#### 4.26 Unique PINS Quiz Solution



### Unique PINS Quiz

How many unique four digits PINs are possible? Check the correct answer:

- 1,000
- 100,000
- 10,000
- 1,000,000

With ten options for the first digit, ten options for the second digit, and so on, the total number of four digit pins is  $10 * 10 * 10 * 10$ , or  $10^4$ , or 10,000.

#### 4.27 Brute Force Quiz



#### Brute Force Quiz

A randomly chosen password has six characters that include upper and lower case letters, digits (0-9) and 10 special characters (examples are + ; etc.). In the worst case, how many attempts must a brute-force method make to determine a password when its hashed value is available?

Check the correct answer:

$6^{72}$

$62^6$

$72^6$



#### 4.28 Brute Force Quiz Solution



### Brute Force Quiz

A randomly chosen password has six characters that include upper and lower case letters, digits (0-9) and 10 special characters (examples are +, ; etc.). In the worst case, how many attempts must a brute-force method make to determine a password when its hashed value is available?

Check the correct answer:

$6^{72}$       $62^6$       $72^6$



With 72 options for each other six characters, the total number of unique passwords is  $72^6$ , which is the number of attempts the hacker will have to make in the very worst case.

#### 4.29 Touch Screen Password Quiz



#### Touch Screen Password Quiz

In smartphone touch screens, pattern based passwords are used to unlock the device. It is believed that such patterns are not random and there is a bias in where users start. This can be explained by...

Check the correct answer(s):

- Users often start at a random point but then fall back to a common pattern
- There is bias in starting at a point near the top left of the screen
- The ease of moving from current to next point introduces bias

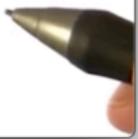
#### 4.30 Touch Screen Passwords Quiz Solution



### Touch Screen Password Quiz

In smartphone touch screens, pattern based passwords are used to unlock the device. It is believed that such patterns are not random and there is a bias in where users start. This can be explained by... Check the correct answer(s):

- Users often start at a random point but then fall back to a common pattern
- There is bias in starting at a point near the top left of the screen
- The ease of moving from current to next point introduces bias



Basically, the idea here is that attackers will likely not have to exhaustively search the space of possible patterns because biases exist that greatly shrink this space into a much smaller space of much more probable patterns.

#### 4.31 Problems with Passwords

Although we use passwords all the time, there are plenty of problems with passwords.

For passwords to be strong - that is, harder to crack - they need to be long and complex. This increases the size of the space that attackers need to search, but it also increases the complexity for users, at the expense of usability.

Even with strong passwords, there are problems that are related to the trusted path. When a system asks you to authenticate yourself, how do you know the system is not being impersonated? **Phishing** and **social engineering** attacks arise from users not authenticating the entities trying to authenticate them.

Unfortunately, once a password is stolen, it can be used many times because a password grants continual access. We can address this by having policies that enforce frequent password change, but again this hurts usability.

Finally, humans have a hard time remembering many different passwords for all of the different accounts they may have. As a result, people will often choose simple passwords that are easy to remember - and guess - and reuse these passwords across accounts.

#### **4.31.1 Best Practices for System Administrators**

Never store plaintext passwords. Instead, store only hashed values generated with a random salt and limit access to password file where the hash value is stored.

Use a slow hash function. The system has to execute this function only once when the user logs in, so speed is not crucial for this use case.

On the other hand, someone running a brute force attack will execute the function again and again, and a slow hash function will slow their attack down.

#### **4.31.2 Best Practices for Users**

Use a password manager to generate and maintain complex passwords for the services you use. Make sure to use a strong password for the password manager itself.

### **4.32 Something You Have Authentication**

Since passwords have problems, we can look at other authentication strategies.

We can authenticate users based on something they have - such as a smart card or a token - instead of using something they know, like a password.

How is authentication implemented with a smart card?

Often, the authentication flow will include a **challenge-response**. In challenge-response the system may send some sequence of characters to the token, and the token will send back a response based on this challenge. This response can be the same sequence encrypted with the token's private key, for example.

Thus, the system can authenticate the smart card and, by proxy, the user in possession of the smart card.

One obvious ramification of this approach is that the users must be in physical possession of the smart card in order to authenticate. If you leave your smart card at home, you'll have to go back and get it.

In addition, a smart card needs special hardware - such as a card reader - installed in order to communicate with the system. This added requirement adds cost to the implementation of this type of authentication system.

Another problem with this approach is misplaced trust. For example, the [RSA SecureID system breach](#) created a situation in which tokens could potentially be forged by attackers.

### 4.33 Something You Are Authentication

Finally, authentication can be based on **biometrics** or, more colloquially, “something you are”.

For example, a system can look at the pattern of ridges and grooves in a supplied fingerprint to verify a user’s identity.

Alternatively, a system can analyze keystroke dynamics, such as how fast a user types or how long it takes a user to transition from certain keys to others. This data can be synthesized to create a representation of a specific user.

(To be fair, keystroke dynamics fall more into the “something you do” category, as opposed to the “something you are” category. Such behavioral identifiers are sometimes referred to as *behaviometrics*.)

Some systems can analyze the sound and speed of your speech to authenticate you by your voice. Other systems can verify your identity by scanning your retina and measuring the underlying blood flow.

There are many different choices for biometrics, but they should all be unique to an individual. A feature that many people share - such as height in inches - is not a sufficient biometric for the purpose of authentication.

In addition, the biometric recording should be the same during each authentication attempt. Otherwise, the system may deny you access: a false negative.

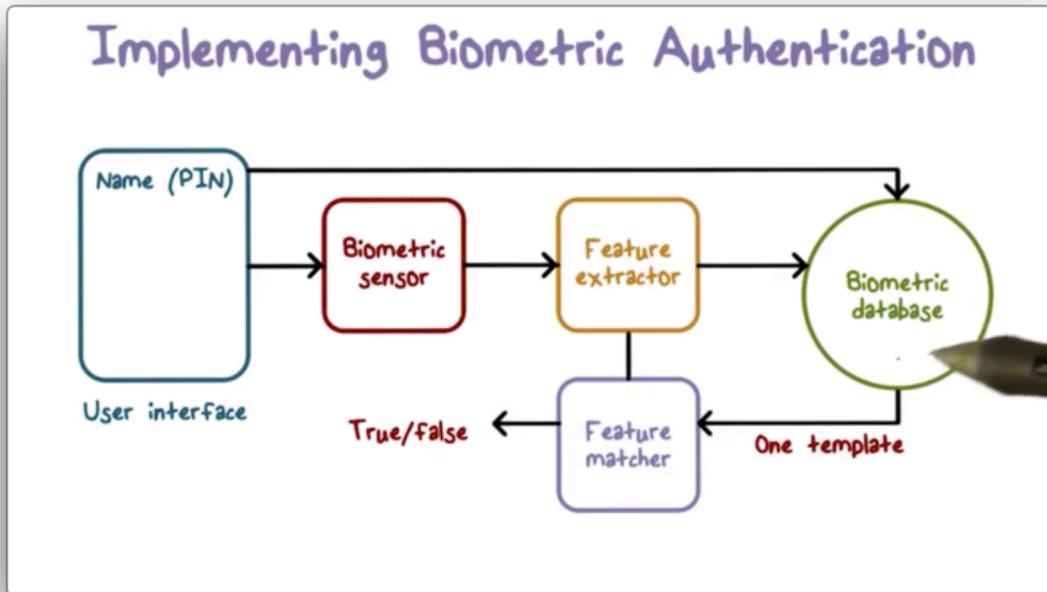
Of course, this consistency is not always possible. For example, a cold can impact your voice and moisture or swelling can obscure your fingerprint.

Instead of looking for exact measurements when using biometrics, a system might use a probability distribution when verifying identity.

The system will have to pick a certain threshold of similarity by which the false negative rate and the false positive rate are balanced appropriately.

False negatives - denying valid users access - hurts usability while false positives - allowing Alice to authenticate as Bob - is insecure.

#### 4.33.1 Implementing Biometric Authentication



After a user supplies their identity to a system, the biometric sensors capture the biometric data of interest.

The system then digitizes this data and extracts certain key features to be used for matching.

The extracted features - or some data derived from them - are matched against the stored data for the given user and, if there is a match, the system grants access to the user.

#### 4.34 Other Methods

##### 4.34.1 Multi-factor Authentication

We can combine different **factors** (something you know, something you have, something you are) into a **multi-factor authentication** strategy.

For example, some websites may require you to log-in with your password and then ask you to enter a code that they text to you. The password is something you know, and the code is something you know based on something you have.

Another example is the ATM card and PIN combination required to withdraw money from the bank. The card is something you have and the PIN is something you know.

Other authentication methods can incorporate other factors. For example, a system can use a location-based factor - “somewhere you are” - by examining IP addresses in network requests.

Multi-factor authentication makes it harder for attackers to impersonate users.

#### 4.34.2 Authentication over a Network

When we are authenticating with services over the open Internet, we no longer have the trusted path to the OS authenticating us.

Cryptography helps secure network communication through encryption and shared secrets. Of course, this doesn’t help against certain *man-in-the-middle* attacks.

#### 4.35 Multi-factor Authentication Quiz



### Multi-factor Authentication Quiz

A multi-factor authentication method will likely reduce false positives. Choose one:

True

False



#### 4.36 Multi-factor Authentication Quiz Solution



## Multi-factor Authentication Quiz

A multi-factor authentication method will likely reduce false positives. Choose one:

True  
 False

Remember, a false positive occurs when a malicious user is granted access to the system as a valid user. The likelihood of this happening decreases when multiple authentication components are employed by the system.

#### 4.37 Chip and Pin Authentication Quiz



#### Chip and Pin Authentication Quiz

Although a "something you have" based authentication method avoids problems associated with passwords, it could also be prone to attacks. For example, read about chip and pin based authentication at the link listed in the instructor notes.

What is the main weakness that is illustrated here?

- Lost cards
- Cloning of cards
- Vulnerabilities in implementation



#### 4.38 Chip and Pin Authentication Quiz Solution



#### Chip and Pin Authentication Quiz

Although a "something you have" based authentication method avoids problems associated with passwords, it could also be prone to attacks. For example, read about chip and pin based authentication at the link listed in the instructor notes.

What is the main weakness that is illustrated here?

- Lost cards
- Cloning of cards
- Vulnerabilities in implementation

Read more [here](#) and [here](#).

#### 4.39 Biometric Authentication Quiz



#### Biometric Authentication Quiz

Biometric authentication based on fingerprints can be hacked if an attacker can gain access to a user's fingerprint.

For example, it has been demonstrated that the Apple's Touch ID can be fooled with lifted fingerprints. See the link in the instructor's note.

Can a similar attack be mounted if voice biometric authentication is used?

Yes       No



#### 4.40 Biometric Authentication Quiz Solution

 **Biometric Authentication Quiz**

Biometric authentication based on fingerprints can be hacked if an attacker can gain access to a user's fingerprint.

For example, it has been demonstrated that the Apple's Touch ID can be fooled with lifted fingerprints. See the link in the instructor's note.

Can a similar attack be mounted if voice biometric authentication is used?

Yes     No



As a basic example, consider someone recording your voice and playing it back to a voice-based authentication system.

## 5 Access Control

### 5.1 Controlling Accesses to Resources

Any program can make a request for a protected resource, so the OS needs to know whether to grant or deny these requests.

For example, if John is a student in a class, and he makes a request to read a file containing the grades for all of the students in that class, should his request be granted?

Remember, authentication tells us the user on whose behalf a process or application makes a request. Authentication allows us to know that the above request came from John.

Authorization, or **access control**, involves determining if a certain requestor is allowed to access the resources they are requesting in the manner in which they are requesting. Access control determines whether John can access the file for reading or not.

Resources - like files - that we have in the system are created by certain users, or **subjects**.

One implementation of access control says that the creator of a resource has full reign over how that resource is accessed.

Indeed, in many systems the idea of a resource owner is defined, and it can be at the discretion of the owner how that resource can be shared.

Of course, there are situations where this is not true. For example, your company may not allow you to decide how you wish to share information related to that company, regardless of whether you created the information or not.

## 5.2 Access and Authentication

In order to understand whether a resource access request should be granted or denied, we need to know who is allowed to access which resources in a given system.

There are two components to the problem of access control.

First, system administrators need to specify who has access to what. This specification is called an **access control policy**.

Once the administrators have delivered this policy to the system, the system needs to monitor each request and ensure that any accesses it grants are consistent with the policy.

Complete mediation is essential for access control to work. There is no point having a system enforce policies if a user can step around the system.

## 5.3 Access Control Matrix Defined

We can abstract the information about “who has access to what” into a data structure called an **access control matrix** (ACM).

Like any matrix, an ACM has rows and columns. The rows of an ACM correspond to the users, subjects or groups present in the system. The columns of an ACM correspond to the system resources that need to be protected.

Each cell in the ACM, indexable by user and resource, contains a set of operations that the queried user can perform on the queried resource.

For example if we have an ACM  $A$ , a user  $U$ , and a resource  $R$ , then  $A[U, R]$  contains the list of operations that  $U$  can perform on  $R$ .

## 5.4 Access Control Matrix

Here is an example of an  $m \times n$  matrix, with  $m$  rows and  $n$  columns.

**Implementing Access Control**

List all processes and subjects in a matrix

A <sub>11</sub>	A <sub>12</sub>	A <sub>13</sub>	...	A <sub>1n</sub>
A <sub>21</sub>	A <sub>22</sub>	A <sub>23</sub>	...	A <sub>2n</sub>
A <sub>31</sub>	A <sub>32</sub>	A <sub>33</sub>	...	A <sub>3n</sub>
.	.	.	.	.
.	.	.	.	.
A <sub>m1</sub>	A <sub>m2</sub>	A <sub>m3</sub>	...	A <sub>mn</sub>

Size  
 $m \times n$



The first row in this matrix corresponds to the first user, the second row to the second user, and so on until the  $m$ -th user in the  $m$ -th row.

Likewise, the first column in this matrix corresponds to the first object, the second column to the second object, and so on until the  $n$ -th object in the  $n$ -th column.

Each cell in this matrix contains the authorized operations that a user can perform on an object. For example, the cell at  $A[3, 2]$  contains the operations that user three can perform on object two.

The operations present will be some subset of the possible operations that can be performed on the given resource. In the case of a file  $F$  - which can be readable, writeable, and/or executable - and user  $U$ ,  $A[U, F]$  will have some combination of read, write, and execute bits present.

## 5.5 Data Confidentiality Quiz



### Data Confidentiality Quiz

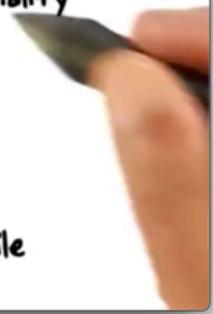
Select the best answer to complete this sentence:

A file is created by a certain user who becomes its owner. The owner can choose to provide access to this file to other users. If file data confidentiality is desired, the owner should control who has...

Read access to the file

Write access to the file

Both read and write access to the file



## 5.6 Data Confidentiality Quiz Solution



### Data Confidentiality Quiz

Select the best answer to complete this sentence:

A file is created by a certain user who becomes its owner. The owner can choose to provide access to this file to other users. If file data confidentiality is desired, the owner should control who has...



Read access to the file



Write access to the file



Both read and write access to the file



Controlling read access is connected to data confidentiality, while controlling write access is connected to data integrity.

## 5.7 Determining Access Quiz



### Determining Access Quiz

Select the best answer to the question:

The access control policy in a system can either define positive access for a certain subject or can specify that the subject be denied access. Consider a case where subject Alice belongs to a group All-Students. The system specifies that members of the group All-Students be able to read file foo but Alice is denied access for it. In such a case, what should the system do?

- Alice has access because she is member of All-Students so she must be allowed to read foo
- Negative access should take precedence and Alice's request must be denied

+read

read

## 5.8 Determining Access Quiz Solution



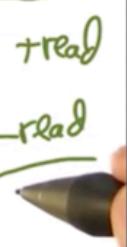
### Determining Access Quiz

Select the best answer to the question:

The access control policy in a system can either define positive access for a certain subject or can specify that the subject be denied access. Consider a case where subject Alice belongs to a group All-Students. The system specifies that members of the group All-Students be able to read file foo but Alice is denied access for it. In such a case, what should the system do?

Alice has access because she is member of All-Students so she must be allowed to read foo *+read*

Negative access should take precedence and Alice's request must be denied *-read*



Access control conflicts can be securely resolved by denying access.

## 5.9 Discretionary Access Control Quiz



### Discretionary Access Control Quiz (DAC)

In discretionary access control, access to a resource is at the discretion of its owner. Let us assume owner Alice of file foo can choose to grant read access to foo to another user Bob but can prevent Bob from propagating this access right to others. Does this ensure that a third user, Charlie, can never read the data from foo?

- Yes, Charlie is not granted access so cannot read
- No, there may be another way for Charlie to access the data from foo

## 5.10 Discretionary Access Control Quiz Solution



### Discretionary Access Control Quiz (DAC)

In discretionary access control, access to a resource is at the discretion of its owner. Let us assume owner Alice of file foo can choose to grant read access to foo to another user, Bob but can prevent Bob from propagating this access right to others. Does this ensure that a third user, Charlie, can never read the data from foo?

Yes, Charlie is not granted access so cannot read

No, there may be another way for Charlie to access the data from foo

Bob can write the contents of the file to a new file that he owns, and share that file with Charlie.

## 5.11 Implementing Access Control

A system will likely have multiple users and many different objects that it needs to protect; therefore, an access control matrix can be very large.

This matrix will also be relatively *sparse*. With many users in a system, and many objects that any individual user cannot access, the majority of the cells in the matrix will be empty.

How should we represent the access control matrix within our system?

If we focus on each column of the matrix, we can make a list of all of the access rights users have for a given object.

Consider an object  $O$ . If user  $A$  has read access to  $O$  and user  $B$  has write access to  $O$ , and user  $C$  has no access to  $O$ , we can represent the accesses permitted on  $O$  as

1  $[(A, r), (B, w)]$

This is called an **access control list** (ACL). ACLs are associated with each resource, and enumerate the authorized users and types of operations permitted on that resource.

Alternatively, we can focus on each row in the matrix, which corresponds to a particular user.

Consider a user  $U$ . If  $U$  can read object  $X$ , and can write object  $Y$ , but cannot access object  $Z$ , we can represent the authorized operations that  $U$  can perform as

```
1 [(X, r), (Y, w)]
```

This type of list is called a **capability list** (C-list), and identifies which objects a user can access and in what fashion.

## 5.12 Example Access Control Matrix

Implementing Access Control			
	X	Y	Z
A	rwx	r	
B		rw	rx
C		rw	rx

**ACLs**

$$X \rightarrow [(A, rwx)]$$

$$Y \rightarrow [(A, r)(B, rw)(C, rw)]$$

$$Z \rightarrow [(B, rx)(C, rx)]$$

**C-lists**

$$A \rightarrow [(X, rwx)(Y, r)]$$

$$B \rightarrow [(Y, rw)(Z, rx)]$$

$$C \rightarrow [(Y, rw)(Z, rx)]$$

In this example, we have a system with users A, B, and C, and objects X, Y, and Z.

We can create ACLs by looking “down” the column for each object, and we can create C-lists by looking “across” the row for each user.

## 5.13 ACL Implementation

Since an ACL determines who can access a resource that needs to be protected, ACLs must be stored in the kernel. Otherwise, any untrusted application could change access rules arbitrarily.

Within the kernel, one natural place to store an ACL for a resource is the same place that we store other metadata for that resource.

For example, a file has a bunch of metadata associated with block locations and file size and so forth, so storing the ACL with this metadata seems logical.

When a request for an object occurs, the operating system must traverse the ACL to determine if an **access control entry** (ACE) is present for the requesting user and, if so, whether the requested action is permitted.

For example, if user [A](#) is trying to read file [F](#), the operating system must retrieve the ACL for [F](#), find the ACE for [A](#) and check to see whether the “read” operation is present in the list of authorized operations.

## 5.14 C Lists Implementation

A capability is an unforgeable handle for a resource, which a user can present to the system to gain access to the resource. A capability is essentially a token that proves that a user has been given permission to access a resource.

Since capabilities have this unforgeable requirement, they must be stored in the kernel. Unlike ACLs, they are not going to be stored with resources themselves, since they are not scoped to individual objects.

Instead, the kernel must maintain a user catalogue of capabilities that defines what any particular user can access.

In some cases, a C-list can be stored in the objects or resources themselves, as was in the case in the [Hydra](#) operating system from Carnegie Mellon, but this is less common.

How can authorizations be shared?

In the case of ACLs, a user who has permission to grant access to a resource can make a system call and authorize that access for another user. The operating system will create a new ACE for the newly authorized user and add it to the ACL for the resource.

For C-lists, sharing requires propagation of capabilities from one user to another.

Possession of a capability means that you can access the resource. There is no system-level access check as is the case with ACLs.

## 5.15 ACL and C Lists Implementation: ACL vs C list

If we are developing a new operating system, should we use ACLs or C-lists? What are the pros and cons of choosing one over the other?

### 5.15.1 Efficiency

With ACLs, the system needs to traverse the ACL in order to find an ACE corresponding to the user who made the request. There is some processing overhead associated with this traversal.

With capability lists, the mere presentation of a capability is sufficient to access a resource. The system does not have to perform any further checks.

In this manner, C-lists are more efficient than ACLs.

### 5.15.2 Accountability

If you want to know all of the users that have access to a particular resource, you can do this really easily with an ACL since, by definition, it contains the list of users and authorized actions for a given resource.

To determine the same information using C-lists, the system will need to look in every user catalogue and extract the authorizations for the resource in question.

Because of this, accountability is more transparent with ACLs.

### 5.15.3 Revocation

Revocation refers to removing access to a given object for a given user after some initial access has been granted.

Alice can easily revoke Bob's access to file `foo` if ACLs are used. She can simply make a system call, and the system will locate the ACL for `foo`, find the ACE corresponding to Bob, and remove the appropriate permission.

For C-lists, Alice can't remove permissions from Bob's catalogue at will.

Revocation is easier to perform using ACLs.

### 5.16 ACE Quiz



**ACE Quiz**

Select the best answer:

Alice goes to a movie theater and purchases a ticket for her favorite movie. She is allowed access to the movie because she has the ticket. The ticket is more like a...

Access control entry

Capability



### 5.17 ACE Quiz Solution



**ACE Quiz**  
Select the best answer:

Alice goes to a movie theater and purchases a ticket for her favorite movie. She is allowed access to the movie because she has the ticket. The ticket is more like a...

Access control entry  
 Capability



The presentation of the ticket is sufficient to gain access to the theater. No other access checks are required. This is closest in functionality to a capability.

### 5.18 ACE Access Quiz

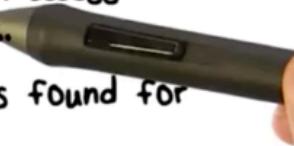


#### ACE Access Quiz

Select the best answer:

Some operating systems (e.g., Windows) include deny or negative access rights. In this case, an access check procedure can terminate as soon as...

- A positive or grant access ACE is found for the requestor
- A negative or deny ACE is found
- The whole ACL must be traversed always



### 5.19 ACE Access Quiz Solution



#### ACE Access Quiz

Select the best answer:

Some operating systems (e.g., Windows) include deny or negative access rights. In this case, an access check procedure can terminate as soon as...

- A positive or grant access ACE is found for the requestor
- A negative or deny ACE is found
- The whole ACL must be traversed always

Negative access rights supersede positive access rights, so you can't terminate as soon as you find a positive access right. You can terminate as soon as you find a negative access right, though.

NB: The third option can't be true if the second option is true.

## 5.20 Revocation of Rights Quiz



### Revocation of Rights Quiz

Select the best answer:

Revocation of certain access rights can be carried out easily in systems that use...

- ACLs
- C-lists



### 5.21 Revocation of Rights Quiz Solution

The slide contains the following text and graphics:

 Revocation of Rights Quiz  
Select the best answer:  
Revocation of certain access rights can be carried out easily in systems that use...  
 ACLs  
 C-lists

### 5.22 Access Control Implementation Part 1

In Unix, every resource that needs protection is represented as a file.

All users in Unix systems have a unique user id (UID). Users can belong to groups, which each have their own group id (GID). There is a special *world* group which contains every user in the system.

Since resources are files, they can be read, written or executed.

In the access control matrix, each row corresponds to each entity in the system: each UID, each GID, and the special world group. Each column corresponds to each file in the system. Each entry will be a subset of read/write/execute.

The original ACL implementation in Unix systems had a compact, fixed size representation representing the read, write, and execute permissions for the user, the group, and the world.

These three ACEs were represented as a nine bit mask such as 111110100. You may have seen this mask in a slightly different format: `rwXrwx-r--` if you have ever run the `ls -l` command on your Unix machine.

This ACL structure is limited in its flexibility since it provides only three levels of authorization granularity.

Modern operating systems - Linux, BSD, MacOS - provide full ACL support in order to make access control as flexible as necessary.

### 5.23 Access Control Implementation SetUID

Consider the following scenario.

Alice owns an executable file that launches a computer game. Alice also owns a text file that contains the scores of the users playing the game.

Alice wants to allow any user to execute the game file, but wants to restrict write access to the score file unless the game is being played.

We can solve this problem with the **setuid** bit. If this bit is set on an executable file, a process executing this file takes on the *effective user id* of the owner of the file - not the current user - for the duration of the execution.

In other words, when Bob runs Alice's game, the process that is created by Bob has Alice's UID associated with it. This means that Bob can access the score file - which Alice owns - while he plays the game so he can record his scores.

Since the effective UID is set on a per-process basis, this temporary privilege will go away once Bob closes the game.

### 5.24 Access Control Implementation Part 2

A file can be opened with the `open` system call. If the file doesn't exist, this call will create the file and then open it.

The `open` system call takes two arguments: the name of the file and the mode in which you want to access the file. The mode can be read or write, for example.

Once we open a file, the operating system returns a *file descriptor*, which is a small integer. In order to read, write or close a file, we need to pass this descriptor back to the operating system.

If you want to read a file, you can pass the file descriptor to the `read` system call. The call to `read` takes two arguments: the buffer into which you wish to read the data, and the number of bytes you wish to read.

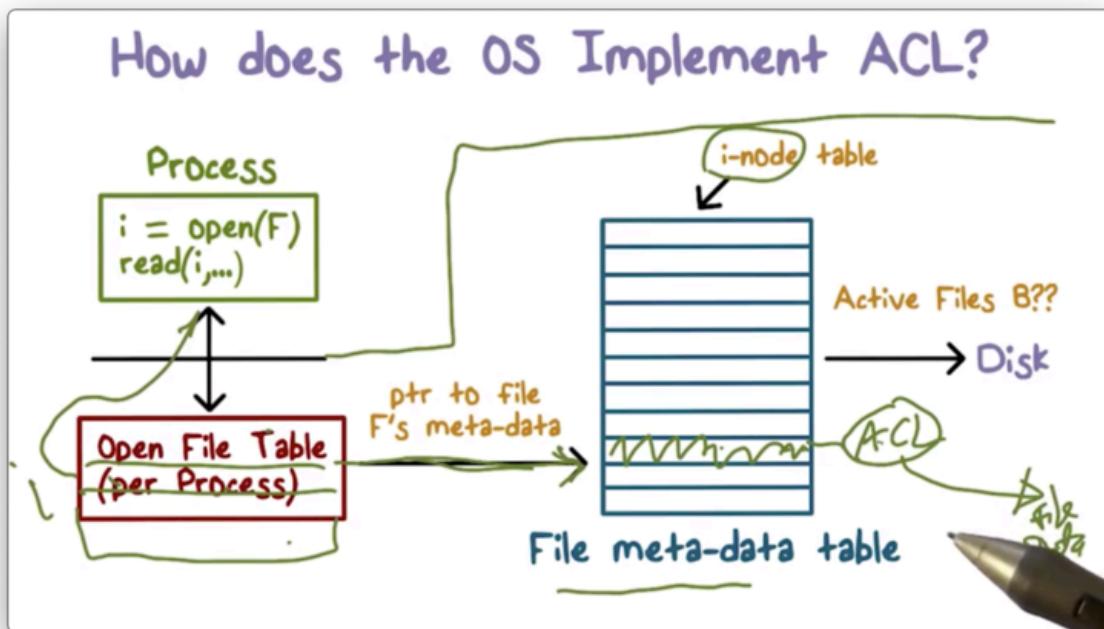
Similarly, there is a `write` system call, which writes a supplied number of bytes from a supplied buffer back into the file referenced by the file descriptor.

When you are done with the file, you can close it with the `close` system call, which also takes the file descriptor as an argument.

Read more about `open`, `read`, `write`, and `close`.

## 5.25 How does the OS Implement ACL?

What happens when you open a file?



When we execute the `open` system call, the very first thing that happens is that we switch into kernel mode and start executing from within the operating system.

Every file has metadata associated with it. This metadata can contain disk block locations, file size, file name, and the ACL.

The data structure that contains this metadata is referred to as an `inode` on Unix systems, and the operating system maintains a table of inodes for all of the files in the system that are actively being accessed.

When we attempt to open a file, the operating system will check the inode table to see if there is an entry for the given file. If not, the operating system will create an entry for that file.

Once the inode for the file is located, access control is performed. The system will locate the ACL and will decide whether to grant access or not based on the bits present in the ACL and the mode in which

the user is requesting to open the file.

If the system determines that access should not be granted, the requesting process is not given with a file descriptor.

If the system determines that access should be granted, it will create an entry in another table, the per-process *file descriptor table*. This table holds pointers to the inode table and contains all of the files currently being accessed by the given file.

The file descriptor returned from this call is nothing more than an index into that table.

What is important to understand here is that after the file descriptor is returned, there are no subsequent access checks. If your permissions to read a file are removed after you have opened a file for reading, you can continue reading using that file descriptor.

Of course, if a process opens a file for reading and then tries to write to the file, the system will prevent it.

### 5.26 Time to Check vs Time to Use Quiz

**Time to Check v.s. Time to Use (TOCTOU) Quiz**

A time-to-check-time-to-use vulnerability arises when access check is performed separately from when a file is read or written. TOCTOU vulnerability arises when...

- File permissions change after an `open()` call completes for the file and before it is closed.
- The file permission change only when the file is currently not opened by any program

### 5.27 Time to Check vs Time to Use Quiz Solution



#### Time to Check vs. Time to Use (TOCTOU) Quiz

A time-to-check-time-to-use vulnerability arises when access check is performed separately from when a file is read or written. TOCTOU vulnerability arises when...



File permissions change after an `open()` call completes for the file and before it is closed.



The file permission change only when the file is currently not opened by any program



As long as you had the permissions when you called `open`, you can access the file using the file descriptor.

### 5.28 Unix File Sharing Quiz



#### Unix File Sharing Quiz

In Unix based systems, a file can be shared by sharing its descriptor.

True

False



### 5.29 Unix File Sharing Quiz Solution



### Unix File Sharing Quiz

In Unix based systems, a file can be shared by sharing its descriptor.

True

False

You would need to somehow add the descriptor to the per-process descriptor table for the process with which you wish to share the descriptor. Since the OS owns this table, mutating it is impossible.

### 5.30 SetUID Bit Quiz



#### SetUID Bit Quiz

An executable file F1 has the setuid bit set and is owned by user U1. When user U2 executes F1 (assuming U2 has execute permission for F1), the UID of the process executing F1 is...

- U1
- U2



### 5.31 SetUID Bit Quiz Solution

**SetUID Bit Quiz**

An executable file F1 has the **setuid** bit set and is owned by user U1. When user U2 executes F1 (assuming U2 has execute permission for F1), the **UID of the process executing F1 is...**

U1  
 U2

The effective UID of a process executing a file with the setuid bit set is the owner of the file, not the user who created the process.

### 5.32 Role Based Access Control

Some systems use **role-based access control** (RBAC). In RBAC, the rights to access certain resources are associated with roles instead of users.

Once a user authenticates with the system, they can take on one or more roles that allow them to interact with the system in certain ways.

A role is often closely linked to job function. For example, people in payroll might assume a role that allows them to access salary data, while engineers might have a role that allows them to deploy code.

A user can have more than one role. The employees described above might take on a general employee role which allows them to connect to a local intranet or access a company directory.

[Security-Enhanced Linux](#) (SELinux) supports RBAC.

### 5.32.1 RBAC Benefits

Since the policy defines which roles have access to which resources, the policy doesn't need to change when any one user leaves the organization.

In addition, a new employee can easily be set up. Instead of having to assign them individual privileges one by one, a system administrator can assign them to one or more roles and instantly grant the appropriate amount of access.

In these regards, RBAC makes policy administration simpler.

### 5.32.2 Least Privilege

Remember, *least privilege* is the idea that a user should always execute with the smallest number of privileges or access rights needed to complete a task.

The point of least privilege is really damage containment. If something goes wrong, a user can't damage resources that they had no business accessing in the first place.

In RBAC, a user can start in one role and access a subset of the files that are only available to that role. At a later point, the user can switch roles and access a different set of files associated with the new role.

Without roles, a user just has static access to the complete universe of things they would ever need to access. Roles allow for much tighter access control scoped by functionality.

### 5.33 RBAC Benefits Quiz



#### RBAC Benefits Quiz

In systems that do not support RBAC but allow user groups to be defined, benefits of RBAC can be realized with groups.

True

False



### 5.34 RBAC Benefits Quiz Solution



#### RBAC Benefits Quiz

In systems that do not support RBAC but allow user groups to be defined, benefits of RBAC can be realized with groups.

True

False



### 5.35 Access Control Policy Quiz



#### Access Control Policy Quiz

Fail-safe defaults implies that when an access control policy is silent about access to a certain user U...



- Access must be denied when U makes a request
- Access can be granted because it is not explicitly denied

### 5.36 Access Control Policy Quiz Solution



### Access Control Policy Quiz

Fail-safe defaults implies that when an access control policy is silent about access to a certain user U...

Access must be denied when U makes a request

Access can be granted because it is not explicitly denied



From a security standpoint, denying access is a fail-safe default. It never fails to keep your system secure.

## 6 Mandatory Access Control

### 6.1 Discretionary Access Control

In the last lesson we talked about access control.

In particular, we focused on **discretionary access control** (DAC), whereby the user who creates a resource is the owner of that resource and can choose to give access to other users.

### 6.2 Two Problems with DAC

To illustrate the first problem with DAC, let's consider the following scenario.

Alice owns a file and gives read access to Bob, but not to Charlie. Bob copies the contents into a new file, and shares that file with Charlie. Charlie now has effective access to the file that Alice didn't want to share with him in the first place.

While DAC allows users to control direct access to a file, we can't actually prevent the spread of data contained in that file; that is, we can't control the *information flow*.

Another problem with DAC is that it doesn't reflect how most organizations treat their data. Most employers do not leave the decisions about data sharing in the hands of their employees, but rather mandate explicit policies about who can share what.

To address these problems, we will explore another model for access control: **mandatory access control** (MAC). In mandatory access control, decisions about sharing information are not made at the discretion of the user.

### 6.3 DAC Quiz



**DAC Quiz**  
Check the best answer:

In a certain company, payroll data is sensitive. A file that stores payroll data is created by a certain user who is an employee of the company. Access to this file should be controlled with a...

DAC policy that allows the user to share it with others judiciously

It must use a MAC model as the company must decide who can access it



## 6.4 DAC Quiz Solution

**DAC Quiz**  
Check the best answer:

In a certain company, payroll data is sensitive. A file that stores payroll data is created by a certain user who is an employee of the company. Access to this file should be controlled with a...

DAC policy that allows the user to share it with others judiciously

It must use a MAC model as the company must decide who can access it

DAC can't control information flow, so we must use MAC.

## 6.5 Mandatory Access Control (MAC) Models

Hospitals are a great example of an entity that needs to employ mandatory access control.

Hospitals store and process electronic health records (EHRs), which contain medical information about patients. We regard health information as highly sensitive and, as such, seek to limit who can access and/or share this information.

In addition, there are regulatory requirements that limit how medical information can be shared and control how patient records can be accessed. [HIPAA](#) is the legislation in the United States that dictates how records can be accessed and shared.

Of course, there is a need to share your medical information - if you see a new doctor, for instance.

The point is that these access control decisions are not dictated by the individual performing data entry on the patient's record, but rather by the hospital and the government.

## 6.6 Implementing MAC

In mandatory access control, users and resources/documents will have certain labels associated with them.

A **label** is simply an identifier that will tell us how sensitive certain information is, or how privileged a certain user is. Labels can also contain categories, which can be used to group users or documents.

While the TCB does not assign these labels, it uses labels to determine whether a given user can access a given resource.

In order to grant access decisions, we have to be able to compare labels. Labels must have an order so that the system can make consistent decisions.

For example, if `A > B > C`, the system will deny access to a document with label `A` to a user with label `B`, but will grant that user access to a document with label `C`.

The exact nature of what the labels look like, how we compare them, and the result of the comparison depends on the particular policy being implemented.

In the Department of Defense, the labels will include a clearance level for users and a classification level for documents, as well as a compartment. The compartment is used to describe what kind of information a document contains and what categories of information a user can access.

In the commercial world, we don't have notions of clearance and classification, so the labels will look different.

The concerns are different as well. For example, we may have a policy that prevents conflict of interest. A user may not be able to access resources from which they stand to enjoy a material gain.

## 6.7 Implementing MAC Example

In the Department of Defense, documents have classifications and users have clearances.

The sensitivity levels present in the DoD system, in order from most sensitive to least sensitive, are: Top Secret, Secret, Confidential, Restricted, and Unclassified.

A label contains a sensitivity level and a compartment that describes what kind of data is contained in the document.

Let's consider two different documents, which contain information about various arms stockpiles.

One document may have the label (`TS, {nuclear, chemical}`), identifying it as a top secret document concerned with nuclear and chemical weapons.

The other document may have the label (`S, {nuclear, conventional}`), identifying it as a secret document containing information about nuclear and conventional weapons.

A document may pertain to multiple topics, so the compartment can have multiple entries.

The [Bell-LaPadula Model](#) (BLP) makes use of labels and provides a set of rules that determines when a document can be read or written.

## 6.8 Health Data Quiz

**Health Data Quiz**  
Check the best answer:

A hospital is found to be lax in securing access to patient records after it suffers a major breach. It may have violated the following policy:

HIPAA  
 BLP

## 6.9 Health Data Quiz Solution



**Health Data Quiz**  
Check the best answer:

A hospital is found to be lax in securing access to patient records after it suffers a major breach. It may have violated the following policy:

HIPAA  
 BLP



BLP is concerned with military/governmental intelligence. HIPAA is concerned with health information.

### 6.10 Security Clearance Quiz



#### Security Clearance Quiz

Check the best answer:

Highly sensitive defense or intelligence information  
should only be accessed by cleared personnel.  
Approximately, how many people in the United States  
have various types of clearances?

10,000

100,000

1,000,000

5,000,000



## 6.11 Security Clearance Quiz Solution



### Security Clearance Quiz

Check the best answer:

Highly sensitive defense or intelligence information should only be accessed by cleared personnel.

Approximately, how many people in the United States have various types of clearances?

10,000       1,000,000  
 100,000       5,000,000



Source: [Washington Post](#)

## 6.12 Comparing Labels

In order to fully implement access control using labels, we need some strategy for comparing labels.

Sensitivity levels in the DoD scheme are ordered as follows:

1 TS > S > C > U

This ordering is *total*, which means that if you pick any two sensitivity levels, you can always determine which level is more sensitive than the other.

Labels also have compartments, which comprise sets of different topics.

Two sets  $S_1$  and  $S_2$  can be oriented such that

- $S_1$  is contained in  $S_2$
- $S_2$  is contained in  $S_1$
- Neither set contains the other

Sets are compared by containment. A set that contains another set is “greater” than that set. Note that this ordering is *partial*, because two sets that don’t contain each other cannot be ordered.

When we want to order labels, we must look at both the sensitivity level and the compartment

### 6.13 Comparing Labels Example

Based on the comparison rules just described, the following chart depicts the circumstances under which

- one label dominates another
- two labels are equal
- two labels are not comparable

### Comparing Labels

$$L_1 = (l_1, \text{Comp}_1), L_2 = (l_2, \text{Comp}_2)$$

$L_1$  dominates  $L_2$  :  $l_1 > l_2$  and  $\text{Comp}_1 \geq \text{Comp}_2$

or  $L_1$  is dominated by  $L_2$  :  $l_1 < l_2$  and  $\text{Comp}_1 \leq \text{Comp}_2$

or  $L_1 = L_2$  :  $l_1 = l_2$  and  $\text{Comp}_1 = \text{Comp}_2$

or  $L_1$  and  $L_2$  are not comparable :  $L_1 \not\geq L_2$  and  $L_1 \not\leq L_2$   
and  $L_1 \neq L_2$

### 6.14 Ordering Among Labels

Let’s try to compare the following labels.

```
1 L1 = (TS, {A,B,C})  
2 L2 = (S, {A, B})  
3 L3 = (S, {B, C, D})
```

L<sub>1</sub> dominates L<sub>2</sub> because TS > S, and {A, B, C} > {A, B}.

L<sub>2</sub> cannot be compared with L<sub>3</sub> because while S == S, {A, B} cannot be compared to {B, C, D}

Similarly, L<sub>1</sub> and L<sub>3</sub> cannot be compared. While TS > S, {A, B, C} cannot be compared with {B, C, D}.

### 6.15 Order Quiz

The slide features a decorative icon of two crossed swords with green and purple gemstones at their hilt. To the right of the icon, the text "Order Quiz" is written in a purple, handwritten-style font. Below it, the instruction "Select the best answer:" is written in a teal, handwritten-style font. The main question is: "The '<' relation among all real numbers defines a...". Two answer options are provided, each preceded by a square checkbox:

- Total order
- Partial order

A photograph of a person's hand holding a black pen, pointing towards the bottom right corner of the slide. The pen is angled diagonally, with the tip pointing towards the bottom right.

### 6.16 Order Quiz Solution



**Order Quiz**  
Select the best answer:

The " $<$ " relation among all real numbers defines a...

Total order  
 Partial order



Given any two real numbers, one number is always greater than the other.

### 6.17 Label Domination Quiz



#### Label Domination Quiz

Select the best answer:

If  $L_1 = (\text{secret}, \{\text{Asia, Europe}\})$   
and  $L_2 = \{\text{top-secret, } \{\text{Europe, South-America}\}\}, \dots$

L1 dominates L2

L2 dominates L1

Neither L1 nor L2 dominates the other one



### 6.18 Label Domination Quiz Solution



### Label Domination Quiz

Select the best answer:

If  $L_1 = (\text{secret}, \{\text{Asia, Europe}\})$   
and  $L_2 = (\text{top-secret}, \{\text{Europe, South-America}\}), \dots$

L1 dominates L2

L2 dominates L1

Neither L1 nor L2 dominates the other one

While `secret < top-secret , {Asia, Europe}` cannot be compared with `{Europe, South-America}`.

### 6.19 Sensitive Data Quiz



#### Sensitive Data Quiz

Select the best answer:

Assume that label L1 of a document D1 dominates label L2 of document D2 when these labels are defined by (sensitivity level, compartment).

- D1 contains more sensitive data than D2.
- D2 is more sensitive than D1.
- The data contained in D2 has a narrower scope as defined by its compartment

## 6.20 Sensitive Data Quiz Solution



### Sensitive Data Quiz

Select the best answer:

Assume that label L1 of a document D1 dominates label L2 of document D2 when these labels are defined by (sensitivity level, compartment).

D1 contains more sensitive data than D2.

D2 is more sensitive than D1.

The data contained in D2 has a narrower scope as defined by its compartment

$L_1 > L_2$

$L_1 \cdot \text{comp} \supseteq L_2 \cdot \text{comp}$

In order for  $D_1$  to dominate  $D_2$ ,  $D_1$  must have a higher sensitivity level than  $D_2$ .

In addition, the compartment of  $D_1$  must contain the compartment of  $D_2$  in order to be 'greater' (based on the ordering rules for sets). For this to be the case, the compartment of  $D_2$  must be a subset of ("narrower" than) the compartment of  $D_1$ .

## 6.21 Using Labels for MAC: Confidentiality

The **Bell and LaPadula Model** (BLP) is a model that deals with confidentiality. Its development was funded by the DoD.

The model is concerned with disclosure of information and preventing information of one classification level from being shared with individuals possessing a clearance level of lower sensitivity.

Since the model was developed for the DoD, this model assumes the classification/clearance levels of top secret, secret, classified, and unclassified. In addition, a label can have a compartment that describes what a given document is about.

There are two main operations a user can perform with a file - read and write - and so the model defines a rule for each operation.

### 6.21.1 Read-Down Rule

The read rule says that a user with label L<sub>1</sub> can read a document with label L<sub>2</sub> only if L<sub>1</sub> dominates L<sub>2</sub>.

For example, a user with secret clearance may read a classified document.

This rule is the *read-down rule*, also known as the **simple security property**. A user can read documents that are classified at their clearance level or lower.

### 6.21.2 Write-Up Rule

The write rule says that a user with label L<sub>1</sub> can write a document with label L<sub>2</sub> only if L<sub>2</sub> dominates L<sub>1</sub>.

For example, a user with secret clearance may write a top secret document.

This rule is the *write-up rule*, also referred to as the **star property** (\*-property). A user can write documents that are classified at their clearance level or higher.

What is the rationale for write-up?

If I am a user with top secret clearance and I write a secret document, I may - inadvertently or maliciously - include top secret information in that document.

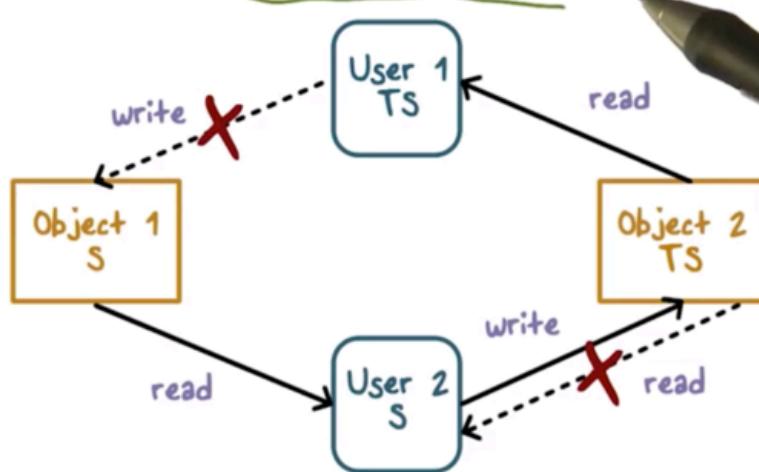
Since users with secret clearance can now read this document, we have broken the containment of top secret information to the top secret clearance level.

For this reason, users are only allowed to write documents at their clearance level or higher.

## 6.22 Preventing Information Flow with BLP

Information flow is a problem that cannot be addressed with discretionary access control. Mandatory access control - in particular, the BLP model - solves the information flow problem.

## Preventing Information Flow with BLP



Basically, we want to ensure that top secret information never ends up in the hands of individuals with less than top secret clearance.

The write-up and read-down rule prevent more sensitive information from flowing into the hands of users with a less sensitive clearance.

On one hand, users with secret clearance are not allowed to read top secret documents.

On the other hand, users with top secret clearance are not allowed to write secret documents, lest they reveal some top secret information in them.

A user with secret clearance is thus unable to access top secret information because: one, they are explicitly prohibited from reading top secret documents, and; two, users with top secret information are prohibited from writing that information into secret documents.

### 6.23 Unclassified Documents Quiz



#### Unclassified Documents Quiz

Select the best answer:

Since an unclassified document contains no sensitive information, it can be read or written by anyone in a system that implements BLP

True

False

### 6.24 Unclassified Documents Quiz Solution



**Unclassified Documents Quiz**  
Select the best answer:

Since an unclassified document contains no sensitive information, it can be read or written by anyone in a system that implements BLP

True  
 False



Write-down says that individuals cannot write documents with a classification that is less than their security clearance. Therefore, unclassified documents cannot be written by individuals holding a security clearance of classified, secret, or top secret.

### 6.25 Classified Data Quiz



#### Classified Data Quiz

Select the best answer:

BLP allows an unclassified user to write a top secret document.

True

False



## 6.26 Classified Data Quiz Solution



 Classified Data Quiz  
Select the best answer:

BLP allows an unclassified user to write a top secret document.

True  
 False

Because of the write-up rule, individuals are allowed to write documents at a classification level that is greater than their clearance level. Individuals with unclassified security clearance are thus allowed to write top secret documents.

### 6.27 BLP Model Quiz



### BLP Model Quiz

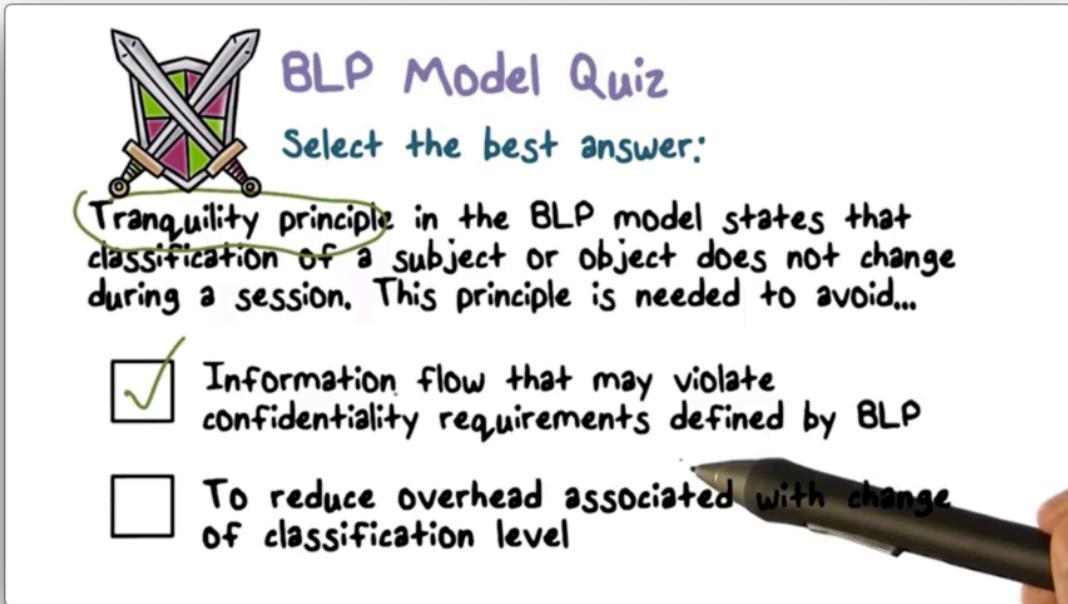
Select the best answer:

Tranquility principle in the BLP model states that classification of a subject or object does not change during a session. This principle is needed to avoid...

- Information flow that may violate confidentiality requirements defined by BLP
- To reduce overhead associated with change of classification level



## 6.28 BLP Model Quiz Solution



The slide features a logo of two crossed swords on a shield-like base. To the right, the text "BLP Model Quiz" is written in purple, followed by "Select the best answer:" in blue. Below this, a green oval highlights a statement: "Tranquility principle in the BLP model states that classification of a subject or object does not change during a session. This principle is needed to avoid...". Two options are listed with checkboxes: a checked box for "Information flow that may violate confidentiality requirements defined by BLP" and an unchecked box for "To reduce overhead associated with change of classification level". A hand holding a pen is visible on the right side of the slide.

**BLP Model Quiz**

Select the best answer:

Tranquility principle in the BLP model states that classification of a subject or object does not change during a session. This principle is needed to avoid...

Information flow that may violate confidentiality requirements defined by BLP

To reduce overhead associated with change of classification level

For example, if a user is writing to a top secret document, and the classification level suddenly changes to secret, the write-up rule is violated and information is flowing in the wrong direction.

## 6.29 Other MAC Models

Instead of focusing on confidentiality, the **Biba Model** focuses on integrity.

Remember, integrity is concerned with information quality, while confidentiality is concerned with information disclosure.

The Biba rules are the inverse of the BLP rules.

When you are concerned about information integrity, you want to *read up*. Information that is of lower quality than the information that you produce is of no interest to you.

In addition, Biba defines a write down rule. If you can produce information at a certain level of integrity, you can surely produce information at a lower integrity. Therefore, you can write documents at or below your integrity level.

For example, the New York Times may have a high level of integrity. That information can likely be trusted. Supermarket tabloids are low integrity. Information in those documents is likely of poor quality.

The compartment component of the integrity label can capture the topic(s) of the document, similar to BLP.

The point is that low integrity information should never flow into high integrity documents.

### **6.30 Policies for Commercial Environments Part 1**

In commercial environments, user clearance is not common. However, other requirements exist.

One common requirement is that data should only be accessed by certain applications, and only certain users should be able to gain access to those applications.

For example, only employees in payroll should have access to salary files.

Another requirement that arises in commercial environments is preventing conflict of interest. Users must be prevented from accessing information that can place them in a conflict of interest.

For example, a financial analyst advising a bank should not have access to confidential information from other banks.

Finally, a separation of duty requirement can be important for reducing the possibility of fraud. If a user can take on multiple roles, it can be easier for them to commit fraud.

For example, a user should not be able to both authorize and execute a financial transaction.

### **6.31 Policies for Commercial Environments Part 2**

Let's look at two different policies that make sense for commercial environments.

#### **6.31.1 Clark-Wilson Policy**

The **Clark-Wilson Policy** says that users should only be able to access certain applications, and only certain applications can access certain objects in the system.

Put another way, the access that a user has to objects in the system is constrained by the applications they are allowed to use.

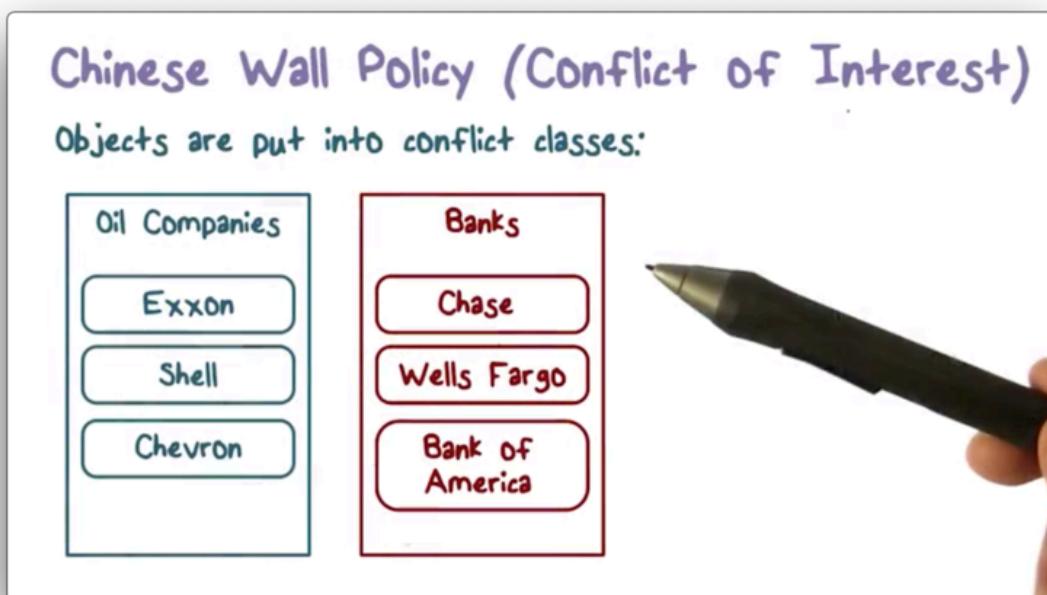
Achieving separation of duty through this policy is as simple as blocking access to the set of programs that would allow a user to conduct fraudulent activity.

### 6.31.2 Chinese Wall Policy

The **Chinese Wall Policy** deals with conflict of interest. Specifically, this model dictates that what a user can access now depends on what they have accessed in the past.

If the documents you have accessed so far have no conflict with the document you are currently requesting, access may be granted. Otherwise, access will be denied.

Let's assume we have the following documents in our system.



The group of documents in the “Banks” box are referred to as a **conflict class**. Since Wells Fargo, Chase, and Bank of America are all competitors, access to documents about more than one of these banks can present an individual with a conflict of interest.

Similarly, the “Oil Companies” conflict class contains documents about competitors Exxon, Shell and Chevron.

The Chinese wall policy basically says that a user can access any object as long as they have not previously accessed an object from the same conflict class.

For example, once a user accesses a document pertaining to Exxon, they can access a document pertaining to Chase, but cannot access a document pertaining to Shell.

### 6.32 Clark Wilson Quiz



#### Clark-Wilson Quiz

Check the best answer:

Clark-Wilson is a mandatory access control policy because...

- Any user in a company can decide what files can be accessed by a program
- Only the company can decide (e.g., sysadmin) what files can be accessed by a program.



### 6.33 Clark Wilson Quiz Solution



**Clark-Wilson Quiz**  
Check the best answer:

Clark-Wilson is a mandatory access control policy because...

Any user in a company can decide what files can be accessed by a program

Only the company can decide (e.g., sysadmin) what files can be accessed by a program.



In mandatory access control, sharing decisions are not made at the discretion of the user.

### 6.34 COI Quiz



#### COI Quiz

Select the best answer:

A large law firm currently has two client companies that compete with each other. Thus, its lawyers working on cases related to one company must not be able to access documents related to the other company. To ensure proper access control, which policy should the law firm use?

- Clark-Wilson
- Chinese Wall

### 6.35 COI Quiz Solution



### COI Quiz

Select the best answer:

A large law firm currently has two client companies that compete with each other. Thus, its lawyers working on cases related to one company must not be able to access documents related to the other company. To ensure proper access control, which policy should the law firm use?

Clark-Wilson *Cor*

Chinese Wall *✓*



Competition implies that there is a possibility for a conflict of interest. Chinese Wall is best at preventing these situations.

### 6.36 RBAC Quiz



#### RBAC Quiz

Select the best answer:

Role-based access control (RBAC) is often used in a commercial setting. RBAC is an example of MAC because...

- File permissions are associated only with roles and not users
- Only the company can decide roles of its employees



### 6.37 RBAC Quiz Solution



**RBAC Quiz**  
Select the best answer:

Role-based access control (RBAC) is often used in a commercial setting. RBAC is an example of MAC because...

- File permissions are associated only with roles and not users
- Only the company can decide roles of its employees



In mandatory access control, the company decides who can share what.

### 6.38 MAC Support Quiz



#### MAC Support Quiz

Select the best answer(s):

Which of the following operating systems supports a  
BLP-like model?

SELinux

Windows

MacOS

SCOMP



### 6.39 MAC Support Quiz Solution



**MAC Support Quiz**  
Select the best answer(s):

Which of the following operating systems supports a BLP-like model?

SELinux       MacOS  
 Windows       SCOMP



### 6.40 Revisiting TCB

How do we know that a certain system that claims to be a trusted computing base can actually be trusted?

First, it is important to talk about terminology.

When we talk about something being secure, we are usually talking about a binary quality of a system. Either something is secure or it isn't.

When we talk about a system being trusted, we mean that we have a high level of confidence that the system will uphold the requirements that we expect it to uphold.

This high level of assurance is what we expect from the trusted computing based comprised of the operating system and the underlying hardware.

Any other claims we are able to make about the security implemented in the system - access control, for example - depend on the security of the trusted computing base.

## 6.41 Trusting Software

Software is supposed to perform a set of functions. If it is going to implement certain functionality, it has to be functionally correct. It should do what it was designed to do.

To implement these functions, the software has to maintain certain data structures and state. Software often exposes an interface/API to allow users to consume and manipulate this state. Malicious users will use these entry points to attempt to compromise the system. The software must maintain the integrity of the data it relies on, especially in the face of malicious input.

The TCB stores and consumes sensitive data. It must protect this data and address the confidentiality/disclosure of this data. It must perform these duties in the presence of untrusted software.

Our assurance in the TCB describes how confident we are in its ability to carry out the above functions. We cannot always demand formal proof that a program works as expected, but there are certain actions that can increase our confidence.

For example, experts can analyze a program and assure the level of trust that a system provides. It is their word on which we derive our trust in the system.

In addition, we think about our requirement/needs as forming a specification of the security expectations we have for a system. Some vendor who builds the TCB (operating system) thus builds the system to those requirements.

The verification of trust should be formally (mathematically) derived where possible, but this level of certification is not always feasible.

## 6.42 TCB Design Principles

Trust in the trusted computing base is going to come from a number of design principles we expect a TCB to embody.

A trusted computing base should follow the principle of least privilege. Any time a user is executing, they should be provided with the least amount of privilege required to complete their task.

Remember, a trustworthy system is one that performs with a high likelihood of security. That being said, there is no guarantee. If something goes wrong - a malicious user gains access, for example - least privilege provides damage containment.

A trustworthy system should abide by economy of mechanism. The trusted code should be small and it should be simple. Simple code is easier to analyze and less code means less bugs.

A trustworthy system should have an open design. Trust is not going to come through blindly believing a software vendor. Security by obscurity is a lie. We want open design so we can know exactly how a certain level of security is being achieved.

A trusted system should enforce complete mediation. Every access is checked by the system and attempts to bypass the system must be prevented.

A trusted system should provide fail-safe defaults. This means that the default behavior should be the safest behavior. When we talk about access control, for example, the fail-safe default is to deny access.

Finally, a trusted system should be psychologically acceptable. We have to make the right assumptions about what users are able to do and what they are not able to do. Users avoid security that gets in their way.

Trust comes from building a system with these principles in mind.

#### 6.43 Least Privilege Quiz



#### Least Privilege Quiz

Select the best answer:

Least privilege is useful for damage containment when something goes wrong. Is this principle applicable to a TCB that must be trusted?

NO, because a TCB is guaranteed to function correctly

Yes, because TCB only provides high assurance and not a guarantee

#### 6.44 Least Privilege Quiz Solution



### Least Privilege Quiz

Select the best answer:

Least privilege is useful for damage containment when something goes wrong. Is this principle applicable to a TCB that must be trusted?

NO, because a TCB is guaranteed to function correctly

Yes, because TCB only provides high assurance and not a guarantee



The TCB provides high assurance, not certainty.

### 6.45 TCB High Assurance Quiz



#### TCB High Assurance Quiz

Select the best answer:

A TCB vendor claims its proprietary techniques help ensure high assurance, but cannot be disclosed. What principle does it violate?

- Complete mediation
- Open design

### 6.46 TCB High Assurance Quiz Solution



**TCB High Assurance Quiz**  
Select the best answer:

A TCB vendor claims its proprietary techniques help ensure high assurance, but cannot be disclosed. What principle does it violate?

Complete mediation  
 Open design



Security by obscurity violates open design.

### 6.47 Design Principle Quiz



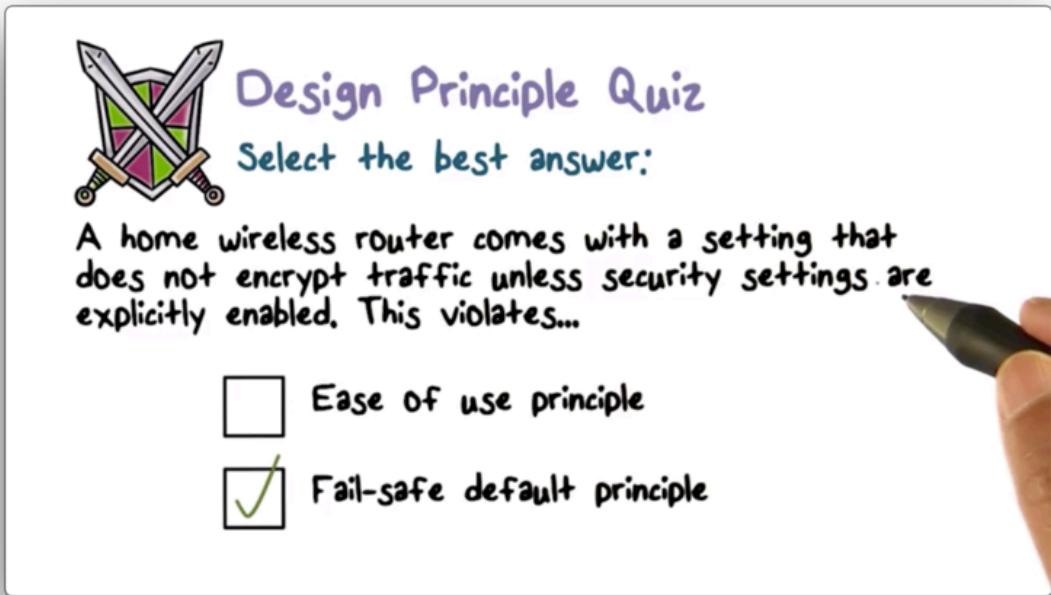
#### Design Principle Quiz

Select the best answer:

A home wireless router comes with a setting that does not encrypt traffic unless security settings are explicitly enabled. This violates...

- Ease of use principle
- Fail-safe default principle

### 6.48 Design Principle Quiz Solution



A hand-drawn style card titled "Design Principle Quiz". It features a shield with crossed swords icon and the text "Select the best answer:". Below is a question and two options with checkboxes. A hand holding a pen is shown on the right side.

**Design Principle Quiz**  
Select the best answer:

A home wireless router comes with a setting that does not encrypt traffic unless security settings are explicitly enabled. This violates...

Ease of use principle  
 Fail-safe default principle

A fail-safe default is one that provides security unless otherwise specified. In this case, the default should be traffic encryption.

### 6.49 Support Key Security Features

If we claim a system to be a trusted computing base, it must fulfill certain key requirements. Specifically, the system must implement certain functions that are relevant to security.

For example, the system must address authentication, otherwise it cannot know who is making a request for a resource in the system.

Additionally, the system must perform access control against the objects it protects. We talked about two different kinds of access control.

Systems may support mandatory access control, using the BLP model or other models that we discussed. SELinux is an example of a platform that supports MAC.

Systems can also support discretionary access control, whereby the creator of a resource has control over who can and cannot access it.

## 6.50 Data Protection

One of the requirements of a TCB is that it must be tamper-proof. In order to achieve this, an operating system must protect the data it uses.

When we talk about security features of trusted operating systems, one aspect that we can focus on is **object reuse protection**.

During execution, a process can allocate some temporary memory or disk space for storing information. When that process completes, the operating system needs to reclaim those resources to give to other processes.

If a process leaves behind some (potentially sensitive) data and the operating system reallocates the resources containing that data to a new process, the new process can potentially read that data. The operating system will have broken process isolation.

A system must understand that a process can leave behind some artifacts in these reusable resources and must ensure that they are “zeroed out” before being allocated again.

**Secure file deletion** refers to the operating system procedure of overwriting reclaimed disk space with varying patterns of zeros and ones in order to obfuscate the original contents.

**Secure disk destruction** is not really the job of the TCB, but rather is just a good practice when disposing of a computer. This involves [degaussing](#) or otherwise physically destroying your disk during disposal.

## 6.51 Trusted Paths

A TCB must ensure complete mediation. We have to guarantee that there is no way to get to a resource without passing through the trusted computing base.

We also need to have a trusted path from the user to the secure system. Without the trusted path, a malicious program could spoof the interface of the trusted system.

Alternatively, a malicious program could tap into the communication path between a user and the TCB in the absence of a trusted path.

For example, a keylogger could tap the communication path from the keyboard to the operating system and record your password.

A trusted path confers confidence that a user is communicating exclusively with the trusted computing base.

### 6.51.1 Auditing

We've talked a lot about prevention mechanisms, such as authentication and authorization, which are used to block inappropriate access to system resources.

The reality is that no matter how hard we work at prevention, things will go wrong. With this realization, we have to employ detection mechanisms in addition to prevention mechanisms.

We can keep an **audit log** of what happens in the system, and can use this log to detect unusual behavior that may be indicative of system misuse/abuse.

Naturally, the audit log, like all other data managed by the TCB, must also be tamper-proof.

[Butler Lampson](#) describes the gold standard of security as authentication, authorization and audit. The symbol for gold is "Au" in the periodic table.

### 6.52 Kernel Design

One approach for creating a trusted computing base is to design a **security kernel** where you bring the security-relevant functions together and place them at the core of the system you are trying to build.

In this approach, the security mechanisms and enforcement strategies live in the kernel and ensure the proper government of the rest of the system.

Kernel design strives to find the smallest possible system that encapsulates the necessary security mechanisms. This size requirement helps to ensure that the kernel is isolated from other parts of the system. In addition, smaller kernels lend themselves to easier analysis/verification.

Like other trusted computing bases, a security kernel must be tamper-proof and enforce complete mediation.

### 6.53 Kernel Design and TCB

The mechanisms that are necessary for the correct enforcement of security policy belong in the security kernel.

Resources are implemented using physical hardware that we attach to the system, and the correct implementation of security blocks direct access to those physical resources.

Complete mediation can also be handled by virtualization. In virtual environments, host operating systems have handles for virtual resources, and the hypervisor allows us to achieve both hardware isolation and logical OS separation.

## 6.54 Revisiting Assurance

In addition to describing the functionality that a TCB must possess, we need some *assurance* that it is actually carrying out those functions.

One basic check involves interacting with the developers of the system and ensuring that they have implemented all of the necessary requirements.

A more structured way to provide assurance is to extensively test the system. There are three main types of testing.

One type of testing looks at functionality. We write tests to ensure that our functions execute and our software is not “buggy”.

Another type of testing is **penetration testing**. In penetration testing, the tester approaches the software with the adversarial mindset and specifically tries to uncover vulnerabilities in the system.

The final kind of testing is **regression testing**, which we run when we enhance a system to make sure that new features don’t break functionality or hurt performance of existing features.

As a word of caution: testing can only demonstrate the existence of a problem. Tests cannot demonstrate the absence of a problem. For example, you cannot run a suite of tests and then claim that the system is completely secure.

Even better than testing is to have a formal proof that irrefutably verifies the functionality and security of a system.

## 6.55 Assurance Challenges

### 6.55.1 Functional Testing

A developer must generate the right set of test cases when testing a system. Test cases demonstrate the existence of problems, and without a sufficient number of test cases, problems may go undiscovered.

In testing a system, you want to have good code coverage. Ideally, you want to be able to test every execution path present in the system, so you can concretely understand any behavior that the system will exhibit when it is deployed.

The problem with this ideal is that execution paths grow exponentially in a system. For example, every **if-else** construct doubles the number of execution paths that need to be tested.

In addition, a system may behave differently in different execution environments, which also adds to the difficulty of achieving comprehensive testing.

### 6.55.2 Penetration Testing

In penetration testing, we deliberately employ an ethical hacker to attempt to defeat the security measures we have in place.

As the good guys, we have the burden of ensuring that every possible execution path is secure. In penetration testing, the tester just needs to find *some* execution path containing a vulnerability that can be exploited in order to demonstrate a problem.

If they can't find any issues, this doesn't prove that we don't have a problem. Testing - penetration or otherwise - cannot demonstrate the absence of a problem.

### 6.55.3 Formal Verification

A formal verification is a mathematical specification of a program's behavior. This verification proves which security assertions and properties hold throughout the execution of a system.

There are automated ways to formally verify a program, such as [model checking](#) and [automated theorem proving](#).

These approaches typically look at the initial set of state variables in a program and watch how the code changes these variables. Throughout the execution of the program, assertions about security are verified against the program state.

The problem with model checking is the same that we had with testing. The state space grows exponentially which makes this approach unfeasible for anything other than relatively small programs.

Model checking pioneers won the Turing Award in 2007 and this field remains an area of active research.

### 6.56 Reducing TCB Size Quiz



#### Reducing TCB Size Quiz

Check all applicable answers:

We discussed the need for reducing the size of the TCB. This helps with...

- Testing of the TCB
- Verification of the TCB
- Isolation of the TCB



### 6.57 Reducing TCB Size Quiz Solution



#### Reducing TCB Size Quiz

Check all applicable answers:

We discussed the need for reducing the size of the TCB. This helps with...

- Testing of the TCB
- Verification of the TCB
- Isolation of the TCB



### 6.58 Testing TCB Quiz



#### Testing TCB Quiz

Check the correct answer:

Testing is challenging for a complex system like a TCB because of...

- It is hard to cover all executions
- It can show both existence and absence of problems



### 6.59 Testing TCB Quiz Solution



## Testing TCB Quiz

Check the correct answer:

Testing is challenging for a complex system like a TCB because of...

- It is hard to cover all executions
- It can show both existence and absence of problems



Testing can't show the absence of problems.

### 6.60 Model Checking Quiz



#### Model Checking Quiz

A key problem with model checking is...

- It cannot show absence of a problem
- It does not scale to practical large size systems

## 6.61 Model Checking Quiz Solution



**Model Checking Quiz**  
A key problem with model checking is...

- It cannot show absence of a problem
- It does not scale to practical large size systems

## 6.62 Security Evaluations: The Orange Book

Many people are faced with the problem of trying to evaluate the level of security present in a system.

The Department of Defense created the [Trusted Computer System Evaluation Criteria](#) to address this problem.

This document defines a ranking of system security using the following divisions (letters) and classes (numbers).

1 D < C1 < C2 < B1 < B2 < B3 < A1

Division D is reserved for systems that do not meet the requirements for a higher division.

A system has to implement authentication and discretionary access control to be considered for division C. A system must implement mandatory access control to be considered for division B. A system must be formally verified via mathematical techniques to be considered for division A.

Most commercial systems have been evaluated at C1, C2, or B1.

In order to advance to B2, a system needs to demonstrate a proof of correctness for the underlying security model and provide a narrative specification for the TCB.

A system that provides a formal verification of TCB correctness is a candidate for B3/A1.

### 6.63 Security Evaluations: Common Criteria

After time, multiple countries began to recognize the importance of security evaluation, and TCSEC was replaced by an international standard which combined US, Canadian and European efforts: the [Common Criteria](#).

The idea is that as a consumer of a system, users specify what security they want to see in their software. Vendors implement these requirements and make claims about their solution meeting users' needs. The evaluator determines if the claims that are made match the specifications.

The **evaluation assurance level** (EAL) rates the trustworthiness of the system. EAL1 is the most basic rating and EAL7 is the most rigorous.

### 6.64 TCSEC Divisions Quiz



TCSEC Divisions Quiz

Many widely used operating systems do not support MAC and hence cannot be in a TCSEC division higher than...

D

C

### 6.65 TCSEC Divisions Quiz Solution



### TCSEC Divisions Quiz

Many widely used operating systems do not support MAC and hence cannot be in a TCSEC division higher than...

D

C



### 6.66 Earning an EAL4 Certification Quiz



#### Earning an EAL4 Certification Quiz

How did VMware vCloud Networking and Security v5.5 system receive an EAL4+ certification?

- The system developers used formal techniques in its design and testing
- A systematic review and testing process was used by the system developers

### 6.67 Earning an EAL4 Certification Quiz Solution



#### Earning an EAL4 Certification Quiz

How did VMware vCloud Networking and Security v5.5 system receive an EAL4+ certification?

- The system developers used formal techniques in its design and testing
- A systematic review and testing process was used by the system developers



### 6.68 Cost Benefit Certification Tradeoffs Quiz



#### Cost-Benefit Certification Tradeoffs Quiz

Many OS vendors do not aim for the highest certifications because...

- There is no market demand for such certifications
- Cost/benefit tradeoffs dictate the highest certification

### 6.69 Cost Benefit Certification Tradeoffs Quiz Solution

**Cost-Benefit Certification Tradeoffs Quiz**

Many OS vendors do not aim for the highest certifications because...

There is no market demand for such certifications

Cost/benefit tradeoffs dictate the highest certification

## 7 Database Security

### 7.1 Importance of Database Security

Databases store massive amounts of sensitive data. When hackers steal millions of customer records from a company - often containing data like social security numbers, addresses, and credit card numbers - they usually steal this information from one or more databases.

Data often has structure, and databases expose abstractions that help users store their data in structured ways. This data can then be accessed and manipulated using structured queries that we send to the database. **Structured Query Language** (SQL) is a common language for expressing these queries.

Databases store data that is persistent, and the integrity of this data is important. Running a partial query can leave a database in an inconsistent state, which can impact integrity. As a result, queries are often transactional, which means they either run completely or are completely undone.

For example, consider a query that moves money from one account to another account. Both the withdrawal and the deposit must succeed, otherwise both will have to be undone.

When a user executes a query, they can store the result in a virtual database called a **database view**.

For example, a user could query a database of employees and ask for a view that shows the name, address, and phone number, but not the salary, for each employee.

Database views are powerful tool for achieving access control. A database owner can extract some subset of the data and give other users access to only that subset.

## 7.2 Database Threats Quiz



### Database Threats Quiz

Choose the best answer.

Oracle, a major database vendor, sponsored a database security study which identified key security threats. In your view, which of the following is the biggest threat...

External hackers

Insiders and unauthorized users



### 7.3 Database Threats Quiz Solution



## Database Threats Quiz

Choose the best answer.

Oracle, a major database vendor, sponsored a database security study which identified key security threats. In your view, which of the following is the biggest threat...

External hackers

Insiders and unauthorized users



[Source](#)

#### 7.4 Database Hacking Quiz



#### Database Hacking Quiz

Mark all applicable answers.

Databases are attractive targets for hackers because...

- They store information such as SS#, DOB etc. that can be easily monetized
- They store information about lots of users
- Queries languages used to access data can be abused to gain unauthorized access.



## 7.5 Database Hacking Quiz Solution



### Database Hacking Quiz

Mark all applicable answers.

Databases are attractive targets for hackers because...

- They store information such as SS#, DOB etc. that can be easily monetized
- They store information about lots of users
- Queries languages used to access data can be abused to gain unauthorized access.



## 7.6 Relational Database Systems (RDBS) Part 1

Relational databases are widely used in real-world environments.

A relational database consists of **relations**. Relations are also referred to as tables. A table consists of rows and columns.

The columns of a given table describe the attributes of the data to be stored in that table. A database **schema** defines these columns.

For example, we may define the following columns for a table of employees:

- name
- DOB
- salary
- social security number

Tables also have rows. A given row contains a value for each of the attributes as defined by the schema. These rows are called **tuples**.

For example, the tuple for an employee named Bob might look like:

- Bob
- 1/1/1970
- 57,000
- 123-45-6789

Some of the columns in a table hold special values called **keys**. Keys uniquely identify a given tuple. For example, in the above table describing employees, the social security number column contains the key for each tuple since employees can be uniquely identified by this value.

## 7.7 Relational Database Systems (RDBS) Part 2

Let's look at an example of a relation that contains information about employees.

**Relational Database Systems (RDBS)**

**Employee Table**

Ename	Did	Salarycode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380

This table defines the following columns:

- Employee name
- Department ID
- Salary Code
- Employee ID
- Employee phone number

Each employee will occupy one row in this table and will have data for each of the columns.

For example, Neil has an employee ID of 5088, a department ID of 13, a salary code of 12, and a phone number.

The employee ID is used to uniquely identify each employee in this table. The employee ID is the **primary key** for this relation.

Sometimes, you want to perform operations across tables.

For example, imagine there is a table of departments that has columns for a department ID and a department name. Suppose you wanted to create a query that selected the name of each employee and the department name (not ID) where each employee works.

We need a way of referencing the departments relation from the employees relation, so we can **join** the two and select the data we want. We can do this using the department ID.

In the employees table, the department ID is a **foreign key**. It references a primary key of another table: the departments table. Foreign keys are used to define relationships between tables and can be used to query those relationships.

## 7.8 Relational Database Systems (RDBS) Part 3

There are many different operations that we can perform on tables.

We can *create* a table. We can *select* certain information out of a table. We can *insert* new tuples or columns. We can *update* attribute values for certain tuples. We can perform *join* operations to combine multiple tables. We can *delete* tuples, columns, tables, and entire databases.

The following is a SQL query - a selection - that retrieves all of the attribute values for each employee in the above employee relation that works in the department with ID 15.

```
1  SELECT * FROM Employee  
2 WHERE DID = '15';
```

### 7.9 Key Value Quiz



#### Key Value Quiz

Choose the best answer:

Two tuples (rows) in a relation can have the same primary key value.

Yes

No

### 7.10 Key Value Quiz Solution



**Key Value Quiz**  
Choose the best answer:

Two tuples (rows) in a relation can have the same primary key value.

Yes

No



A primary key uniquely identifies a row.

### 7.11 Database Views Quiz



#### Database Views Quiz

Choose the best answer.

We can use a database view to enhance data security because...

- It can exclude sensitive attributes that should not be accessible to certain users
- A view can only be accessed by a single user

## 7.12 Database Views Quiz Solution



### Database Views Quiz

Choose the best answer.

We can use a database view to enhance data security because...

- It can exclude sensitive attributes that should not be accessible to certain users
- A view can only be accessed by a single user



## 7.13 Database Access Control

There are two basic SQL commands for implementing database access control.

### 7.13.1 Grant

The SQL command for granting privileges has the following syntax.

```
1 GRANT      { privileges | role }
2 [ON        TABLE]
3 TO        { user | role | public }
4 [IDENTIFIED BY password]
5 [WITH      GRANT OPTION]
```

The first line of the command describes what we would like to grant. We can grant either privileges - like `SELECT` - or an entire role, such as `admin`.

We can optionally scope our grant to a specific table.

Our grant must have a recipient, be it a particular user, role, or `public`. `Public` refers to the entirety of users/roles that interact with the database.

We can optionally require users to identify themselves using a certain password if they wish to revoke this grant.

Finally, we can optionally allow the grantee to further propagate their access with the `WITH GRANT OPTION` clause.

For example, if we would like to grant Alice the ability to perform a `SELECT` on any table, and propagate that access to others, we could issue the following command:

```
1 GRANT SELECT ON ANY TABLE TO Alice WITH GRANT OPTION;
```

Alternatively, we could grant access for `INSERT`, `UPDATE` and/or `DELETE`.

### 7.13.2 Revoke

The SQL command for revoking privileges has the following syntax.

```
1 REVOKE { privileges | role }
2 [ON      TABLE]
3 FROM    { user | role | public }
```

The first line of the command describes the privilege or role that we would like to revoke.

We can optionally scope our revocation to a specific table.

Finally, we must revoke access from a particular entity. This entity can be a user, role, or the general public.

For example, to revoke the access we just gave to Alice, we issue the following command:

```
1 REVOKE SELECT ON ANY TABLE FROM Alice;
```

### 7.14 Database Access Control Quiz



#### Database Access Control Quiz

Choose the best answer.

Alice has SELECT access to a table and she can propagate this access to Bob when...

- Alice was granted this access with GRANT option
- She can always propagate an access she has

### 7.15 Database Access Control Quiz Solution



#### Database Access Control Quiz

Choose the best answer.

Alice has SELECT access to a table and she can propagate this access to Bob when...

- Alice was granted this access with GRANT option
- She can always propagate an access she has

### 7.16 Cascading Authorizations Quiz



#### Cascading Authorizations Quiz

Choose the best answer.

Cascading authorizations occur when an access is propagated multiple times and possibly by several users. Assume that Alice grants access to Bob who grants it further to Charlie. When Alice revokes access to Bob, should Charlie's access be also revoked?

Yes

No



### 7.17 Cascading Authorizations Quiz Solution



### Cascading Authorizations Quiz

Choose the best answer.

Cascading authorizations occur when an access is propagated multiple times and possibly by several users. Assume that Alice grants access to Bob who grants it further to Charlie. When Alice revokes access to Bob, should Charlie's access be also revoked?

Yes       No



### 7.18 DAC or MAC Quiz



#### DAC or MAC Quiz

Choose the best answer.

Database access control can be managed centrally by a few privileged users. This is an example of...

DAC

MAC



### 7.19 DAC or MAC Quiz Solution



**DAC or MAC Quiz**  
Choose the best answer.

Database access control can be managed centrally by a few privileged users. This is an example of...

DAC  
 MAC



### 7.20 Attacks on Databases: SQL Injections Part 1

One famous attack that is carried out against databases is the SQL injection attack.

In an **SQL injection attack**, an attacker sends a malicious command to the database, which allows them to interact with database data in unauthorized ways. The results of this malicious query can result in unauthorized reading and/or writing of data.

These attacks can disclose large amounts of data - an attack on confidentiality - and they can also corrupt data - an attack on integrity.

Understanding how an SQL injection attack is carried out first requires some understanding of common application architecture.

Often, the database resides in the back-end of the system. The front-end of the system that the user interacts with is typically a web application.

The user will present data to the application, often via forms, and the application will translate this data into database queries.

The application will send these queries to the database and then present the results to the user in some informative way.

The SQL injection attack occurs as a result of the application not properly sanitizing the input received from the user before sending that input to the database.

## 7.21 Attacks on Databases: SQL Injections Part 2

Consider a user interacting with a shipping platform.

Perhaps the application presents the user with a web form that asks them to enter a shipping city and then shows them all of the shipments destined for that city.

The script code might look something like this:

```
1 var Shipcity;
2 Shipcity = Request.form("Shipcity");
3 var sql = "select * from OrdersTable where Shipcity = '" + Shipcity + "';
```

If the user submits “Redmond” on the form, the SQL query sent to the database becomes:

```
1 SELECT * from OrdersTable where Shipcity = 'Redmond';
```

What if the user submits “Redmond”; DROP table OrdersTable;”?

This SQL query will be sent to the database:

```
1 SELECT * from OrdersTable where Shipcity = 'Redmond'; DROP table
OrdersTable;
```

In this case, the user is taking a simple text input field and is injecting SQL into that field to be sent to the database.

The `SELECT` statement will still be executed, but it will be followed by a `DROP table` statement, which will delete all of the orders.

The vulnerability that we have in the web application is that it is not checking input. It is accepting any input and passing it blindly back to the database.

## 7.22 SQL Injection Defenses

As always, input checking remains a crucial defense for blocking SQL injection. If we assume all input is evil, we will be much more careful with that input before we send it to the database.

In addition, the Open Web Application Security Project (OWASP) defines a list of [ten proactive controls](#) for guarding against common software vulnerabilities, including SQL injection.

### 7.23 SQL Login Quiz

 **SQL Login Quiz**  
Mark all applicable answers.

A web application script uses the following code to generate a query:

Query = "SELECT accounts FROM users WHERE login = '" + login + "' AND pass = '" + password + "',  
AND pin = " + pin; The various arguments are read from a form to generate Query.

This query is executed to get a user's account information when the following is provided correctly...

Login name     Password     PIN

## 7.24 SQL Login Quiz Solution

 **SQL Login Quiz**  
Mark all applicable answers.

A web application script uses the following code to generate a query:

Query = "SELECT accounts FROM users WHERE login = '" + login + "' AND pass = '" + password + "' AND pin = '" + pin; The various arguments are read from a form to generate Query.

This query is executed to get a user's account information when the following is provided correctly...

Login name     Password     PIN



## 7.25 SQL Login Quiz 2



### SQL Login Quiz #2

Choose the best answer.

Query = "SELECT accounts FROM users WHERE login = '' + login + '' AND pass = '' + password + '' AND pin = '' + pin; The various arguments are read from a form to generate Query.

If a user types " or 1 = 1 --" for login in the above query...

- Query will fail because the provided login is not a correct user
- An injection attack will result in all users' account data being returned

## 7.26 SQL Login Quiz 2 Solution

 **SQL Login Quiz #2**  
Choose the best answer.

Query = "SELECT accounts FROM users WHERE login = '' + login + '' AND pass = '' + password + '' AND pin = '' + pin; The various arguments are read from a form to generate Query.

If a user types "'or 1 = 1 --' for login in the above query...

Query will fail because the provided login is not a correct user  
 An injection attack will result in all users' account data being returned

## 7.27 Inference Attacks on Databases Part 1

**Inference attacks** occur when a user is able to make inferences about data that they are not authorized to access based on queries that they are authorized to execute.

## 7.28 Inference Attacks on Databases Part 2

Consider a database of students with the following schema:

- student ID
- student standing (junior/senior)
- exam 1 score
- exam 2 score
- final exam score

Suppose that students are not authorized to execute a query that will reveal another student's exam grade. However, students are authorized to execute a query `Q` that returns the average grade for an exam across all students.

An attacker wants to gain access to the exact score of another student for a particular exam and may be able to infer the grade through via [Q](#).

Inference attacks sometimes require some additional outside information. In this case, let's suppose that the attacker knows that the particular student takes the exam late.

The attacker can deduce the exam grade of this student by executing [Q](#) before the student takes the exam and then again after the student takes the exam.

For example, if there are 10 students in the class and the [Q](#) returns 100 before the student takes the exam, and 99 after, we can calculate the student's score as:

```
1 99 = (9 * 100 + x) / 10;  
2 990 = 900 + x;  
3 x = 90;
```

Again, the problem demonstrated here is that authorized queries like [Q](#) allow users to make inferences about data that they are not authorized to access.

## 7.29 Inference Attacks on Databases Part 3

The previous example might be a little contrived, so let's consider another scenario.

Suppose that [Q](#) is augmented to allow users to constrain the group of students over which they want to see the average score calculated.

For example, since our database has a field for student year - either junior or senior - suppose [Q-senior](#) allows students to calculate the average of only seniors and [Q-junior](#) allows students to calculate the average of only juniors.

A junior might execute [Q-junior](#) as a way to compare themselves with their peers.

Of course, a problem arises when there are only two juniors present in the class. One junior executing [Q-junior](#) can easily deduce the score of the other junior.

## 7.30 Defenses Against Inference Attacks

One way that we can guard against inference attacks is by preventing the access of aggregate information about a set of tuples when the size of that set is too small or too large.

In the most extreme case, aggregate information about a set containing one tuple is simply just the data present in that tuple.

For example, if a group containing one student has an average score of 90 on an exam, that one student scored a 90.

In addition, if you are selecting aggregate information about a set that is very large, the aggregate information roughly holds for everyone, and if it holds for everyone, it holds for a given user as well.

### 7.31 Defenses Against Inference Attacks Process

We can further defend against inference attacks by removing identifying information from databases. We can **de-identify** the exam score table by dropping the student ID column.

Even with de-identification, inference attacks may still be possible.

For example, if we know that there is one junior in the class, we can still `SELECT` the tuple where the student year is “junior” to reveal information about that student. We must remove or change the student year column to **anonymize** this student.

In addition to de-identification and anonymization, we can use **generalization**. For example, we might generalize the data in the student year column to say either “upperclassmen” or “underclassmen” instead of revealing the exact year.

This still may not be sufficient to defend against inference attacks if the data is not diverse enough.

### 7.32 SQL Inference Attack Quiz



### SQL Inference Attack Quiz

Choose the best answer.

The database that stores student exam scores allows queries that return average score for students coming from various states. Can this lead to an inference attack in this system?

Yes, depending on how many students come from each state

No, it is not possible

### 7.33 SQL Inference Attack Quiz Solution



#### SQL Inference Attack Quiz

Choose the best answer.

The database that stores student exam scores allows queries that return average score for students coming from various states. Can this lead to an inference attack in this system?

- Yes, depending on how many students come from each state
- No, it is not possible

### 7.34 SQL Inference Attack Quiz 2



#### SQL Inference Attack Quiz #2

Choose the best answer.

Assume in (1), the data in the database is de-identified by removing student id (and other information such as names). Furthermore, the field that has the state of the student is generalized by replacing it with the US region (e.g., Midwest). The generalization ensures that there are at least two students from each region. Are inference attacks still possible?

Yes

No

### 7.35 SQL Inference Attack Quiz 2 Solution



#### SQL Inference Attack Quiz #2

Choose the best answer.

Assume in (1), the data in the database is de-identified by removing student id (and other information such as names). Furthermore, the field that has the state of the student is generalized by replacing it with the US region (e.g., Midwest). The generalization ensures that there are at least two students from each region. Are inference attacks still possible?



Yes



No

Consider the case where one student from a region containing two students retrieves the grade information about that region.

## 8 Malicious Code

### 8.1 What is Malware Quiz



#### What is Malware? Quiz

What are the estimated yearly losses due to cybercrime worldwide?

- \$100 million - \$500 million
- \$500 million - \$1 billion
- \$100 billion - \$500 billion

## 8.2 What is Malware Quiz Solution

**What is Malware? Quiz**

What are the estimated yearly losses due to cybercrime worldwide?

- \$100 million - \$500 million
- \$500 million - \$1 billion
- \$100 billion - \$500 billion

## 8.3 Types of Malware

There are two major types of malware.

The first type of malware embeds itself in a host program and executes on the system when its host runs.

The second type of malware interacts with systems as independent programs that can run by themselves.

Malware can embed itself in a host through

- trap doors
- logic bombs
- Trojan horses
- viruses
- browser plugins/extensions

Examples of independent malware include

- worms

- botnets
- advanced persistent threats (APTs)

## 8.4 Trap Doors

**Trap doors** - also known as backdoors - are secret entry points to a system that are typically only known to the developer of the system.

Trap doors are typically activated when a user provides a special sequence of input or a special user ID to the system.

Trap doors can be really helpful for programmers who need or want rapid access to a system without providing the proper user authentication.

Of course, an attacker with possession of a trap door can wreak havoc on a system.

A famous but naive version of a trap door - sometimes referred to as an **easter egg** - is the [flight simulator](#) embedded in the 1997 version of Microsoft Excel, which was activated by an undocumented series of commands entered by the user.

## 8.5 Logic Bombs

A **logic bomb** is essentially a trigger planted in a program. When the triggering condition is met, the planted code is executed.

For example, the [Code Red](#) worm contained a logic bomb that launched a denial of service attack at the White House web server on a specific date.

## 8.6 Trojan Horses

Trojan horses get their name from an ancient Greek story from the Trojan War. From [wikipedia](#):

In the canonical version, after a fruitless 10-year siege, the Greeks constructed a huge wooden horse, and hid a select force of men inside including Odysseus. The Greeks pretended to sail away, and the Trojans pulled the horse into their city as a victory trophy. That night the Greek force crept out of the horse and opened the gates for the rest of the Greek army, which had sailed back under cover of night. The Greeks entered and destroyed the city of Troy, ending the war.

In the context of malware, a Trojan horse is a piece of malicious code embedded in a utility program that a user runs frequently. When the utility program runs, the malicious code runs with it.

An example of a trojan horse is a login program that performs keylogging. This program will allow the user to login by calling the real login subroutine - a useful utility - while also stealing the user's credentials.

Many malicious browser extensions/plugins perform keylogging under the guise of offering some helpful utility.

## 8.7 Viruses

A **virus** infects a program by modifying the program code so that when a program runs the virus code also runs. A virus is unique in that it is able to spread, much like a real virus.

There are four main stages in the lifecycle of a virus.

The first phase is the *dormant* phase. In this phase, the virus has infected the host system, but remains idle.

The second phase is the *propagation* phase. In this phase, the virus multiplies and spreads. It can copy itself into other programs on the same host, or it can send itself to other hosts - by way of an email attachment, for example.

In the *triggering* phase, the virus is activated for execution. A user clicking on an email attachment containing a virus may trigger that virus for activation.

The final phase is the *execution* phase. In this phase, the virus actually performs its malicious work. For example, an executing virus might delete all of the files on disk.

## 8.8 Host Required Malware Quiz 1



### Host-Required Malware Quiz #1

Determine which category each of these belongs to:

- An email attachment that when being opened will send itself to all people in the user's address book.
- A customized keyboard app that logs user input and sends it to a server on the Internet.
- Part of a program will only run if the computer is at the user's home, and it will upload all MS Word docs to a web site.
- A login program with an undocumented option (e.g., DEBUG) that would allow an attacker to supply any username and password to gain access to the computer.

T = trapdoor, L = logic bomb, H = trojan horse, V = virus

## 8.9 Host Required Malware Quiz 1 Solution



### Host-Required Malware Quiz #1

Determine which category each of these belongs to:

- V An email attachment that when being opened will send itself to all people in the user's address book.
- H A customized keyboard app that logs user input and sends it to a server on the Internet.
- L Part of a program will only run if the computer is at the user's home, and it will upload all MS Word docs to a web site.
- T A login program with an undocumented option (e.g., DEBUG) that would allow an attacker to supply any username and password to gain access to the computer.

T = trapdoor, L = logic bomb, H = trojan horse, V = virus



### 8.10 Host Required Malware Quiz 2



#### Host-Required Malware Quiz #2

Which type of malware would be best for each of the given tasks?

- spy on employees of a specific company
- cripple an organization's computers
- quickly spread information and drive traffic to a specific website

T = trapdoor, L = logic bomb, H = trojan horse, V = virus

### 8.11 Host Required Malware Quiz 2 Solution



### Host-Required Malware Quiz #2

Which type of malware would be best for each of the given tasks?

- H spy on employees of a specific company
- L cripple an organization's computers
- V quickly spread information and drive traffic to a specific website

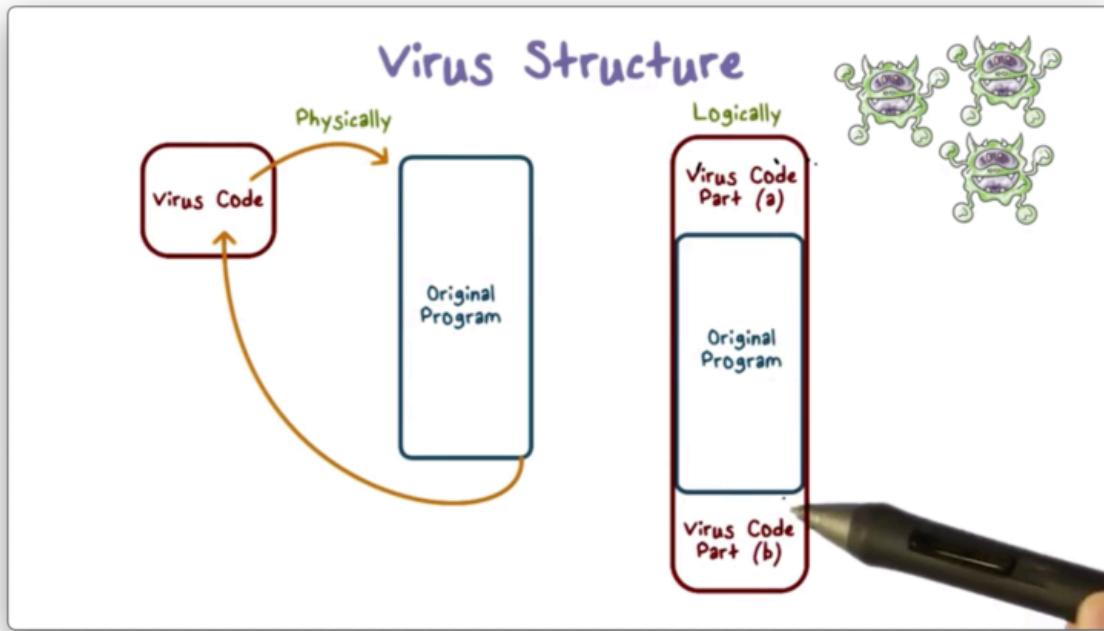
T = trapdoor, L = logic bomb, H = trojan horse, V = virus



### 8.12 Virus Structure

A virus infects a program by modifying the program code. In order to achieve this, the virus code has to be physically inserted into the program file.

When an infected program runs, the virus code runs first. The virus code then runs the original program so that the user doesn't suspect that the program has been infected. Finally, the virus code runs again, often to perform some cleanup to avoid detection.



The first line of the infected program must ensure that the virus runs immediately. This can be achieved with an instruction that calls the `main` function of the virus.

The virus code must also place a marker on the infected program to indicate that the program has been infected. Without this flag, a program could be repeatedly infected.

When the virus executes, it first finds other programs to infect. It will scan other applications on the system and infect those that do not have the special infected flag set.

In addition to infecting other programs, the virus can perform other malicious activities on the system.

Finally, the virus will transfer control to the original program so that normal work can be performed. This helps to prevent the user from detecting the infection.

The virus can perform other actions in order to avoid detection.

For example, when a virus is inserted into a program file, the size of that file increases. This increase can be a telltale sign that a program has been infected; therefore, the virus code might compress the infected program so that the file size appears unchanged.

### 8.13 Types of Viruses

A **parasitic virus** scans non-running programs on the system - for example, those that reside on the hard drive - and then infects those programs.

A **memory-resident virus** is typically part of an operating system. When the operating system runs, the virus is loaded into memory and can infect any running program on the system.

A **macro virus** is a virus embedded in a document. The virus runs when the document is opened.

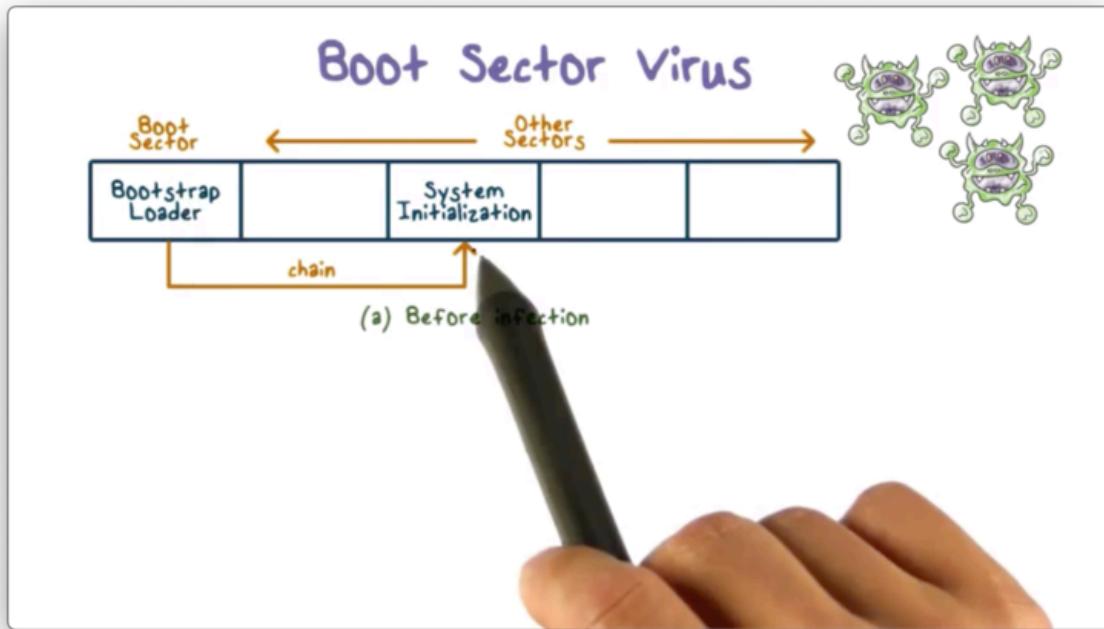
A **boot sector virus** resides in the boot sector of the hard drive and executes whenever the system is booted.

A **polymorphic virus** “looks different” with each infection. This is achieved by encrypting a portion of the virus code with a randomly generated key during each infection. The purpose of using polymorphic viruses is to avoid detection by anti-virus systems that rely on [virus signatures](#).

Any of these viruses can be polymorphic.

### 8.14 Boot Sector Virus

In order to understand a boot sector virus, we need to understand how the boot sector works.



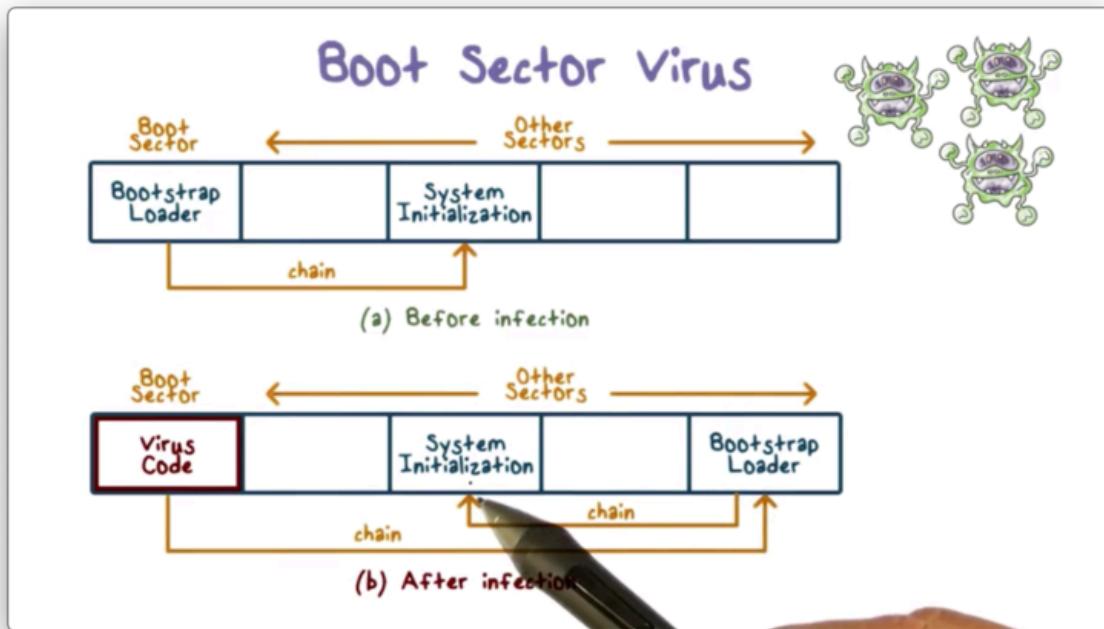
A **boot sector** is a special sector on the hard drive of a system. When a system is booted, the code in the boot sector - called the **bootstrap loader** - always runs first.

The bootstrap loader is typically responsible for loading the operating system.

When a boot sector virus infects a system, the virus code is inserted in the boot sector. This ensures that the virus always executes first during system boot.

During system boot, the virus performs its malicious functions, such as infecting other programs, spreading to other systems, or destroying useful documents.

After the virus executes, it transfers control to the original bootstrap loader in order to give the appearance that the system is functioning normally.



## 8.15 Macro Viruses

A **macro** is a program embedded in a document, such as a Microsoft Word document. A macro typically contains instructions for some useful functions, such as opening a file or starting a new application.

Because a macro is an executable program, it can be infected by viruses just like any other executable programs.

Macro viruses are unique in that users don't typically expect a document to contain a virus. As a result, attackers have had success spreading infected documents via email attachments.

When an unsuspecting user clicks on the email attachment and opens the document, the macro executes, and the macro virus runs.

The macro virus can perform malicious activities such as sending the infected document to every person in the user's address book.

A macro virus can also copy its macro to the global macro file. Whenever a user opens a new document, this now global macro will be copied into the new document, meaning that all new documents on the user's system will be infected.

### 8.16 Types of Viruses Quiz



#### Types of Viruses Quiz

Which type of virus begins on the operating system level?

- Macro virus
- Boot sector virus
- Memory-resident virus

### 8.17 Types of Viruses Quiz Solution



### Types of Viruses Quiz

Which type of virus begins on the operating system level?

- Macro virus
- Boot sector virus
- Memory-resident virus

Macro viruses run when an infected document is opened with a given application. Boot sector viruses run before the operating system is loaded.

### 8.18 Rootkit

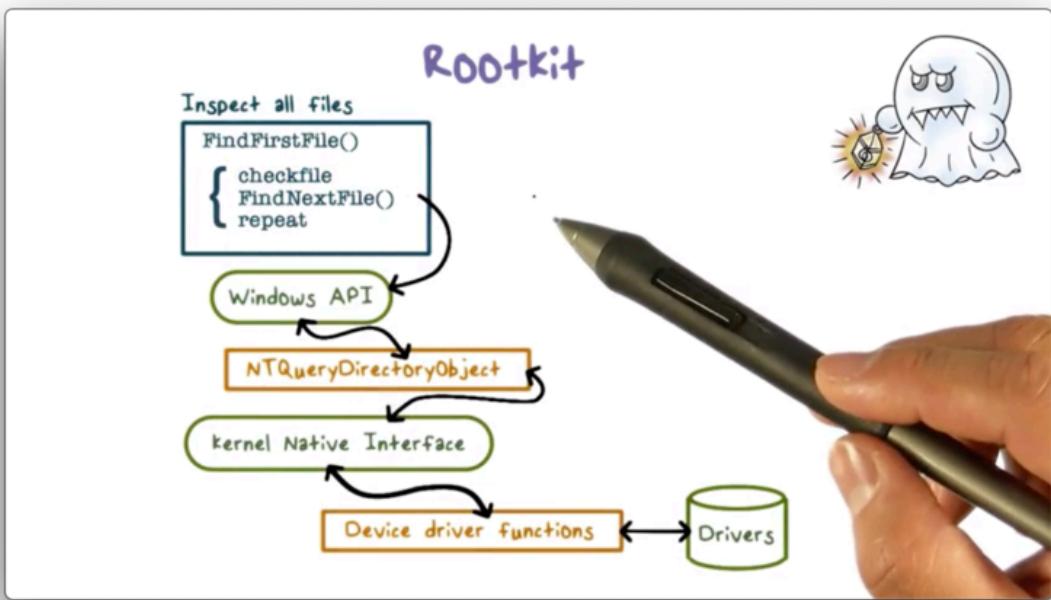
A **rootkit** is a program that typically modifies some of the code and data structures in an operating system in order to perform some malicious activities.

A rootkit can be used to hide malware from a user.

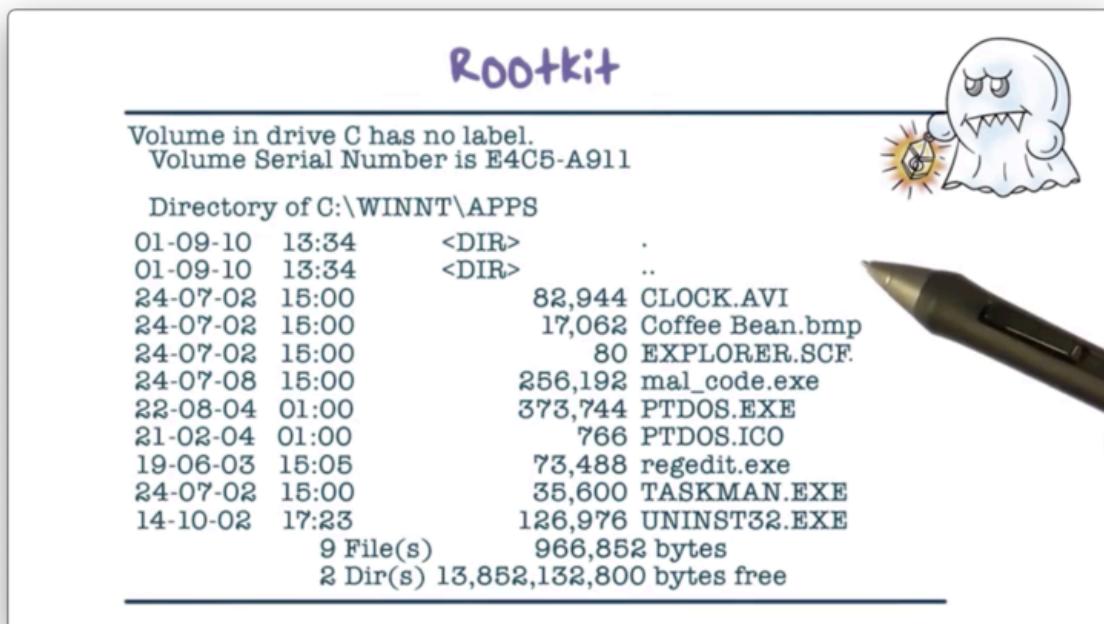
For example, when the user executes the `ls` command to list the contents of a directory, the rootkit can change the output of this command so that the user will not see the malware file.

Similarly, when the user executes the `ps` command to see what programs are running on the system, the rootkit can change the output of this command to hide the running malware.

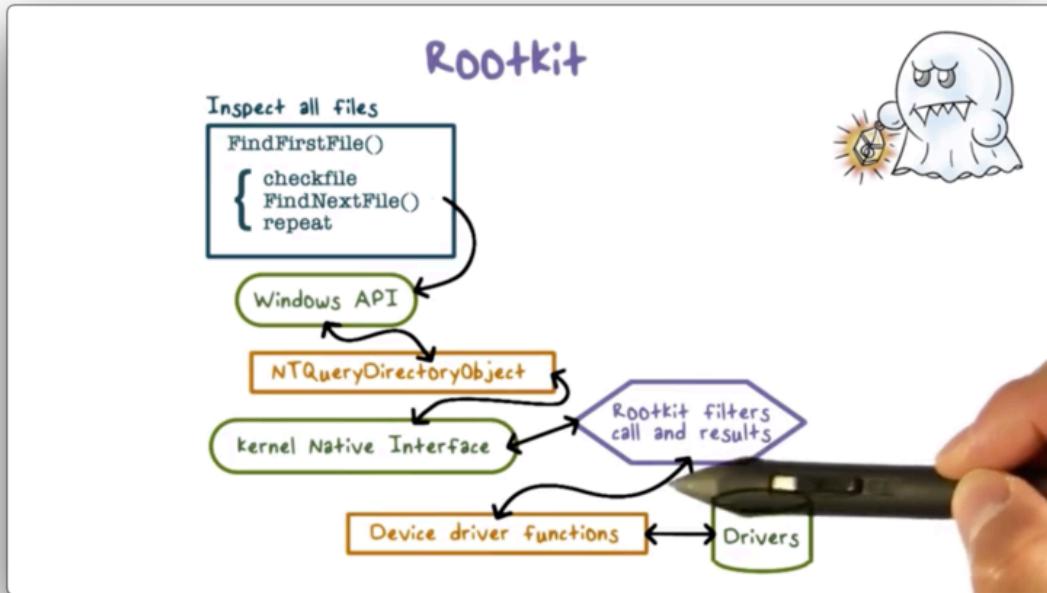
Here is the functional flow for listing files using the windows command `dir`.



Before the rootkit has been embedded, here is what an operating system will typically return when the user lists the files in a directory.



The installed rootkit intercepts any call to the operating system and then determines whether the call will reveal the malware. If so, the rootkit alters the result to hide the malware; otherwise, it passes the result to the caller unmodified.



For example, the rootkit can intercept the directory listing from the `dir` command and remove the malware entry.

## Rootkit



Volume in drive C has no label.  
Volume Serial Number is E4C5-A911

Directory of C:\WINNT\APPS

01-09-10 13:29	<DIR>	.
01-09-10 13:29	<DIR>	..
24-07-02 15:00		82,944 CLOCK.AVI
24-07-02 15:00		17,062 Coffee Bean.bmp
24-07-02 15:00		80 EXPLORER.SCF
22-08-04 01:00		373,744 PTDOS.EXE
21-02-04 01:00		766 PTDOS.ICO
19-06-03 15:05		73,488 regedit.exe
24-07-02 15:00		35,600 TASKMAN.EXE
14-10-02 17:23		126,976 UNINST32.EXE
		8 File(s) 710,660 bytes
		2 Dir(s) 13,853,472,768 bytes free

### 8.19 Rootkit Quiz



## Rootkit Quiz



Which operating systems can be affected by Rootkit?

- Linux
- iOS
- Windows
- Android

## 8.20 Rootkit Quiz Solution



### Rootkit Quiz

Which operating systems can be affected by Rootkit?

- Linux
- iOS
- Windows
- Android



### 8.21 Truth and Misconceptions Quiz



#### Truth and Misconceptions about Malicious Software Quiz

Put a 'T' in the box for any statement you think is true and an 'F' for any statement you think is false.

- Can only infect Microsoft Windows
- Can modify hidden and read-only files
- Spread only on disks or in email
- Cannot remain in memory after reboot
- Cannot infect hardware
- Can be malevolent, benign, or benevolent

## 8.22 Truth and Misconceptions Quiz Solution



### Truth and Misconceptions about Malicious Software Quiz

Put a 'T' in the box for any statement you think is true and an 'F' for any statement you think is false.

<input checked="" type="checkbox"/> F	Can only infect Microsoft Windows
<input checked="" type="checkbox"/> T	Can modify hidden and read-only files
<input checked="" type="checkbox"/> F	Spread only on disks or in email
<input checked="" type="checkbox"/> T	Cannot remain in memory after reboot
<input checked="" type="checkbox"/> T	Cannot infect hardware
<input checked="" type="checkbox"/> F	Can be malevolent, benign, or benevolent



## 8.23 Worms

Worms are independent malicious programs that typically use network connections to spread from one system to another.

Worms first appeared in the 1990s - coinciding with the rapid expansion of the Internet - and marked a major advance in malware.

Worms later evolved into botnets around 2005, which remain the dominant form of malware we see today.

## 8.24 The Internet Worm

One of the most famous worms is the **Morris Worm**, named after its creator Robert Morris. This worm is also referred to as the *Internet Worm*.

According to Morris, this worm was used to gauge the size of the Internet by measuring how many computers were connected together.

### 8.24.1 Programming Error

However, there was a programming error in the code which transformed this intellectual exercise into a damaging attack.

The worm would ask a target system if it had been infected. Even if the target responded that it had, the worm would infect the system again, one out of seven times.

This reinfection rate proved too aggressive, and many systems that had been infected multiple times crashed.

Sysadmins had to disconnect their servers from the Internet in order to disinfect them. Since many servers were infected, Internet service overall was disrupted.

### 8.24.2 Implementation

When the Morris Worm identified the next target to infect, it looked for several security flaws that it knew how to exploit.

From [court documents](#):

Morris identified four ways in which the worm could break into computers on the network: (1) through a “hole” or “bug” (an error) in SEND MAIL, a computer program that transfers and receives electronic mail on a computer; (2) through a bug in the “finger demon” program, a program that permits a person to obtain limited information about the users of another computer; (3) through the “trusted hosts” feature, which permits a user with certain privileges on one computer to have equivalent privileges on another computer without using a password; and (4) through a program of password guessing, whereby various combinations of letters are tried out in rapid sequence in the hope that one will be an authorized user’s password, which is entered to permit whatever level of activity that user is authorized to perform.

Specifically, the `fingerd` daemon was exploited via a buffer overflow vulnerability, and `sendmail` was exploited via a trapdoor that accepted shell commands.

Once the worm gained access to a target system, it then loaded a small piece of code - the bootstrap loader - which fetched the rest of the worm code. Password-based authentication was used to ensure that only the bootstrap loader could fetch the worm code.

The worm employed a number of tricks to hide itself.

For example, once the worm code was loaded into memory, it deleted its original file from disk and encrypted/decrypted itself as necessary. In addition, the worm even periodically changed its process name and ID, so a sysadmin looking at running programs could not easily discover the worm.

### 8.24.3 What We Learned

Most of the flaws exploited by the Morris Worm were not only well known at the time but also had security patches available. The worm taught us the importance of frequent security scanning and patching.

We also learned that we need to have a fast and coordinated response to a major security incident like the Morris Worm. The US government established the [Computer Emergency Response Team](#) (CERT), which is responsible for issuing alerts about security flaws and recommendations about patches.

### 8.25 Worm Quiz

 **Worm Quiz**

Which of the following methods can be used to spread a worm? Check all that apply:



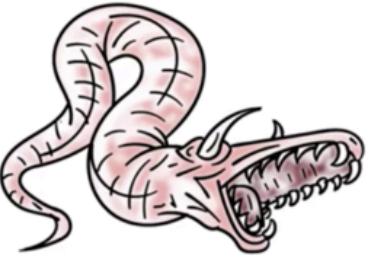
- email
- instant messaging
- downloading files
- watching a video on netflix
- clicking on a popup
- using facebook

## 8.26 Worm Quiz Solution



### Worm Quiz

Which of the following methods can be used to spread a worm? Check all that apply:



- email
- instant messaging
- downloading files
- watching a video on netflix
- clicking on a popup
- using facebook



## 8.27 Malware Prevention & Detection Approaches

We can prevent malware from infecting our computer systems by limiting our contact with the outside world. In other words, if we don't accept documents or programs from any external source, we can greatly reduce our chance of infection.

Obviously, this approach imposes a major inconvenience to a computer user.

Instead of preventing infection, we can try to improve how we detect an infection. We can continually improve the monitors that we use to watch for telltale signs of infection.

In addition, we can focus on removal. Once we detect a malware infection, we remove the malware and potentially also patch the system.

Given that prevention severely hampers productivity, detection is the main countermeasure that we use.

There are 4 generations of antivirus software, each with different strategies for detecting malware infection.

### 8.27.1 Simple Scanners

**Simple malware scanners** scan program files looking for signatures (patterns) of known viruses. If a scanner finds a signature, it means that the program file has been infected by a known virus.

A **virus signature** is typically a unique sequence of instructions in the virus code or the unique infection marker that the virus would use.

These simple scanners are not effective against polymorphic viruses because each instance of a polymorphic virus is encrypted with a random key, meaning that there is no unique signature across all instances of the same virus.

### 8.27.2 Heuristic Scanners

**Heuristic scanners** look for the possible effects of infection. For example, if a program file has been infected, the [checksum](#) of the file will change because the file contents have changed.

This approach can be defeated if the malware deliberately ensures that the checksum after infection remains the same, which can be done by including some additional bytes at the end of the file.

### 8.27.3 Activity Traps

**Activity traps** monitor the system by watching for particular kinds of activities that malware would usually perform on a system, such as reading a password file and sending it over the Internet.

These detectors are based on our knowledge of malware activities. Therefore, they are not effective against malware that performs new kinds of malicious activities.

### 8.27.4 Full-Featured Analysis

A full-featured analysis typically involves multiple detection approaches, such as host-based monitoring - which includes activity traps and scanners - and network-based monitoring, which examines network traffic to detect suspicious activity.

This strategy can also include a sandbox-based analysis. A **sandbox** is typically a secure, observable environment on the system that is isolated from the rest of the system.

We can observe a suspicious executable in a sandbox first to ensure that it will not cause any damage to the system or network before allowing it to run outside of the sandbox.

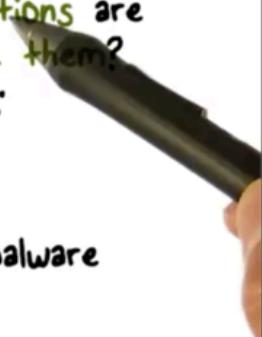
### 8.28 Malware Prevention & Detection Quiz



#### Malware Prevention & Detection Quiz

Given that signature-based anti-virus solutions are not always effective, why do we still use them?  
Mark the statements either true or false:

- they are very efficient .
- effective against known malware
- good "first-line" defense



### 8.29 Malware Prevention & Detection Quiz Solution



#### Malware Prevention & Detection Quiz

Given that signature-based anti-virus solutions are not always effective, why do we still use them?  
Mark the statements either true or false:

- T they are very efficient
- T effective against known malware
- T good "first-line" defense



### 8.30 Most Expensive Worm Quiz



#### The Most Expensive Worm Quiz

Which of the worms described below caused the greatest financial damage?

- ILOVEYOU:** Sent by email with the subject "ILOVEYOU". It had an attachment that, when executed, deleted all files on the host computer.
- CODE RED:** a worm that took advantage of a buffer overflow vulnerability in Microsoft servers. Infected machines would launch 'denial of service' on IP addresses.
- Morris Worm:** 99 lines of code that Robert Morris, a Cornell student launched to find out the size of the internet.

### 8.31 Most Expensive Worm Quiz Solution



## The Most Expensive Worm Quiz

Which of the worms described below caused the greatest financial damage?

- ILOVEYOU:** Sent by email with the subject "ILOVEYOU". It had an attachment that, when executed, deleted all files on the host computer.
- CODE RED:** a worm that took advantage of a buffer overflow vulnerability in Microsoft servers. Infected machines would launch 'denial of service' on IP addresses.
- Morris Worm:** 99 lines of code that Robert Morris, a Cornell student launched to find out the size of the internet.



[Source](#)

## 9 Modern Malware

### 9.1 Past Malware

Before 2005, most malware was used for “fun” and/or “fame”.

During this era, malware was often designed for the sake of experimentation, and was used to demonstrate some capability - for example, how fast a piece of malware could spread.

Only a few instances of malware in this time period were actually used for denial-of-service attacks and website defacements.

### 9.2 Modern Malware

Since 2005, malware has been used to make compromised computers and networks perform malicious activities in order to elicit financial and even geopolitical gain.

Whereas computers of the past were targets of malware, they are now weapons that malware can control and deploy for profit and gain.

Since modern malware are now used to do real work, they tend to be technically sophisticated and use the latest technologies.

For example, they may utilize popular peer-to-peer protocols and applications to set up communications. They might use cloud computer servers to support their activities. Some malware might use the latest cryptography algorithms to perform authentication and encryption to protect their communications from subversion and analysis.

### 9.3 Botnet

A **bot**, also called a zombie, is a compromised computer under the control of an attacker.

The bot code present on the compromised system is responsible for communicating with the attacker's server and carrying out malicious activities per the instructions it receives from this server.

Therefore, a **botnet** is a network of bots controlled by an attacker that is used to perform coordinated malicious activities.

With a network of bots, the aggregated computational power can be very large. An attacker can launch a variety of attacks using such a powerful platform.

Indeed, most Internet-based cyber attacks today are carried out by botnets.

#### 9.4 Bot Quiz



#### Bot Quiz

Match the bot with its definition by putting the correct letter in the box.

- |                                      |  |
|--------------------------------------|--|
| <input type="checkbox"/> Spamming    | A. Used by botmasters to fraudulently increase revenue from advertisers. |
| <input type="checkbox"/> Click Fraud | B. Used to gather valuable financial information.                        |
| <input type="checkbox"/> Phishing    | C. Infected machines send out unsolicited emails.                        |

## 9.5 Bot Quiz Solution



### Bot Quiz

Match the bot with its definition by putting the correct letter in the box.

C Spамминг

A. Used by botmasters to fraudulently increase revenue from advertisers.

A Click Fraud

B. Used to gather valuable financial information.

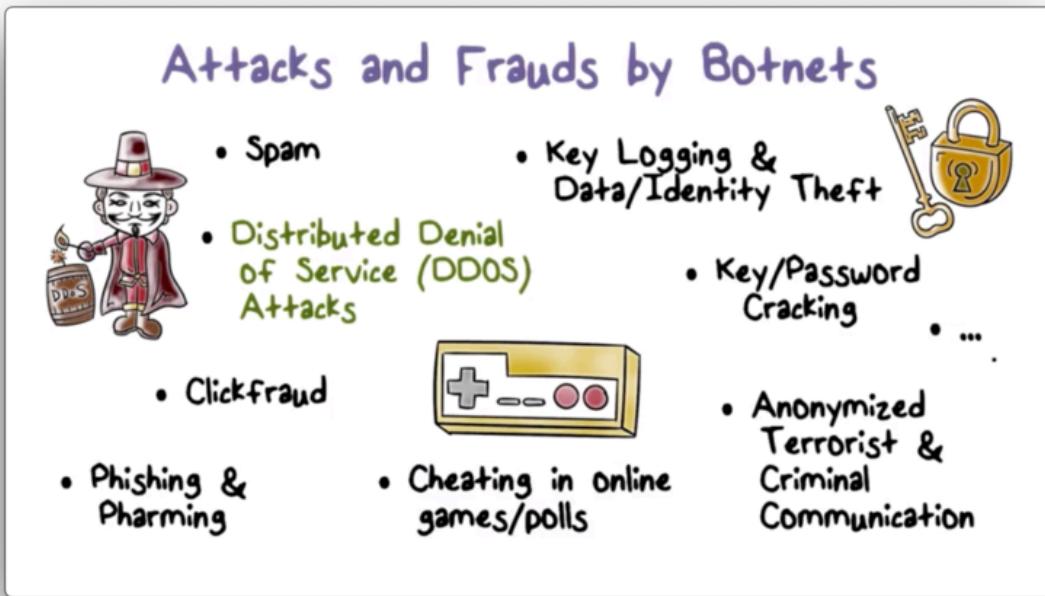
B Phishing

C. Infected machines send out unsolicited emails.



## 9.6 Attacks and Frauds by Botnets

Regardless of the method used, modern botnets usually have one or two goals in mind: illegal profits or political activism. Here are some examples of attacks and frauds by botnets.



### 9.7 DDoS Using Botnets

Let's look at a typical DDoS scenario using botnets.

First, the attacker selects a victim and decides when to attack. Next, the attacker sends a command to all of the bots in the botnet. This command might tell the bots to all send connection requests to the victim at once.

The result is that the victim receives connection requests from many bots at the same time. As a result, the victim is overloaded, and the denial of service is complete.

### 9.8 Amplification Distributed Reflective Attacks

A typical defense against DDoS is to buy more servers or bandwidth. However, DDoS attacks can be amplified to make this type of defense very expensive.

The following is an example of such an **amplification attack**.

On the Internet, there are many *open recursive DNS servers* that any Internet-connected machine can query. A typical DNS query asks for the IP address associated with a particular domain name.

Users can also query these servers for the TXT record for a domain. This record often contains a lot of information, and the size of the query response can be more than 1500 bytes.

Attackers will instruct their bots to query these servers for this TXT record. They will **spoof** (forge) the query such that the source IP address will point to the victim's IP address. As a result, the response will be sent to the victim, not the bot.

The attacker thus amplifies their query traffic - at around 60 bytes per query - into a much larger amount of attack traffic - at 1500+ bytes per response - directed at the victim.

With just a few thousand bots, querying multiple DNS servers, the attacker can send several gigabytes of traffic to the victim.

## 9.9 DDoS Quiz



### DDoS Quiz

Put a check next to each of the following statements about DDoS that are true.

- The attacker does not have to use his own computer in the attack.
- Since there are so many computers involved in the attack it is difficult to distinguish legitimate from malicious traffic.
- The characteristics of DNS servers help mitigate the effect of DDoS attacks.

## 9.10 DDoS Quiz Solution



### DDoS Quiz

Put a check next to each of the following statements about DDoS that are true.

- T The attacker does not have to use his own computer in the attack.
- T Since there are so many computers involved in the attack it is difficult to distinguish legitimate from malicious traffic.
- F The characteristics of DNS servers help mitigate the effect of DDoS attacks.



Remember, the characteristics of DNS servers can be used to amplify the effects of DDoS attacks, not mitigate them.

## 9.11 Botnet Command and Control

In order for a botmaster to make use of a botnet, he needs to communicate with the bots. In particular, there needs to be **command & control** (C&C) from the botmaster to the bots.

For example, a bot should be able to report its current status to the botmaster. A botmaster should be able to direct a bot to download the latest version of the bot code as well as instruct the bot to perform an attack.

Without C&C, a botnet is just a collection of isolated infected machines which the botmaster will not be able to aggregate into a single computational workhorse.

## 9.12 Botnet C&C Problem

Suppose we have downloaded some malware source code from the web, and we've configured it to our liking. We've used some social engineering to start spreading out our compiled malware via

email.

As our code spreads, we as botmasters are faced with an important question: How can we identify and contact the infected machines so we can start to use them?

The simplest solution is to have the infected victims contact us.

### **9.13 Botnet C&C: Naive Approach**

We may hardcode our email address or our IP address in the malware to give the compromised computers a point of contact upon infection.

This approach is not stealthy. It is a safe bet that security admins will eventually find out that they have bots on their network. When they do, they may be able to obtain your bot code and recover the hardcoded address through malware analysis. With this address, they will be able to identify us.

This approach is also not robust. The single rallying point hardcoded in the malware also represents a single point of failure. For example, if we have hardcoded an email address and the email account has been banned, our command and control center becomes completely unreachable.

### **9.14 Botnet C&C Design**

Since having the compromised computers contact the botmaster is neither stealthy nor robust, this strategy is only used by [script-kiddies](#) and first time malware authors.

A botmaster will focus on efficiency and reliability when designing bot code. They must be able to coordinate enough bots to perform a specific task, which requires efficient, reliable communication.

In addition, the botmaster will design for stealth and robustness. Stealth makes it hard to detect C&C traffic, while robustness makes it hard to disable or block this traffic.

### 9.15 C&C Design Quiz



#### C&C Design Quiz

Mark the following statements as either true (T) or false (F):

- Bots have more sophisticated communication capabilities than worms and viruses
- Bots require direct communication with the C&C server before beginning an attack
- A botnet will be less likely to be found if it uses custom communication protocols

## 9.16 C&C Design Quiz Solution



### C&C Design Quiz

Mark the following statements as either true (T) or false (F):

- T Bots have more sophisticated communication capabilities than worms and viruses
- F Bots require direct communication with the C&C server before beginning an attack
- F A botnet will be less likely to be found if it uses custom communication protocols



The second answer is false. Bot code can have logic bombs or other triggers that enable bots to attack without contacting a C&C server.

The third answer is also false. A botnet is more likely to be found using custom communication protocols, as admins observing the network are more likely to detect strange types of traffic flowing from their system.

## 9.17 DNS Based Botnet C&C

Many botnets use DNS for C&C.

A key advantage is that DNS is used whenever a machine on the Internet needs to talk to another machine because DNS stores the mapping between domain names and IP addresses. DNS is always allowed in a network, so DNS traffic will not stand out.

The botmaster will distribute the malware code, inside of which the domain name of the C&C server is hardcoded. Let's assume that this domain name is `hackers.com`.

To perform C&C, the bot will ask the DNS server for the IP address for `hackers.com`. The DNS server will tell the bots what the IP address is, and the bots will use that address for communication.

Botmasters prefer *dynamic* DNS providers because they allow for frequent changes between domain names and IP addresses. This means that the botmaster can rapidly switch where they host their server, and can quickly change the mapping so that `hackers.com` points to the new address.

If we can detect that `hackers.com` is used for botnet C&C, then we can identify any computer that connects to it as a bot.

How can we determine that a domain name is used for C&C?

The way that bots look up a domain will be different from a machine that looks up a web server because of legitimate user activities.

For example, if a domain is being looked up by hundreds of thousands of machines all over the Internet, and yet this domain is unknown to Google search, this is an anomaly.

We can use anomaly detection at the dynamic DNS provider level to examine domain queries in order to identify potential botnet C&C servers.

Once we identify that `hackers.com` is used for botnet C&C, there are a number of responses available.

The DNS provider can point the entry for `hackers.com` to the IP address of a sinkhole. The result of this change is that bot traffic will be routed to the sinkhole instead of the C&C server, effectively severing the communication link between the botmaster and the bots.

In addition, security researchers can monitor the sinkhole and, by looking at the IP address of each incoming connection, can determine the distribution of the bots throughout the Internet.

### 9.18 Botnet C&C Quiz



#### Botnet C&C Quiz

Which of the following C&C schemes provide:

- Efficient/reliable communications
- Stealth communications (hard to detect)
- Resilient communications (hard to disrupt)

Check all that apply:

- A Gmail account is used for C&C, email address hardcoded in botcode
- P2P protocol is used for C&C, query string is hardcoded in botcode
- A "news" web site has been set up for C&C, i.e. commands can be "parsed" from news articles. Website and parsing logic hardcoded in botcode.



### 9.19 Botnet C&C Quiz Solution



### Botnet C&C Quiz

Which of the following C&C schemes provide:

- Efficient/reliable communications
- Stealth communications (hard to detect)
- Resilient communications (hard to disrupt)

Check all that apply:

- A Gmail account is used for C&C, email address hardcoded in botcode
- P2P protocol is used for C&C, query string is hardcoded in botcode
- A "news" web site has been set up for C&C, i.e. commands can be "parsed" from news articles. Website and parsing logic hardcoded in botcode.



A single gmail account, hardcoded in bot code, is both easy to detect and easy to disrupt.

P2P traffic will easily stand out in an enterprise network where peer-to-peer communications are not typically allowed.

A news site can be hard to detect, because traffic to news websites is common. However, if the site is identified as being malicious, it can easily be blocked.

### 9.20 Advanced Persistent Threat

The latest type of modern malware is the **advanced persistent threat** (APT). Whereas botnets often have bots all over the Internet, APTs tend to be localized to specific target organizations.

#### 9.20.1 Advanced

APTs can use advanced malware that is specifically crafted for a targeted organization.

Often, the malware used is just a customized version of a common malware. Starting from a common malware provides APT designers with both convenience and deniability; that is, security admins in the target organization will not suspect that a common-looking malware is actually an APT.

The operations carried out by APTs are also advanced.

APTs are not used for common attacks/frauds like spamming, click fraud, phishing. An APT is much more likely to be used for a high value operation, such as stealing the design of a new airplane.

### **9.20.2 Persistent**

Once the malware gets into an organization, it will stay there for a long time, carrying out its attack in a “low and slow” fashion.

For example, rather than sending out the design of the airplane - a large amount of data - onto the Internet at once, the APT can break the data into multiple chunks and then transmit each chunk whenever a user connects to the Internet.

Of course, this approach exfiltrates the designs much more slowly, but can do so without raising any suspicion.

### **9.20.3 Threat**

APTs are a very dangerous threat because they tend to target high value organizations and information.

## **9.21 APT Lifecycle**

The attacker starts by identifying the attack target, such as an automobile or aerospace company.

Next, the attacker will research the target, looking for information that can help in the attack.

For example, the attacker can learn about the target organization’s network and therefore the vulnerabilities of the organization’s network services through network scanning and analysis.

Many companies list their C-level executives on their webpages, and attackers can research these individuals to find their company email addresses, which they can use as a contact point.

Once they have completed their research, the attacker can develop a method to penetrate the organization’s network.

For example, the attacker may exploit a [zero-day](#) vulnerability in the company’s web server.

Alternatively, they may choose to engage in a so-called **spear phishing** attack. This type of phishing attack is usually directed to a C-level executive within the organization, and is called spear phishing because it is aimed at a single individual that the attacker has spent a lot of time researching.

The exploit succeeds once the APT has gained a foothold within the organization and establishes communication back to the attacker.

Once the communication link has been created, the attacker can push software updates and commands to the APT, such as requesting that it exfiltrate confidential data back to the attacker.

Finally, the APT will try not to raise any suspicion so that it can remain undetected.

For example, it will only exfiltrate data to the attacker when there are other legitimate network connections. In addition, it will keep its footprint as small as possible, only infecting the machines that it needs for its tasks.

## 9.22 APT Characteristics

The most dangerous and advanced APTs are those that use a **zero-day exploit** or specially crafted malware.

A zero-day attack exploits a previously unknown vulnerability. Since the vulnerability is unknown at the time of the attack, intrusion detection systems will likely not have a signature for the exploit and, even if they did, there is no patch available.

Zero-day exploits can be very successful and very stealthy.

Similarly, a specially crafted malware is usually designed to defeat the signature and behavior monitors in a particular detection system. Like zero-day exploits, this class of malware has a very high chance of succeeding and proceeding undetected.

APTs are characterized not only by their ability to thwart anti-malware software, but also by their ability to employ social engineering strategies to fool even the most sophisticated users.

For example, an APT might start out by passively monitoring email traffic in order to understand who speaks to whom, what certain people talk about, and what type of attachments are sent. With this knowledge, the APT can then successfully forge email from one person to another.

In addition, the APT can conduct **man-in-the-middle** attacks to make such social engineering strategies very successful.

For example, if an email recipient is uncertain about an email attachment, they may send an email out to the sender asking for clarification. The APT can intercept this message and respond as the “sender”, confirming the message and the (malicious) attachment.

APTs are also designed to blend in with normal activities to avoid detection, achieving their goals in a “low and slow” fashion.

For example, if the APT is designed to change the setting of a controller in a nuclear plant, it will not make the changes all at once. Instead, it will make small incremental changes over time to accomplish the eventual attack goal. This was the strategy of the famous [Stuxnet](#) malware.

Because APTs are designed to blend in with normal system activities, it can be very hard for anomaly detection systems to catch them.

APTs are often designed to stay in the compromised organization for a long time, always looking for more valuable data to steal.

An APT can carry out many different attacks, on different users, in different parts of the system. At each step, the APT may use different malware code, and it may clean up after itself as it travels throughout its host.

In other words, the footprint of an APT often remains very small and ever-changing.

### 9.23 APT Quiz



### APT Quiz

Match the items below by writing the corresponding letter in the boxes:

<input type="checkbox"/> Boy in the Browser <input type="checkbox"/> Clickjacking <input type="checkbox"/> Man in the Browser <input type="checkbox"/> Man in the Middle <input type="checkbox"/> Keyloggers	<ul style="list-style-type: none"><li>A. Eavesdrops</li><li>B. Modifies web pages covertly</li><li>C. Covertly records keystrokes</li><li>D. Covertly changes a computer's network routing</li><li>E. Web users unknowingly click on something that is not as it is portrayed</li></ul>
--	---

## 9.24 APT Quiz Solution



### APT Quiz

Match the items below by writing the corresponding letter in the boxes:

<p><input type="checkbox"/> D Boy in the Browser</p> <p><input type="checkbox"/> E Clickjacking</p> <p><input type="checkbox"/> B Man in the Browser</p> <p><input type="checkbox"/> A Man in the Middle</p> <p><input type="checkbox"/> C Keyloggers</p>	<p>A. Eavesdrops</p> <p>B. Modifies web pages covertly</p> <p>C. Covertly records keystrokes</p> <p>D. Covertly changes a computer's network routing</p> <p>E. Web users unknowingly click on something that is not as it is portrayed</p>
---	--

## 9.25 APT Example

A CEO gets an email with a PDF attachment containing pie charts of sales activities. When the CEO opens the PDF using the adobe acrobat browser plugin, the crafted attack data embedded in the PDF document causes a zero-day exploit that breaks out of the plugin sandbox and compromises the browser.

The attack code downloads a malicious browser extension - a malware embedded within the browser. From this point on, the extension will infect every attachment the CEO sends via email.

Eventually, the malware gets on the computer of a user who has admin access to the company's server. The malware can now steal the server credentials from the user and thus, the valuable data residing on the server.

This example captures several key characteristics of APTs. First, the users do not realize that their computers and network have been compromised. Second, the APT activity blends in with normal user activity. For example, the APT doesn't send its own emails; instead, it only modifies emails sent by the CEO. Third, the APT takes its time to get to the key individual and steal the server credentials.

## 9.26 Malware Analysis

Malware analysis produces information about malware that can be used for detection and response. There are two typical approaches to malware analysis.

In **static analysis**, we look at the program or the instruction set of the malware to understand what the malware would do if it was executed. We want to understand the malware's behavior without actually executing it.

There are limitations to static analysis. Some program behaviors that depend on runtime conditions or user input data can not be precisely identified by looking at the source code. Additionally, binary code can be obfuscated.

Another approach is **dynamic analysis**. Here, we run the malware program and try to analyze its runtime behavior to try to understand what the malware is doing when it is executed.

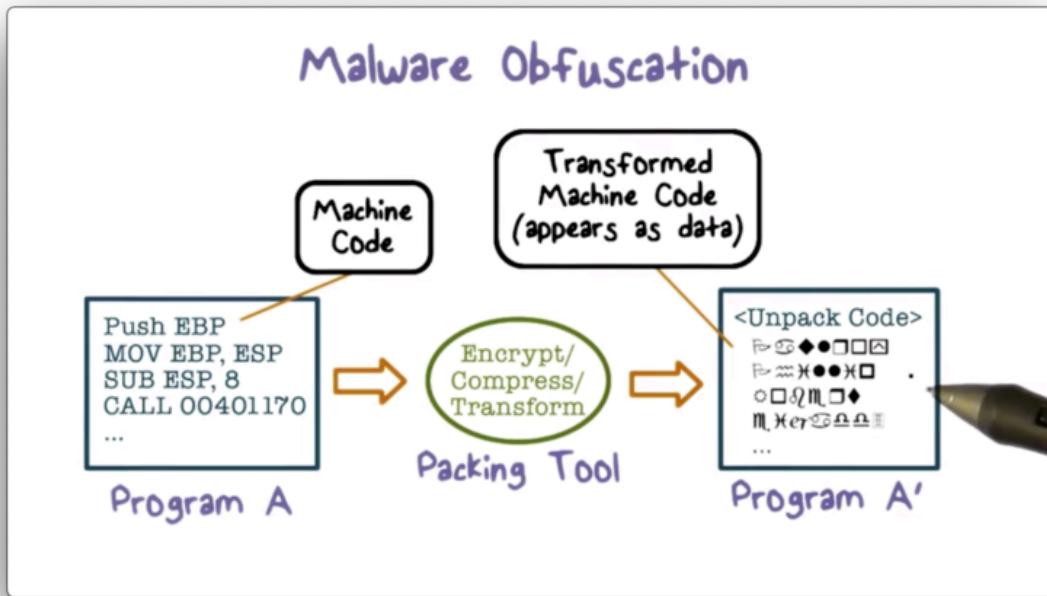
We can perform dynamic analysis in different levels of granularity. A fine-grained analysis might look at the malware execution on an instruction by instruction basis. For a more coarse-grained approach, we might only look at the system calls that the malware invokes.

Dynamic analysis is not without its limitations. Any particular run of the malware only reveals behaviors of that run. Malware can actively try to resist analysis by delaying execution until a certain command is given or a trigger is tripped. For any run, the conditions may not be right for the malware to exhibit certain behaviors, and so those behaviors are not revealed.

Because of the limitations of each type of analysis, a typical malware analysis system will employ both static and dynamic analysis.

## 9.27 Malware Obfuscation

**Packing** refers to the process of compressing and encrypting part of an executable program. The result of packing is that part of the executable becomes data instead of instructions. To execute the packed instructions, the packing tool must include code in the packed executable that **unpacks** these instructions.



The packing program will encrypt the malware using a randomly generated key, which means that each subsequent packed malware will look completely different from the last. Consequently, a signature-based approach is not effective in detecting packed malware.

Can we use the unpacking instructions as a signature to detect malware? Unfortunately not. Malware is not the only application that performs packing/unpacking. Legitimate programs may use packing to hide certain logic and/or data, often for intellectual property reasons such as copyright protection.

## 9.28 Unpacking

Most modern malware uses packing, and there are many **packers** available. Furthermore, there are hundreds of thousands of packed malware samples released to the Internet every day by attackers.

We cannot manually analyze all of the malware samples. Instead, we need an automatic approach to unpack and analyze malware. Such an approach has to be universally applicable to all of the malware samples.

We can run static analysis over the packed malware to get a set of instructions in the packed program. We can then run the malware and use dynamic analysis to identify instructions that are not present in the statically identified set of instructions. These instructions must have been unpacked just before execution.

Once we identify the unpacked code, we can then apply other anti-malware techniques such as signature scanning to identify the malware logic. Since the unpacked malware code looks the same across malware instances, we can still use a signature-based approach to identify these infections.

### 9.29 Malware Analysis Quiz



### Malware Analysis Quiz

What approach(es) can be used to detect the example APT malware (the malicious browser extension)?

- A network monitor that analyzes traffic to detect anomalies or known bad traffic (e.g., to known bad domains)
- A host monitor that examines operating systems activities (e.g., access to files)
- A malware analysis system that identifies malicious logic (e.g., running the browser in a sandbox & tracing its execution)



### 9.30 Malware Analysis Quiz Solution



### Malware Analysis Quiz

What approach(es) can be used to detect the example APT malware (the malicious browser extension)?

- A network monitor that analyzes traffic to detect anomalies or known bad traffic (e.g., to known bad domains)
- A host monitor that examines operating systems activities (e.g., access to files)
- A malware analysis system that identifies malicious logic (e.g., running the browser in a sandbox & tracing its execution)



## 10 Firewalls

### 10.1 Defense in Depth

When it comes to defense against attacks, the most important principle is to employ **defense in depth**. In other words, we should utilize multiple layers of defense mechanisms.

The first line of defense is prevention: prevention stops attacks from getting into our networks and systems in the first place. Inevitably though, some attacks will defeat the prevention mechanism. For example, anti-virus software may not prevent a user from downloading a Trojan horse, since Trojan horses appear to be legitimate programs.

Detection and response mechanisms make up the second line of defense. These mechanisms monitor activities in our systems and networks to detect attacks and repair damage. Just like with prevention, there will be attacks that can go undetected, at least until they have inflicted their damage. For example, the APT malware examples we discussed previously are often hard to detect because they blend in with normal activities so well.

The third line of defense consists of the attack-resilient technologies that enable the most valuable

system components to survive attacks. For example, a server is a collection of diversified subsystems with different implementations. At least one of these subsystems will not be susceptible to an attack since an attack typically exploits specific vulnerabilities that may be present on some, but not all, components.

## 10.2 What is a Firewall?

To motivate the need for firewalls, let's first take a high-level tour of a typical enterprise network.

An enterprise network is part of the Internet. It typically has an internal, trusted intranet that only the employees can access. At a bank, for example, the trusted part of the network holds the internal email server and the systems that process financial transactions.

The enterprise network can also have a public-facing part. A bank may have a web portal where customers can log in and view their account. These public servers live in the so-called **demilitarized zone** (DMZ), which is a part of the enterprise network that is separated from the trusted network. While customers can interact with the webserver to log in and submit transaction requests, they cannot directly access the servers in the trusted network that are authorizing and processing transactions.

When a company has multiple physical sites, each site can have its own local, trusted network. However, these sites will need to communicate with one another. For example, employees in one bank branch may need to access the corporate network at the headquarters in another city. Such access must take place using the untrusted Internet.

We use routers to get traffic to its destination correctly across the Internet. Each local network has at least one router at its perimeter and these routers together with the core routers in the Internet backbone transport packets from one local area network to another.

While routers can send this traffic, the network needs to decide whether it should allow such traffic, based on security considerations. For example, the bank may want to explicitly disallow any external traffic from reaching the trusted network directly.

A **firewall** is a device that provides secure connectivity between networks. It is used to implement and enforce a security policy for communication between the networks.

### 10.3 Firewalls Quiz



#### Firewalls Quiz

Mark the box next to all those items that firewalls **can stop**:

- Hackers breaking into your system
- Internet traffic that appears to be from a legitimate source
- Viruses and worms that spread through the internet
- Spyware being put on your system
- Viruses and worms that are spread through email



## 10.4 Firewalls Quiz Solution



### Firewalls Quiz

Mark the box next to all those items that firewalls **can stop**:

- Hackers breaking into your system
- Internet traffic that appears to be from a legitimate source
- Viruses and worms that spread through the internet
- Spyware being put on your system
- Viruses and worms that are spread through email

## 10.5 Firewall Design Goals

A firewall is designed to enforce security policies on inbound and outbound network traffic. All traffic must be checked at the firewall, and only traffic that has been authorized by the security policy is allowed to pass through.

In addition to correctly enforcing the security policies, a firewall must also be dependable. This means that the firewall must not be easily crashed or disabled by an attack. A disabled firewall can no longer enforce the security policies of the network.

## 10.6 Firewall Access Policy

A critical component in the planning and implementation of a firewall is defining a suitable access policy. A network access policy specifies the types of traffic that are authorized to pass through the firewall. Traffic can be authorized using many different criteria, including address ranges, protocols, applications, and content types.

Policies should be developed through a security risk assessment conducted by the organization. This

assessment will reveal what types of traffic the organization must support and what security risks are associated with this traffic, which will inform the firewall implementation.

## 10.7 Firewall Limitations

A firewall cannot provide protection against traffic that does not flow through it, such as traffic that is routed around it or traffic that is internal to the network. Additionally, if a firewall is misconfigured, it may not properly enforce the security policy against the traffic that it sees.

## 10.8 Additional Convenient Firewall Features

A firewall can log all traffic that passes through it, and the log can be analyzed later to learn more about the traffic, such as the traffic volume to a specific part of the network.

Firewalls can also provide **network address translation**. This is useful when multiple machines in the internal network have to share a single public IP address. For outbound traffic leaving the network, the firewall rewrites the local source IP address of a packet coming from an internal host to the public IP address. For inbound traffic, the firewall translates the destination IP address of the packet from the public IP address to the local IP address of the internal host.

A firewall can also provide encryption services. Two trusted networks may choose to send encrypted traffic to one another as a way to ensure that their communication cannot be eavesdropped on by other untrusted networks on the Internet.

### 10.9 Firewall Features Quiz



### Firewalls Features Quiz

Mark all the answers that apply:

Malware can disable:

- Software firewalls
- Hardware firewalls
- Antivirus checkers

Firewalls can stop/control:

- Pings
- Packet Sniffing
- Outbound network traffic

### 10.10 Firewall Features Quiz Solution



## Firewalls Features Quiz

Mark all the answers that apply:

**Malware can disable:**

- Software firewalls
- Hardware firewalls
- Antivirus checkers

**Firewalls can stop/control:**

- Pings
- Packet Sniffing
- Outbound network traffic



### 10.11 Firewalls and Filtering

The main mechanism in firewalls is traffic filtering. The firewall stops each packet - whether inbound or outbound - and checks it against the security policy. After performing the check, the firewall will decide whether to allow or discard the packet.

### 10.12 Filtering Types

There are two main types of filtering: packet filtering and session filtering. In packet filtering, the firewall examines each packet by itself, and forwards or rejects the packet based on its attributes. In session filtering, a packet is examined within the context of a session. A firewall must maintain some additional state about each connection it receives to perform this type of higher-level filtering.

### 10.13 Packet Filtering

A packet-filtering firewall makes decisions on a per-packet basis. Since no other information is needed besides the current packet, the firewall does not have to maintain any state information about other

packets it has seen. Packet-filtering firewalls are the simplest and most efficient firewalls, but they are not robust against attacks that span multiple packets, especially when each packet by itself is not indicative of an attack.

### 10.14 Packet Filtering Firewall

A packet-filtering firewall is typically set up as a list of rules, where each rule maps one or more packet attributes to an action. The attributes are used to match incoming TCP/IP packet header fields and the actions are typically either forward or discard.

The firewall may define rules that look at many different packet headers. For example, the firewall might inspect the source and/or destination IP address of the packet. Additionally, the firewall may look at the source and/or destination transport-level address (port number). The firewall may also take the IP protocol field - whether TCP, UDP or ICMP - into account.

A firewall with multiple interfaces may also match packets based on which interface the packet was received on or which interface the packet is being sent to. This approach is useful when there are multiple ports in an enterprise network that require multiple security policies.

If a packet doesn't match any of the rules defined in the firewall, then a default action must be taken. There are two default policies: the default discard policy and the default forward policy.

The **default discard policy** says that if no rule matches the packet, then the packet will be discarded. This is a more secure policy because it provides more control over the traffic that is allowed in the network. However, it can be a hindrance to users who wish to engage with new network services and must tell the sysadmin to enable such traffic.

The **default forward policy** says that if no rule matches the packet, the packet should be forwarded. Compared with the default discard policy, this policy is more user-friendly, albeit less secure. The security admin must react to each new security threat and add the appropriate rules to the firewall if they choose to use this policy.

### 10.15 Firewall Filtering Quiz

 Firewall Filtering Quiz

Rank each policy based on user convenience and security.  
Use number 1 for best, 2, 3 for worst

Policy	Ease of Use	Security
Accepts only packets it knows are safe		
Drops packets it knows are unsafe		
Queries user about questionable packet		

### 10.16 Firewall Filtering Quiz Solution



### Firewall Filtering Quiz

Rank each policy based on user convenience and security.  
Use number 1 for best, 2, 3 for worst

Policy	Ease of Use	Security
Accepts only packets it knows are safe	3	1
Drops packets it knows are unsafe	1	3
Queries user about questionable packet	2	2

The first example follows the “default drop” rule, which is high security but requires new services to be expressly allowed. The second example follows the “default forward” rule, which is easier to use at the expense of security. The final approach sits in between the two in terms of security and ease of use.

### 10.17 Typical Firewall Configuration

Most standard applications that run on top of TCP follow a client-server model. The Simple Mail Transfer Protocol (SMTP), for example, allows client systems to send email to server systems. The client generates new email messages, typically from user input, and the server accepts incoming messages and places them in the appropriate user mailboxes.

SMTP operates by setting up a TCP connection between the client and the server, in which the server uses port 25 and the client uses any port between 1024 and 65535.

The port numbers below 1024 are the [well-known port numbers](#) and are assigned permanently to common applications, such as port 25 for SMTP and port 80 for HTTP. The port numbers between 1024 and 65535 are generated dynamically and have temporary significance only for the duration of the TCP connection from the client to the server.

A packet-filtering firewall that wants to support clients of well-known services must permit inbound network traffic on all these high-numbered ports for TCP-based connections. A firewall that wants to support servers of well-known services must permit inbound network traffic on the well-known port or port ranges associated with those services.

### 10.18 Packet Filtering Examples

Let's look at a simplified rule set for SMTP traffic where the goal is to allow only inbound and outbound email traffic while denying all other traffic.

**Packet Filtering Examples**



Rule	Direction	Src Address	Dest address	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

The rules described here are applied top to bottom for each packet until there is a match.

The first two rules are required to permit SMTP traffic to flow between an external host and an internal provider. Rule one allows inbound SMTP traffic from an external client. Rule two allows outbound traffic over higher-numbered ports. In other words, the first rule allows the external client to make an SMTP request, and the second rule allows the internal server to respond.

Rules three and four are required to permit SMTP traffic to flow between an internal host and an external provider. Rule three allows outbound SMTP traffic from an internal client. Rule four allows inbound traffic over higher-numbered ports. In other words, the third rule allows the internal client to make an SMTP request, and the fourth rule allows the external server to respond.

The final rule is an explicit statement of the default policy, which is to deny packets that don't match any of the previous rules.

### 10.19 Modifying the Rules on Source Ports

There are several problems with the ruleset shown above. For example, rule four allows inbound traffic to any destination port above 1023, whereas the original intent is to only allow inbound traffic that is part of an established SMTP connection. We can make our forwarding rules less permissive by looking at the source port in addition to the destination port.

#### Modifying the Rules on Source Ports

Rule	Direction	Src Address	Dest address	Protocol	Dest port	Action	Source
1	In	External	Internal	TCP	25	Permit	>1023
2	Out	Internal	External	TCP	>1023	Permit	25
3	Out	Internal	External	TCP	25	Permit	>1023
4	In	External	Internal	TCP	>1023	Permit	25
5	Either	Any	Any	Any	Any	Deny	

Now rule four only allows incoming traffic to high-numbered ports if that source port for that traffic is 25; that is, incoming connections are only allowed from mail servers.

We can make these rules even more precise. When a TCP connection is established, the ACK bit is set on the packet header. Because the intent of rule four is to allow inbound SMTP traffic that is part of an established connection, we can check that this bit is set before forwarding the packet.

## Modifying the Rules on Source Ports

Rule	Direction	Src Address	Dest address	Protocol	Dest port	Action	Source	ACK
1	In	External	Internal	TCP	25	Permit	>1023	
2	Out	Internal	External	TCP	>1023	Permit	25	
3	Out	Internal	External	TCP	25	Permit	>1023	
4	In	External	Internal	TCP	>1023	Permit	25	SET
5	Either	Any	Any	Any	Any	Deny		

### 10.20 Packet Filtering Advantages

The main advantage of packet-filtering firewalls is their simplicity and ease of implementation. Packet-filtering firewalls are also very efficient and impose very little overhead. Finally, the rules for packet-filtering firewalls can be very general, since they don't have to take higher-level applications into account.

### 10.21 Packet Filtering Weaknesses

Since packet-filtering firewalls do not examine upper-layer application data, they cannot prevent attacks that exploit application-specific vulnerabilities. If a packet-filtering firewall allows traffic to a given application, it allows all traffic to flow to that application indiscriminately.

Additionally, the logging capabilities of packet-filtering firewalls are limited. A firewall cannot log information about packet data that it doesn't examine. For example, a firewall may allow FTP traffic over port 21, and it may log the presence of FTP traffic, but it cannot log the actual FTP data, such as which files are being transmitted.

Since packet-filtering firewalls make decisions on a per-packet basis, they can't defend against attacks that span multiple packets.

Finally, our SMTP example shows that packet-filtering rules tend to have a small number of conditions, which may be too permissive. An attacker might craft traffic that exploits these misconfigurations.

## 10.22 Packet Filtering Firewall Countermeasures

Let's discuss some attacks on packet-filtering firewalls and the appropriate countermeasures.

In a **source IP address spoofing** attack, the attacker sends packets from an outside host but with a falsified source IP address matching an internal host. Since firewalls are typically configured to forward traffic from one internal host to another, the attacker hopes that using a spoofed internal source IP address will make untrusted external traffic look like safe internal traffic.

To counter this attack, the firewall must discard all packets with an internal source IP address that arrive on an external interface. This countermeasure is often implemented at the router external to the firewall.

In a **source routing attack**, an attacker specifies the route a packet should take as it crosses the Internet. The hacker hopes that their selected route will bypass security measures and checks along the way. The countermeasure for this attack is to configure the firewall or router to discard all packets that use this option.

In a **tiny fragment attack**, the attacker uses [IP fragmentation](#) to create extremely small packet fragments and then splits the TCP header information across separate fragments. This attack is designed to circumvent filtering rules that depend on TCP header information.

Typically, a packet filter will make decisions based on the first fragment of a packet. The attacker hopes that the firewall only examines the first fragment and that the remaining fragments - containing header information that would normally cause a packet to be dropped - are passed through.

This attack can be defeated by forcing the first fragment of a packet to contain a predefined minimum amount of transport header information. If the first fragment is rejected, all of the subsequent fragments should also be rejected.

### 10.23 Packet Filtering Quiz



#### Packet Filtering Quiz

In order for a fragmented packet to be successfully reassembled at the destination each fragment must obey the following rules. Mark all answers that are true:

- Must not share a common fragment identification number.
- Each fragment must say what its place or offset is in the original unfragmented packet.
- Each fragment must tell the length of the data carried in the fragment.
- Finally the fragment does not need to know whether more fragments follow this one.

### 10.24 Packet Filtering Quiz Solution



#### Packet Filtering Quiz

In order for a fragmented packet to be successfully reassembled at the destination each fragment must obey the following rules. Mark all answers that are true:

- F Must not share a common fragment identification number.
- T Each fragment must say what its place or offset is in the original unfragmented packet.
- T Each fragment must tell the length of the data carried in the fragment.
- F Finally the fragment does not need to know whether more fragments follow this one.

### 10.25 Stateful Inspection Firewall

In a **stateful inspection firewall**, a packet is analyzed within a larger context. This context often consists of the other packets present in the TCP connection through which the packet is transmitted.

To evaluate an incoming packet within this context, the firewall must record and maintain information about active connections. When a new packet arrives, the firewall updates the information about the connection accordingly and then decides whether the packet should be forwarded based on this context.

A stateful firewall can have a much higher-level view of traffic than a packet-filtering firewall. For example, it may be able to tell that an inbound SMTP packet is a response to a previously outbound packet. Additionally, it can reassemble multiple packets of a connection and inspect the application data, such as the names of files being transmitted during an FTP session.

### 10.26 Connection State Table

Here is an example of a connection table.

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.21.22.12	1046	192.168.1.6	80	Established

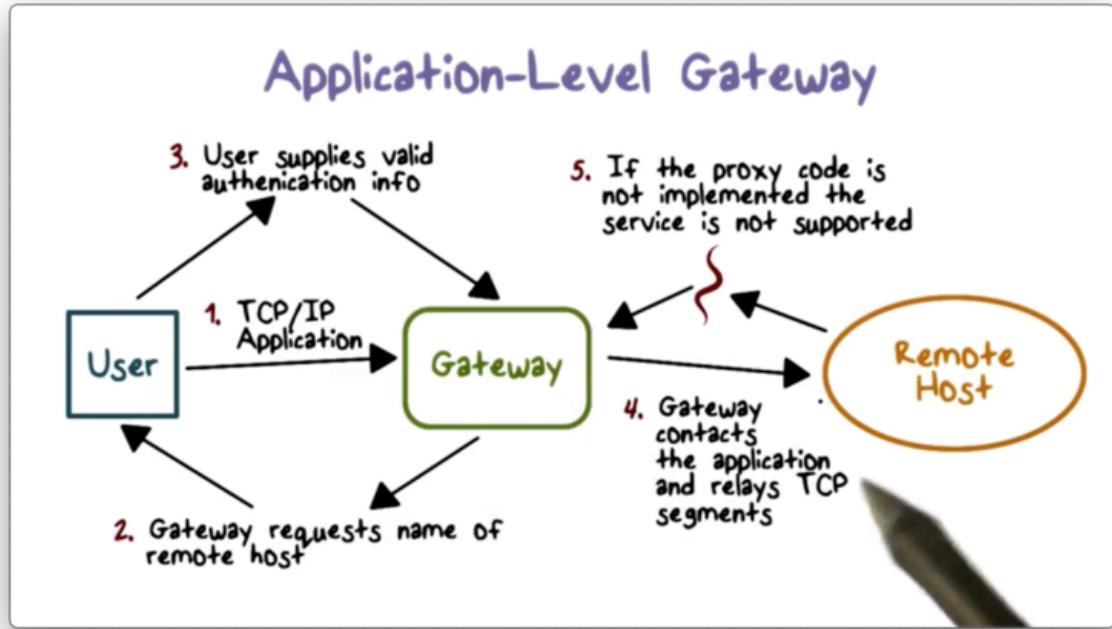
The firewall may maintain internal data structures that are linked to the connection table. For example, the HTTP response of a web server serving a page can span multiple packets, so the firewall might maintain a data structure that keeps track of the HTML that it has already delivered. This data structure will allow the firewall to perform a more specific analysis of the connection.

## 10.27 Application Level Gateway

An **application-level gateway** also referred to as an application proxy, is an application-specific firewall that essentially acts as a *relay* of application-level traffic. From the perspective of an internal client, the gateway is the external server. From the perspective of the external server, the gateway is the internal client.

To use an application proxy, a user first contacts the gateway using an application protocol such as FTP or HTTP. The gateway then asks the user for the name of the remote server, and the user responds with authentication information which the gateway relays. When the server responds, the gateway will analyze the response and potentially deliver it to the user.

The gateway must implement the correct application logic to correctly analyze the server response. For example, if the gateway is proxying web traffic, it must be able to process HTTP responses just like a web browser.



The advantage of using an application proxy is that we can restrict certain application features. For example, a web proxy can prevent active scripts in web pages by removing them from the HTML returned by the remote server. Since application proxies have more application-specific context than packet filters, they tend to be more secure.

The main disadvantage of proxies is that they incur additional overhead since they must examine and forward all traffic in both directions between the client and server. Additionally, we must install or write proxying code for each application we want to protect.

### 10.28 Filtering Quiz



#### Filtering Quiz

Mark each statement as either  
**T for True or F for False:**

- A packet filtering firewall is typically configured to filter packets going in both directions.
- A prime disadvantage of an application-level gateway is the additional processing overhead on each connection.
- A packet filtering firewall can decide if the current packet is allowed based on another packet that it has just examined.
- A stateful inspection firewall needs to keep track of information of an active connection in order to decide on the current packet.

### 10.29 Filtering Quiz Solution



### Filtering Quiz

Mark each statement as either T for True or F for False:

- A packet filtering firewall is typically configured to filter packets going in both directions.
- A prime disadvantage of an application-level gateway is the additional processing overhead on each connection.
- A packet filtering firewall can decide if the current packet is allowed based on another packet that it has just examined.
- A stateful inspection firewall needs to keep track of information of an active connection in order to decide on the current packet.

### 10.30 Bastion Hosts

Application-level gateways typically reside on a dedicated machine called a **bastion host**. These machines are made to be very secure.

Bastion hosts execute a secure version of the operating system and are configured to allow only essential network traffic, such as web and DNS traffic. Additionally, they can require user authentication, even for traffic originating from an internal host.

Each proxy running on the bastion host is configured to allow access only to specific host systems on the internal network. This configuration ensures that compromised proxies don't lead to attacks on the entire internal network.

Each proxy module installed on the bastion host is a very small software package designed with simplicity and security in mind. For example, a typical Unix email application may contain over 100,000 lines of code while an email proxy might contain fewer than 1,000.

A proxy typically performs no disk access other than reading its initial configuration file. This makes it difficult for an attacker to install a Trojan horse on the bastion host and affect the proxy.

Finally, each proxy typically runs as a non-privileged user in a private, secure directory on the bastion host. If a proxy is compromised during an attack, it cannot easily compromise the entire bastion host or affect other proxies.

### **10.31 Host Based Firewalls**

While firewalls are typically installed at network perimeters, host-based firewalls can be used to enforce a host-specific security policy. Similar to a network firewall, a host-based firewall is a software module used to filter and restrict inbound and outbound traffic. Host-based firewalls are commonly installed on important internal servers.

### **10.32 Host Based Firewall Advantages**

There are several advantages to host-based firewalls. First, since these firewalls are host-specific, each instance can implement filtering rules that are tailored to the application it protects. Second, a host-based firewall can stop both internal and external attacks, while a perimeter-resident firewall can only protect against external attacks.

We typically deploy host-based firewalls in addition to network firewalls to provide an additional layer of protection.

### **10.33 Personal Firewalls**

Personal firewalls are often deployed at home routers to protect home networks. These firewalls tend to be simpler than host-based firewalls or network-perimeter firewalls because home networks are often less complex than enterprise networks.

The main goal of the personal firewall is to protect the computers in the home network. For example, it can detect and block incoming remote access attempts and outgoing connections to external servers that are known for performing botnet C&C. Therefore, even when a home computer has been compromised, it cannot participate in malicious activities.

Here are some of the common network services that are typically blocked by a personal firewall. A user may configure the firewall to allow one or more of these services if desired.

## Personal Firewalls - Common Services

- Personal file sharing (548, 427)
- Windows sharing (139)
- Personal Web sharing (80, 427)
- Remote login-SSH (22)
- FTP access (20-21, 1024-65535 from 20-21)
- Remote Apple events (3031)
- Printer sharing (631, 515)
- IChat Rendezvous (5297, 5298)
- ITunes Music Sharing (3869)
- CVS (2401)
- Gnutella/Limewire (6346)
- ICQ (4000)
- IRC (194)
- MSN Messenger (6891-6900)
- Network Time (123)
- Retrospect (497)
- SMB (without netbios-445)
- VNC (5900-5902)
- WebSTAR Admin (1080, 1443)



### 10.34 Advanced Firewall Protection

Besides disabling and enabling certain services, personal firewalls can also provide advanced features.

A firewall can be configured to operate in “stealth mode”: dropping all unsolicited packets it receives from the Internet to hide the system(s) behind it.

A user can configure the firewall to drop all UDP and ICMP packets and only accept TCP packets to certain ports.

Additionally, most firewalls have logging capabilities and they can record all unwanted traffic and activities.

Finally, a user might be able to configure the firewall to only allow certain applications - such as those signed by public keys issued by a valid certificate authority - to accept connections from the Internet.

### 10.35 Personal Firewalls Quiz



#### Personal Firewalls Quiz

A company has a conventional firewall in place on its network. Which (if any) of these situations requires an additional personal firewall?

- An employee uses a laptop on the company network and at home.
- An employee uses a desktop on the company network to access websites worldwide
- A remote employee uses a desktop to create a VPN on the company's secure network.
- None of the above, in each case the employee's computer is protected by the company firewall.



### 10.36 Personal Firewalls Quiz Solution



### Personal Firewalls Quiz

A company has a conventional firewall in place on its network. Which (if any) of these situations requires an additional personal firewall?

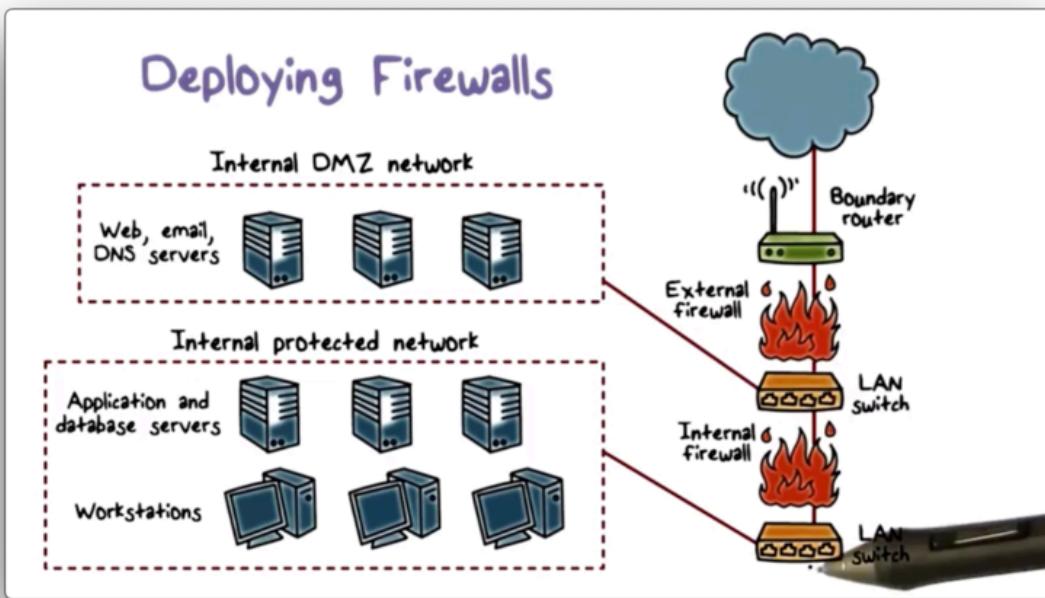
- An employee uses a laptop on the company network and at home.
- An employee uses a desktop on the company network to access websites worldwide
- A remote employee uses a desktop to create a VPN on the company's secure network.
- None of the above, in each case the employee's computer is protected by the company firewall.



If the device is not always protected by the corporate network, as is the case in scenarios 1 and 3, then the personal firewall is needed for additional security.

### 10.37 Deploying Firewalls

The figure below illustrates a common firewall configuration used by enterprise networks.



An external firewall is placed at the edge of the local area network, just inside of the boundary router that connects the corporate network to the Internet. Additionally, one or more internal firewalls protect the bulk of the enterprise network.

Between the internal firewall and the external firewall is a network known as a **demilitarized zone** (DMZ). Systems that require external connectivity but need some protections are typically located in the DMZ, such as corporate web, email, and DNS servers.

The external firewall provides some basic, first-line defense, while still allowing public access to systems in the DMZ. The internal firewall provides additional protection; in particular, it protects the internal trusted network from less-trusted traffic originating in the DMZ.

### 10.38 Internal Firewalls

Compared with the external firewall in the diagram above, the internal firewall performs more stringent filtering. This is because the internal network requires more protection than the public-facing systems in the DMZ.

The internal firewall protects the remainder of the enterprise network from attacks launched from the DMZ. For example, if a public-facing web server is compromised, the internal firewall can block attack traffic originating from that server.

Additionally, the internal firewall controls access to the DMZ from the internal network. Only authorized traffic from the internal network can reach the DMZ to change, say, a setting on a public-facing web server. For example, such traffic may be limited to having a source IP address of a sysadmin's workstation.

Multiple internal firewalls can be used to protect different parts of the internal network from each other; that is, even if one part of the internal network has been compromised, the other parts are still being protected by their local firewalls.

### **10.39 Distributed Firewall Deployment**

A distributed firewall configuration typically includes standalone network firewalls, host-based firewalls, and personal firewalls.

The standalone network firewalls protect the internal network from attacks from the Internet. Additionally, host-based firewalls protect against internal attacks and provide protection tailored to specific machines and server applications. Finally, personal firewalls protect personal computers regardless of where they are in the network.

Security administrators may aggregate and analyze logs from all of the firewalls in the network to get a high-level view of network traffic and compromise. For example, if admins see that multiple host-based firewalls are logging the same attack, they may conclude that a worm is spreading inside the internal network.

#### 10.40 Firewall Deployment Quiz



#### Firewall Deployment Quiz

Choose the most correct answer and enter the corresponding letter in the text box.

Typically the systems in the  require or foster external connectivity such as a corporate Web site, an e-mail server, or a DNS server.

- A. DMZ
- B. IP protocol field
- C. boundary firewall
- D. VPN

### 10.41 Firewall Deployment Quiz Solution



#### Firewall Deployment Quiz

Choose the most correct answer and enter the corresponding letter in the text box.

Typically the systems in the  require or foster external connectivity such as a corporate Web site, an e-mail server, or a DNS server.

- A. DMZ
- B. IP protocol field
- C. boundary firewall
- D. VPN

### 10.42 Stand Alone Firewall Quiz



#### Stand-alone Firewall Quiz

Choose the most correct answer and enter the corresponding letter in the text box.

A  configuration involves stand-alone firewall devices plus host-based firewalls working together under a central administrative control.

- A. packet filtering firewall
- B. distributed firewall
- C. boundary firewall
- D. VPN

### 10.43 Stand Alone Firewall Quiz Solution



#### Stand-alone Firewall Quiz

Choose the most correct answer and enter the corresponding letter in the text box.

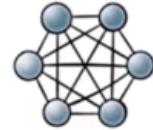
A **B** configuration involves stand-alone firewall devices plus host-based firewalls working together under a central administrative control.

- A. packet filtering firewall
- B. distributed firewall
- C. boundary firewall
- D. VPN



## 10.44 Firewall Topologies

### Firewall Topologies



- Host-resident firewall: includes personal firewall software and firewall software on servers
- Screening router: single router between internal and external networks with stateless or full packet filtering
- Single bastion inline: single firewall device between an internal and external router
- Single bastion T: has a third network interface on bastion to a DMZ where externally visible servers are placed.
- Double bastion inline: DMZ is sandwiched between bastion firewalls
- Double bastion T: DMZ is on a separate network interface on the bastion firewall
- Distributed firewall configuration: used by some large businesses and government organizations

## 11 Intrusion Detection

### 11.1 Defense in Depth

Recall the *defense in depth* principle: we need multiple layers of defense mechanisms. We typically deploy prevention systems to block attacks, but some intrusion attempts are always going to get through. If we can't stop an attack, at the very least, we need to detect it. Specifically, we need to deploy intrusion detection systems to inform us when an attack occurs.

### 11.2 Intrusion Examples

An intrusion is any attack that aims to compromise the security goals of an organization. The following activities are all examples of intrusion.

## Intrusion Examples

- Remote root compromise
- Web server defacement
- Guessing/cracking passwords
- Copying databases containing credit card numbers
- Viewing sensitive data without authorization
- Running a packet sniffer
- Distributing pirated software
- Using an unsecured modem to access internal network
- Impersonating an executive to get information
- Using an unattended workstation



### 11.3 Intrusion Detection Quiz



#### Intrusion Detection Quiz

Select the characteristic that best describes each network security system.

Type (F) for Firewalls or (I) for IDS:

- tries to stop intrusion from happening
- tries to evaluate an intrusion after it has happened
- watches for intrusions that start within the system
- limits access between networks to prevent intrusion

#### 11.4 Intrusion Detection Quiz Solution



#### Intrusion Detection Quiz

Select the characteristic that best describes each network security system.

Type (F) for Firewalls or (I) for IDS:



tries to stop intrusion from happening



tries to evaluate an intrusion after it has happened



watches for intrusions that start within the system



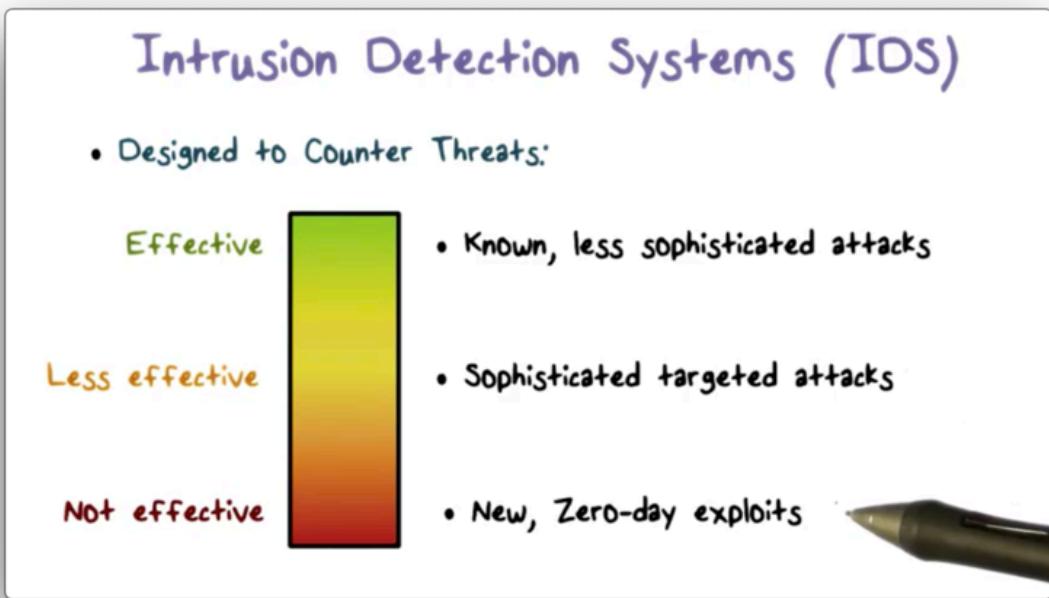
limits access between networks to prevent intrusion

#### 11.5 Intrusion Detection System (IDS)

An **intrusion detection system** (IDS) can be quite effective against well-known or less sophisticated attacks, such as large scale email phishing attacks.

However, as attack techniques become more sophisticated, IDS's become less effective. For example, attackers can blend attack traffic with normal activities, making detection of these attacks much more difficult.

The most sophisticated attackers, such as state-sponsored individuals, may use new, zero-day exploits. IDS's are not effective against brand new types of attacks.



Since IDS's are not completely effective, they need to be part of the broader defense-in-depth strategy for an organization.

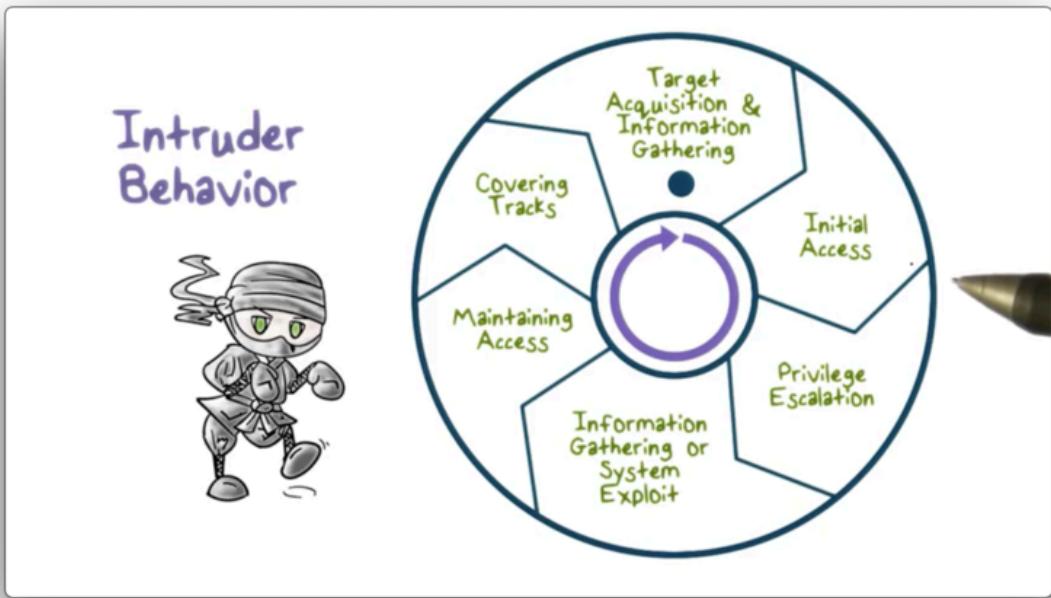
Other standard security strategies include:

- encrypting sensitive information
- providing detailed activity audit trails
- using robust authentication and access control practices
- actively managing operating system/application security

### 11.6 Intruder Behavior

The techniques and behavioral patterns of intruders are continually evolving to exploit newly discovered weaknesses and evade detection and countermeasures. However, attackers generally follow the same high-level steps when conducting an attack.

The following diagram illustrates these steps.



The first step is target acquisition and information gathering. An attacker identifies the target systems using publicly available technical and non-technical information.

From a technical perspective, the attacker might use network tools to analyze target resources, such as determining which network services are accessible over the Internet. A less technical step might involve looking at organizational email patterns - like lastname.firstname@company.com - to discern the addresses of some high-level executives.

The second step is the initial access. An attacker gains access by exploiting a remote network vulnerability. For example, a network service may have a buffer overflow vulnerability that an attacker can exploit remotely. Alternatively, an attacker can gain access through social engineering by, for example, sending an email with a malicious attachment to a company executive.

The third step is privilege escalation. In this step, the attacker tries to use a local exploit to escalate their privilege from an unprivileged user to a root user on the target system.

The fourth step is information gathering or system exploit. After an attacker gains sufficient privilege on a system, they can find out more about the network and the organization, and even move to another target system to advance the exploit further.

The fifth step is maintaining access. An attack might not be a one-time action; that is, an attacker may choose to come back from time to time or continue the exploit over an extended period. The attacker may install backdoors or other malicious software on the target system to maintain access.

Finally, the attacker must cover their tracks. They can edit or disable the system audit logs to remove evidence of attack activities. Alternatively, they might install rootkits to hide any installed malware.

### 11.7 Intruder Quiz



### Intrusion Detection Quiz

Type True (T) or False (F) for each statement:

- An intruder can also be referred to as a hacker or cracker.
- Activists are either individuals or members of an organized crime group with a goal of financial reward.
- Running a packet sniffer on a workstation to capture usernames and passwords is an example of intrusion.
- Those who hack into computers do so for the thrill of it or for status.
- Intruders typically use steps from a common attack methodology.

### 11.8 Intruder Quiz Solution



### Intrusion Detection Quiz

Type True (T) or False (F) for each statement:

- T An intruder can also be referred to as a hacker or cracker.
- F Activists are either individuals or members of an organized crime group with a goal of financial reward.
- T Running a packet sniffer on a workstation to capture usernames and passwords is an example of intrusion.
- F Those who hack into computers do so for the thrill of it or for status.
- T Intruders typically use steps from a common attack methodology.

### 11.9 Types of Backdoors Quiz



#### Types of Backdoors Quiz

Choose the description that best fits each type of backdoor:

Compiler Backdoors

A. This backdoor is hard to detect because it modifies machine code.

Object Code Backdoors

B. This backdoor can only be used by the person who created it, even if it is discovered by others.

Asymmetric Backdoors

C. This backdoor inserts backdoors into other programs during compilation.

### 11.10 Types of Backdoors Quiz Solution



### Types of Backdoors Quiz

Choose the description that best fits each type of backdoor:

<input checked="" type="checkbox"/> C Compiler Backdoors	A. This backdoor is hard to detect because it modifies machine code.
<input type="checkbox"/> A Object Code Backdoors	B. This backdoor can only be used by the person who created it, even if it is discovered by others.
<input type="checkbox"/> B Asymmetric Backdoors	C. This backdoor inserts backdoors into other programs during compilation.

Read more [here](#), [here](#), and [here](#).

### 11.11 Elements of Intrusion Detection

For intrusion detection to be possible, we need to make some important assumptions. First, we have to assume that we can observe system, network, and user activities. Second, we have to assume that we can distinguish intrusive activities from ordinary activities.

When building an intrusion detection algorithm, we must consider the following. First, we need to identify the behavioral *features* and activities that we want to capture from the system. Second, we must synthesize our detection *models*; that is, we need to stitch together the evidence we collect to determine whether a pattern of activity represents an intrusion.

An IDS typically includes several modules, including:

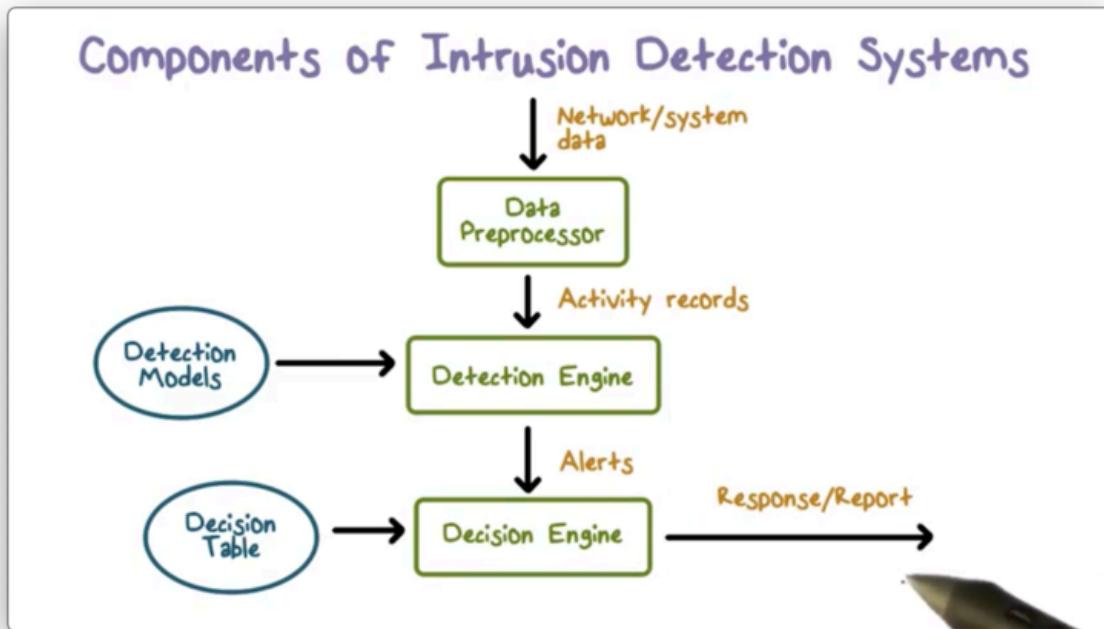
- an audit data preprocessor
- a knowledge base
- a decision engine
- alarm generation and response mechanisms

### 11.12 Components of Intrusion Detection Systems

The input to an IDS is network and system data that provides a snapshot of the activity currently taking place within a computer system. The **data preprocessor** reads this data and extracts *activity records* that are important for security analysis.

The **detection engine** analyzes the activity records using detection models packaged with the IDS. If a detection rule determines that an intrusion is happening, the IDS produces an alert.

The **decision engine** responds to the alert and decides the appropriate action according to a *decision table*. In one case, an action might be tearing down a particular network connection while, in another case, the action might be sending a report to the security admin.



Again, for the IDS to work correctly, we must be able to assume that system activities are observable and that ordinary and intrusive activities have distinguishable features.

### 11.13 Intrusion Detection Approaches

From a modeling and analysis perspective, there are two different approaches to intrusion detection. The **misuse detection** - signature-based - strategy models known intrusions, while **anomaly detection** models look for abnormal behavior, regardless of whether they have explicitly been seen previously in exploits.

From a deployment perspective, we can position an IDS on end hosts or at network perimeters.

From a development and maintenance perspective, we can create build detection models using manual encoding of expert knowledge - basically static rules translated into code - or we can apply machine learning algorithms to data to automatically learn the detection rules.

### **11.14 Analysis Approaches**

Anomaly detection consists primarily of two phases. In an initial training/profiling phase, the anomaly-based IDS collects system data corresponding to ordinary activity and uses data analysis algorithms to constructs a model representing this baseline state. After, the model continuously analyzes live system activity relative to the baseline to determine whether system activity is anomalous.

Misuse or signature-based detection involves first encoding known attacks into patterns or rules and then comparing current system activity against these patterns to determine if there is a match. Obviously, this approach can only detect known attacks.

### **11.15 Analysis Detection Quiz**



#### **Anomaly Detection Quiz**

Check all answers that are true regarding Anomaly detection systems:

- The longer the system is in use, the more it learns about network activity.
- If malicious activity looks like normal traffic to the system, it will not detect an attack.
- False positives can become a problem, normal usage can be mistaken for an attack.

### 11.16 Analysis Detection Quiz Solution



#### Anomaly Detection Quiz

Check all answers that are true regarding Anomaly detection systems:

- T The longer the system is in use, the more it learns about network activity.
- T If malicious activity looks like normal traffic to the system, it will not detect an attack.
- T False positives can become a problem, normal usage can be mistaken for an attack.



### 11.17 Signature Detection Quiz



#### Signature Detection Quiz

Check all answers that are true regarding  
Signature Based detection:

- New threats can be detected immediately.
- When a new virus is identified, it must be added to the signature databases
- Can only detect an intrusion attempt if it matches a pattern that is in the database



### 11.18 Signature Detection Quiz Solution



#### Signature Detection Quiz

Check all answers that are true regarding  
Signature Based detection:

- F New threats can be detected immediately.
- T When a new virus is identified, it must be added to the signature databases
- T Can only detect an intrusion attempt if it matches a pattern that is in the database



### 11.19 A Variety of Classification Approaches

In the training phase, an anomaly-based IDS develops a model of normal or legitimate behaviors by collecting data from the normal operations of the monitored system or network. In the detection phase, the IDS compares observed behavior against the model and categorizes it as either legitimate or anomalous.

IDS developers can use a variety of approaches to construct these models. The statistical approach applies univariate, multivariate, or time-series models to observed metrics. The knowledge-based approach encodes legitimate behavior into a set of rules using expert knowledge. The machine learning approach uses data mining and machine learning techniques to learn a model from training data automatically.

When we compare these approaches, we need to consider both efficiency - how quickly can a system learn and apply a model - and cost - how much data and computation power we need to build that model.

### 11.20 Anomaly Quiz



### Anomaly Quiz

Which of the following could be considered an anomaly to typical network traffic?



- A IP address
- A port address
- Packet length
- Flag setting



### 11.21 Anomaly Quiz Solution



### Anomaly Quiz

Which of the following could be considered an anomaly to typical network traffic?



- A IP address
- A port address
- Packet length
- Flag setting



### 11.22 Statistical Approaches

Statistical approaches use captured system data as training input to develop a model of normal behavior. The earliest approaches used univariate models, where each metric - the CPU utilization of a program, for example - was treated as an independent random variable.

However, this approach was too crude to detect intrusions effectively. Later statistical systems used multivariate models that consider the correlations between the measures. For example, these systems might examine both CPU utilization and memory size of a program together.

More recent systems incorporate time-series models that consider the order and timing between observed events. For example, these systems might examine the amount of time elapsed between when a user logs into a workstation and when that workstation sends an email.

The main advantage of statistical approaches is that they are relatively simple, easy to compute, and they don't have to make many assumptions about behavior. On the other hand, the effectiveness of these approaches relies on selecting the right set of metrics to examine and correlate, which is a difficult task. Additionally, there are complex behaviors that are beyond the capability of these models.

### 11.23 Knowledge Based Approach

Knowledge-based approaches require experts to develop a set of rules that describe the normal, legitimate behaviors observed during training. These rules can separate activities into different classes; for example, a rule might state that a secretary *normally* only uses office productivity programs, like web browsers, email, calendar, and Microsoft office.

These rules can be quite robust, and systems built with this approach are relatively easy to update and improve. On the other hand, these approaches rely on manual efforts of experts, and these experts must have strong knowledge of the data and the domain.

### 11.24 Statistical & Knowledge Based Approaches Quiz



#### Statistical & Knowledge Based Approaches Quiz

Which of these characteristics describes the statistical approach and which describe a knowledge based approach? Write S or K in the box:

- Any action that does not fit the normal behavior profile is considered an attack.
- Any action that is not classified as normal is considered to be an attack.

### 11.25 Statistical & Knowledge Based Approaches Quiz Solution



### Statistical & Knowledge Based Approaches Quiz

Which of these characteristics describes the statistical approach and which describe a knowledge based approach? Write S or K in the box:

**S** Any action that does not fit the normal behavior profile is considered an attack.

**K** Any action that is not classified as normal is considered to be an attack.

### 11.26 Machine Learning Approaches

Machine learning approaches can build a model automatically using the labeled, normal training data. A machine learning algorithm takes as input examples of normal data and outputs a model that is then able to classify subsequently observed data as either normal or anomalous.

Machine learning models are powerful in that they can handle small changes or variations in the observed data and can also capture complex interdependencies between different observed features.

For these approaches to be effective, the normal training data must be representative of normal behaviors; otherwise, the learned model can produce many false positives. Additionally, the training phase of machine learning requires a lot of data, and significant time and computational resources. However, once the model is produced, subsequent analysis is typically reasonably efficient.

### 11.27 Machine Learning Intruder Detection Approaches

IDS can use a variety of machine learning algorithms.

**Bayesian networks** encode the conditional probabilities between observed events - the probability that a user is sending an email if the current time is 2 AM, for example. A system might raise an alert if it observes an event that has a significantly low probability of occurring.

**Markov models** are sets of states connected by transitional probabilities. For example, a system can use Markov models to model legitimate website names. The transitional probabilities from one letter to the next in a URL should be similar to that of real dictionary words.

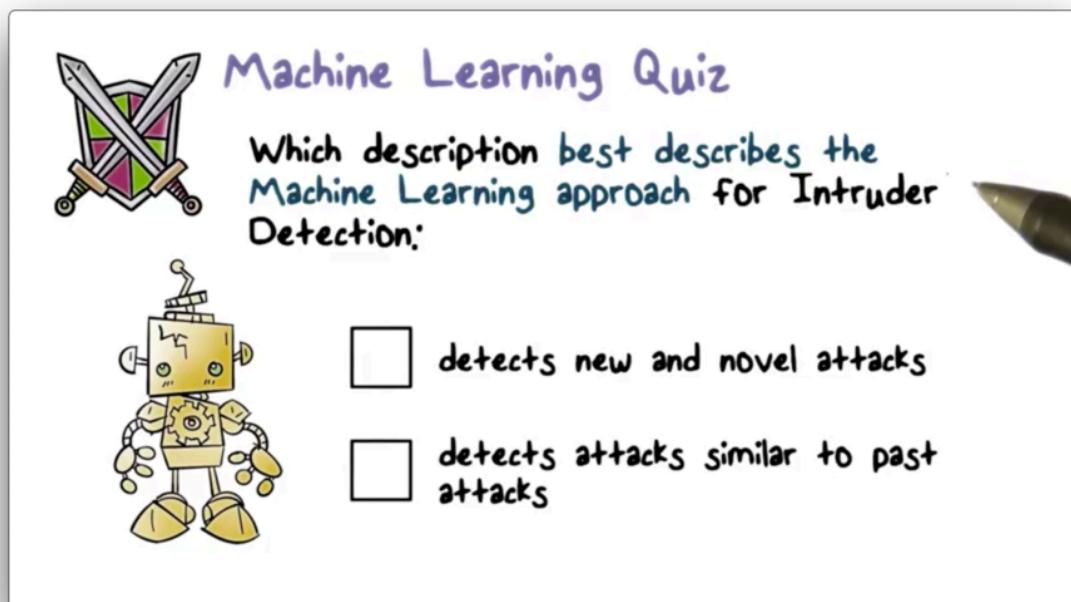
On the other hand, a randomly-spelled website name is anomalous and might indicate a botnet C&C server, since bots don't have to type out URLs.

**Neural networks** simulate how human brains perform reasoning and learning and are one of the most powerful machine learning approaches.

**Clustering** groups training data into clusters based on some similarity or distance measure and then identifies subsequently observed data as either a member of a cluster or an outlier.

For example, legitimate traffic from an internal network to an internal web server shares common characteristics that can be grouped into clusters based on the web pages visited. On the other hand, an attack may access data on the web server that is rarely visited, making it an outlier.

### 11.28 Machine Learning Quiz



The slide features a title 'Machine Learning Quiz' with a shield and crossed swords icon. Below the title is a question: 'Which description best describes the Machine Learning approach for Intruder Detection?'. It includes two options with checkboxes: 'detects new and novel attacks' and 'detects attacks similar to past attacks'. There is also a small robot icon and a pen icon.

Machine Learning Quiz

Which description best describes the Machine Learning approach for Intruder Detection?

detects new and novel attacks

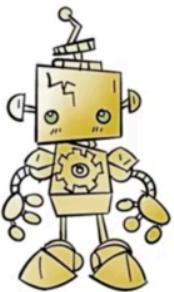
detects attacks similar to past attacks

### 11.29 Machine Learning Quiz Solution



### Machine Learning Quiz

Which description best describes the Machine Learning approach for Intruder Detection:



- detects new and novel attacks
- detects attacks similar to past attacks

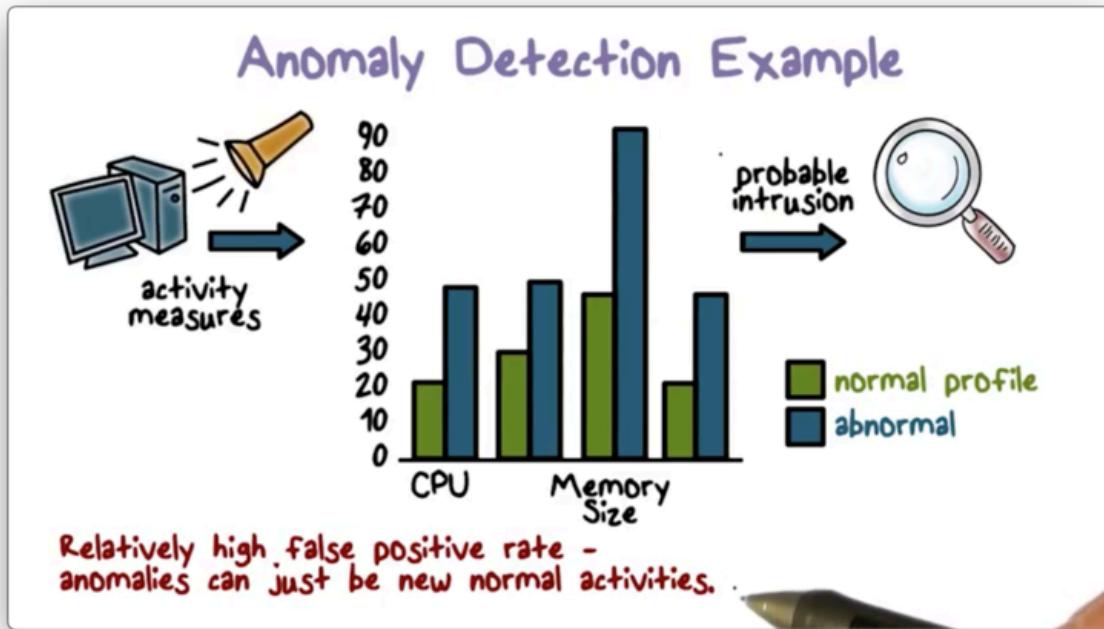


### 11.30 Limitations of Anomaly Detection

A fundamental limitation of anomaly detection models is that they train using only normal or legitimate data. Since these models do not see any intrusion data during training, they cannot be sure that a general anomaly is evidence specifically of an intrusion.

### 11.31 Anomaly Detection Example

Here is an example of a straightforward anomaly detection approach.



First, the system establishes a normal, statistical runtime profile of a program. The system generates this profile by running the program many times, recording important metrics - CPU utilization and memory size, for example - each time, and then computing the mean and variance for each captured metric.

Once the IDS builds the profile, it can then compare subsequent system activity against this profile. If it observes that activity deviates from the means beyond the allowed variances, the IDS outputs an alert.

Again, the main drawback of the anomaly detection approach is that it can produce relatively high false-positive rates because an anomaly can be a new, unobserved, yet normal activity.

### 11.32 Misuse or Signature Detection

Misuse or signature-detection systems surface intrusions by comparing activity data against a local database of known intrusion patterns. If they detect a match in activity patterns, they can conclude that an intrusion has occurred; otherwise, the system is likely functioning normally.

### 11.33 Anomalous Behavior Quiz



#### Anomalous Behavior Quiz

One of the weaknesses of anomalous intruder detection is that a system must learn what is normal behavior. While it is learning this, the network is vulnerable to attack. What can be done to mitigate this weakness?

Write your answer in the textbox:

### 11.34 Anomalous Behavior Quiz Solution



### Anomalous Behavior Quiz

One of the weaknesses of anomalous intruder detection is that a system must learn what is normal behavior. While it is learning this, the network is vulnerable to attack. What can be done to mitigate this weakness?

Write your answer in the textbox:

Uses a Firewall.

### 11.35 Signature Approaches

Many misuse detection approaches use **signatures**, known patterns of malicious activity data, to categorize current system activity as malicious or benign. To maximize the detection rate and minimize the false-positive rate, the set of signatures against which these systems compare activity needs to be as large as possible. Signature-based approaches are most commonly found in anti-virus software and network intrusion detection systems.

### 11.36 Signature Approach Advantages & Disadvantages

Signatures are typically straightforward to understand, and signature matching is very efficient; therefore, signature-based approaches are widely used. On the other hand, significant manual effort must be spent to create new signatures every time a new malware or attack method appears. Additionally, these approaches cannot detect zero-day attacks because those attacks are new and do not yet have known signatures.

### 11.37 Zero Day Market Place Quiz



#### Zero Day Market Place Quiz

In the thriving zero day attack marketplace hackers sell information on software vulnerabilities. Can you guess some of the buyers?



<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>

- Apple
- Google
- Microsoft
- U.S. Government



### 11.38 Zero Day Market Place Quiz Solution



### Zero Day Market Place Quiz

In the thriving zero day attack marketplace hackers sell information on software vulnerabilities. Can you guess some of the buyers?



X	Apple
X	Google
X	Microsoft
X	U.S. Government



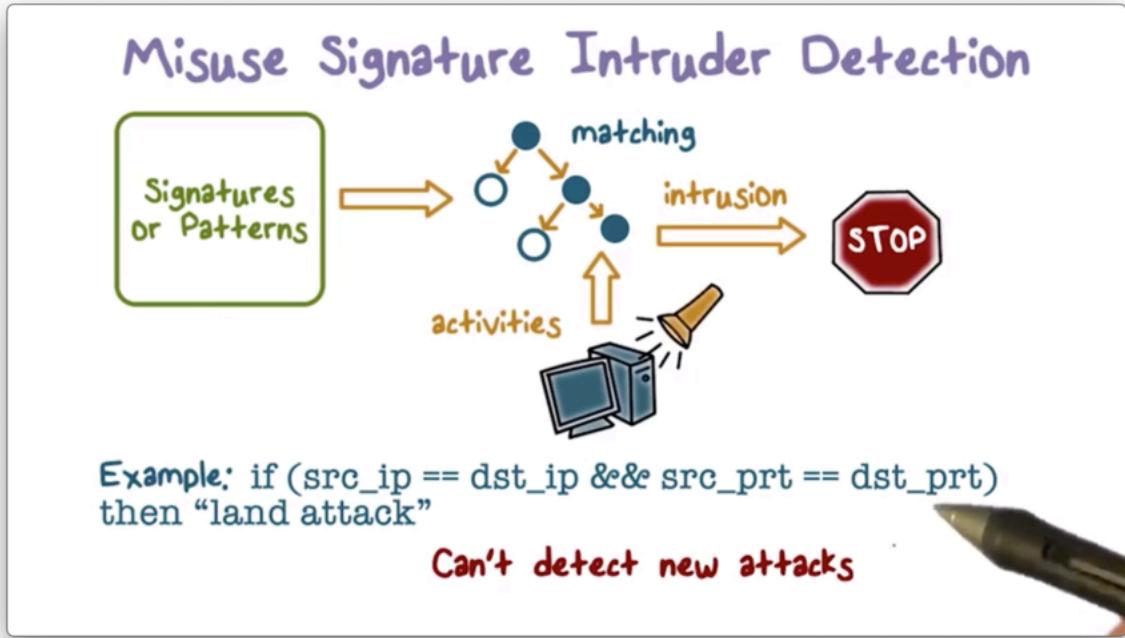
### 11.39 Rule Based Detection

In addition to signature-based strategies, a misuse detection system can also use a more sophisticated, rule-based approach. This approach uses rules to represent known intrusions, which typically match multiple signatures.

These rules are not only specific to known intrusions but can also be tweaked to fit the network, as the same intrusion may leave different evidence in different networks depending on the network setup. [Snort](#) is a widely-known example of a rule-based network intrusion detection system.

### 11.40 Misuse Signature Intruder Detection

The IDS matches the observed activities using a set of attack signatures or patterns. If there is a match, the IDS outputs an alert.



For example, the system may raise an alarm if it spots the so-called "land attack" defined in the picture above. Again, such a simple approach cannot detect new attacks since they don't have signatures.

### 11.41 Attacks Quiz



#### Attacks Quiz

Write the name of each attack next to its definition. The choices are Scanning Attack (S), DOS(D), and Penetration Attack(P).

- an attacker sends various kinds of packets to probe a system or network for vulnerability that can be exploited
- attempts to slow down or completely shut down a target so as to disrupt the service for legitimate users
- an attacker gains an unauthorized control of a system



### 11.42 Attacks Quiz Solution



#### Attacks Quiz

Write the name of each attack next to its definition. The choices are Scanning Attack (S), DOS(D), and Penetration Attack(P).

S

an attacker sends various kinds of packets to probe a system or network for vulnerability that can be exploited

P

attempts to slow down or completely shut down a target so as to disrupt the service for legitimate users

P

an attacker gains an unauthorized control of a system

### 11.43 Monitoring Networks and Hosts

An IDS typically performs passive monitoring; that is, it records and analyzes data about system and network activities while these activities continue to take place.

The IDS doesn't affect activities directly, but rather only outputs alerts. In response to these alerts, the system can intervene and alter activity by, for example, blocking a network connection or terminating a program.

### 11.44 Network IDS

We can deploy an intrusion detection system at the perimeter of a network or subnet to monitor inbound and outbound network traffic. Such a system is called a **network intrusion detection system** (NIDS).

NIDS's typically use a packet-capturing tool like `libpcap` to obtain network traffic data. The captured packet data contains the complete information about network connections; for example, TCP handshake information, as well as all of the URL requests from the client and the returned page contents from the server.

Since a NIDS can see all of the data sent and received by a user's machine, it can raise an alert if it detects something has gone awry. If the user's machine is infected by a bot malware, for example, the NIDS will see attempts to connect to a website for command and control, and it can generate the appropriate alert.

### **11.45 Network Based IDS (NIDS)**

A NIDS monitors traffic in real-time or close to real-time so that it can react to intrusions promptly. It can analyze traffic in multiple layers of the network stack, responding to network-, transport-, or application-level protocol activity.

A NIDS can include several sensors, each of which typically monitors the traffic of a subnet and reports its findings to a backend management server. This server correlates evidence across multiple sensors to provide intrusion detection across the whole network.

Additionally, a NIDS may provide one or more management consoles for use by human analysts needing visibility into network activity.

### **11.46 Host IDS**

We can also deploy an IDS in an end host. Most host-based IDS's use `ptrace` to obtain the system calls made by a program as a way to monitor program behavior.

System call data is crucial for security monitoring because the operating system manages critical system resources. Therefore, whenever a program requests a resource, such as memory allocation or access to the filesystems, networks, or I/O devices, it must make a system call to the operating system.

For example, If a user's browser receives a page containing malicious javascript that breaks the protection in the browser and attempts to overwrite a Windows registry file, the IDS will observe a write system call to the registry file and can generate an alert.

### 11.47 NIDS Quiz



#### NIDS QUIZ

Can you think of a way to reduce the impact of excessive reporting on a system's administrator?

Write your answer in the textbox:



### 11.48 NIDS Quiz Solution



### NIDS QUIZ

Can you think of a way to reduce the impact of excessive reporting on a system's administrator?

Write your answer in the textbox:

Prioritize the alerts

### 11.49 Inline Sensors

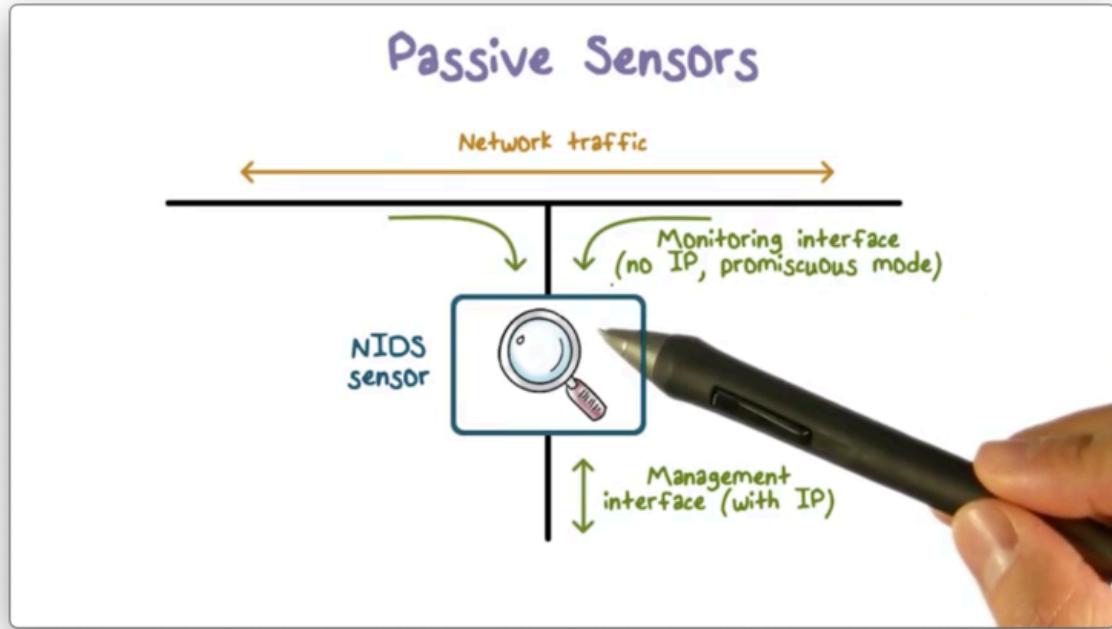
One way to configure a network IDS is by using *inline sensors*. The primary motivation for using inline sensors is to enable them to block an attack upon detection. In such cases, an inline sensor performs both intrusion detection and intrusion prevention.

We must place the sensor at a network point where traffic must pass through it for this strategy to be effective. We can deploy an inline sensor as a combination of a network IDS and a firewall in a single piece of hardware, or we can deploy the inline sensor as a standalone inline network IDS.

### 11.50 Passive Sensors

More commonly, we deploy network IDS's as passive sensors. A *passive sensor* only receives a copy of live traffic, and therefore cannot provide any intrusion protection; however, from a network performance perspective, this approach is much more efficient.

The following diagram illustrates a typical passive sensor configuration.



The sensor connects to the network transmission medium, such as an ethernet cable, through a direct physical tap that provides the sensor with a copy of all network traffic carried by the medium.

The network interface card (NIC) for the tap usually does not have an IP address configured for it, which means that the data collection side of the sensor is unreachable outside of the tap. The sensor has a second NIC that connects to the network with an IP address so that it can communicate with one or more backend management servers.

### 11.51 Firewall Versus Network IDS

A network IDS performs passive monitoring. While the IDS is copying and analyzing network traffic, the traffic continues to flow to its destination.

Traffic analysis can take a lot of computing power, and a large volume of traffic can overload an IDS, preventing it from detecting intrusion in a timely manner. We call this situation *fail-open*, meaning that the network is open to intrusion when the IDS fails.

A firewall performs active filtering. All traffic must pass through the firewall, and the firewall performs relatively simpler and more efficient analysis.

A large volume of traffic can overload a firewall as well. When this happens, it prevents all traffic from passing through. We call this *fail-close*, meaning that when a firewall fails, the internal network is closed to the external network, and is safe from intrusion.

### 11.52 IDS Quiz



#### IDS Quiz

Put a (T) for True next to each true statement and a (F) for False next to each false statement.

- Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified.
- The primary purpose of an IDS is to detect intrusions, log suspicious events, and send alerts.
- Signature-based approaches attempt to define normal, or expected, behavior, whereas anomaly approaches attempt to define proper behavior.
- A network IDS sensor monitors a copy of network traffic; the actual traffic does not pass through the device.
- Network-based intrusion detection makes use of signature detection and anomaly detection.

### 11.53 IDS Quiz Solution



## IDS Quiz

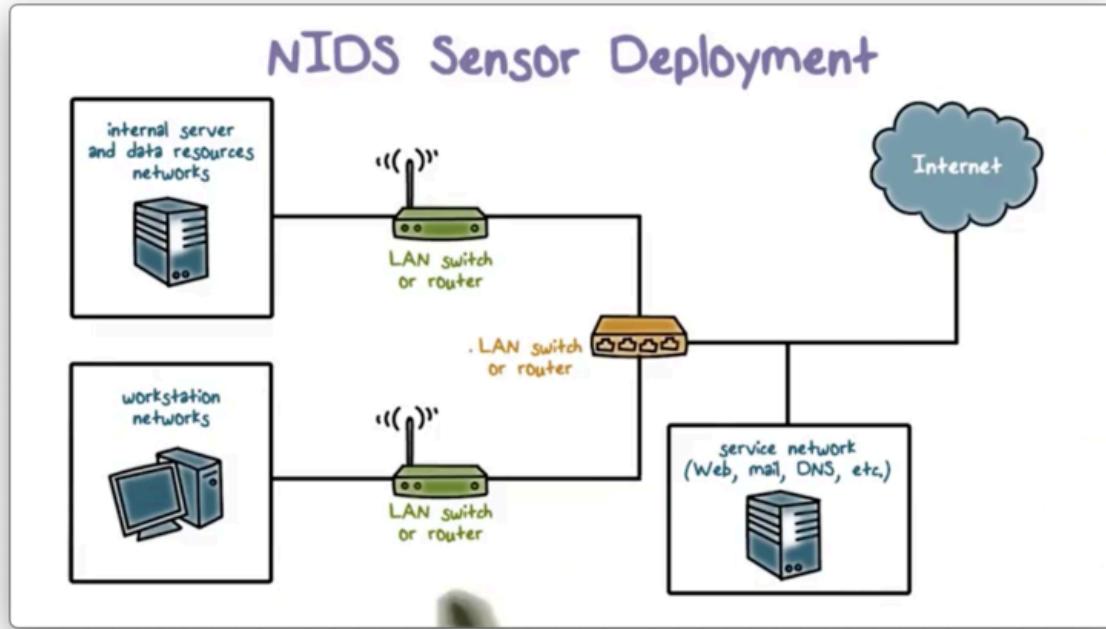
Put a (T) for True next to each true statement and a (F) for False next to each false statement.

- T Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified.
- T The primary purpose of an IDS is to detect intrusions, log suspicious events, and send alerts.
- F Signature-based approaches attempt to define normal, or expected, behavior, whereas anomaly approaches attempt to define proper behavior.
- T A network IDS sensor monitors a copy of network traffic; the actual traffic does not pass through the device.
- T Network-based intrusion detection makes use of signature detection and anomaly detection.

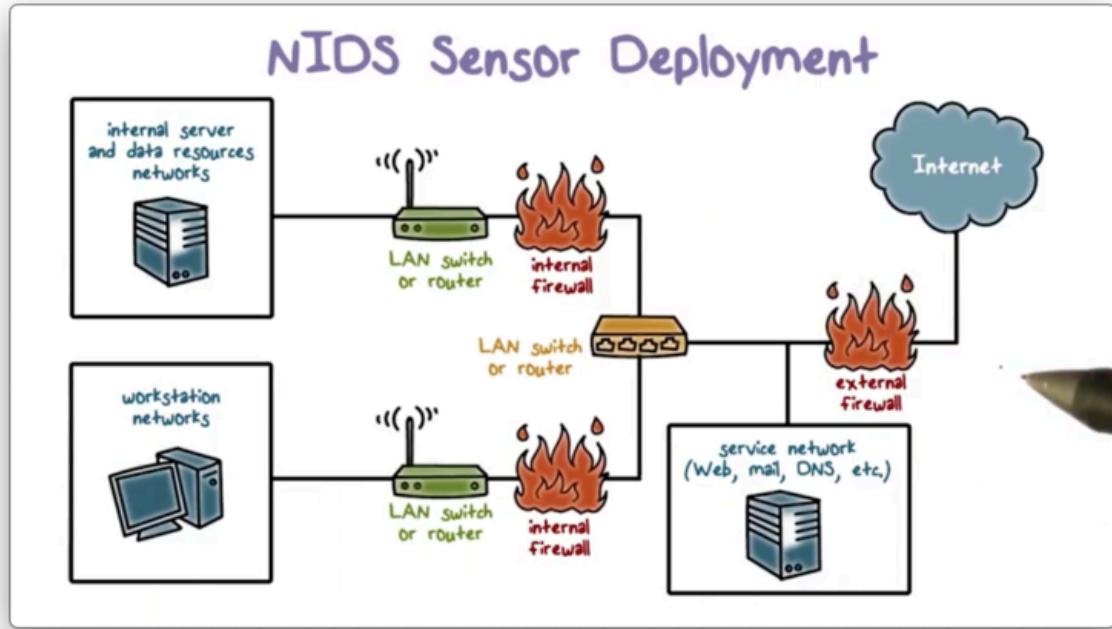


### 11.54 NIDS Sensor Deployment

Here is an example of an enterprise network configuration. The internal network has multiple subnets, and the enterprise has public-facing services, such as a public web server.



Recall our lecture on firewalls. We typically want to place an external firewall to protect the entire enterprise network. Additionally, we want to protect the internal network from the public-facing servers. We place these servers in a DMZ, and we use internal firewalls to monitor traffic between the DMZ and the internal subnets. The internal firewalls also monitor traffic between the subnets.



A common location for an IDS sensor is just inside the external firewall, a position which has several advantages.

An IDS in this position has an excellent vantage point. It can see attacks from the outside world aimed at the internal network as well as the public-facing servers in the DMZ.

Additionally, because the IDS in this position monitors all outgoing traffic across the entire enterprise network, it can also detect attacks originating from a compromised server either in the DMZ or the internal network.

As well, since the IDS sits just inside the external firewall, we can compare the logs of the IDS against the logs of the firewall to see whether the firewall missed an attack it should have prevented.

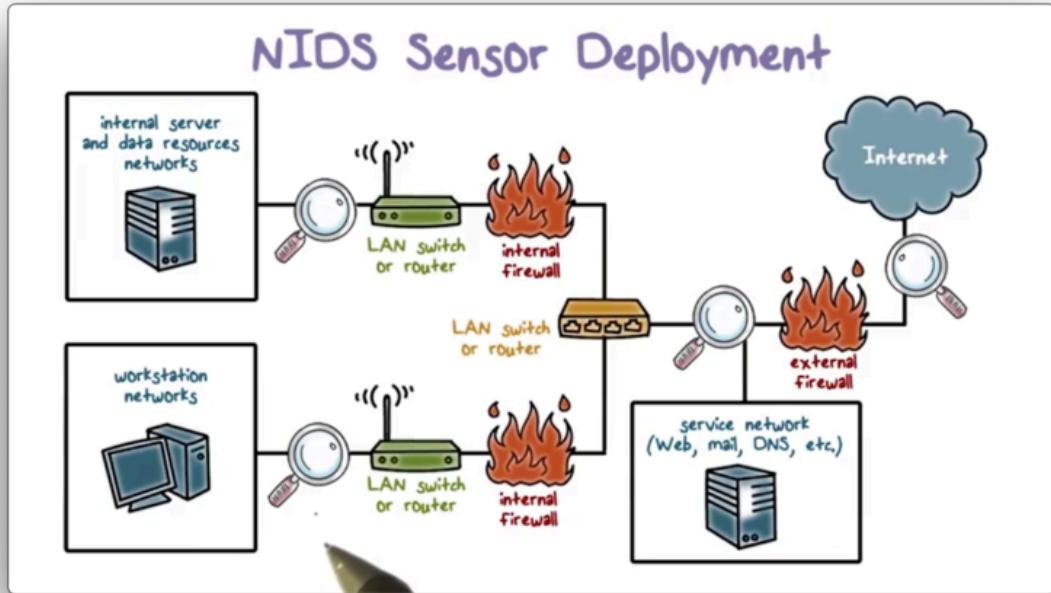
We can also place our IDS between the external network and the Internet. The main advantage of this location is that the IDS can see all attempted attacks aimed at the enterprise network, including those attacks that might be filtered by the external firewall.

For example, if the external firewall is overloaded, it not only drops incoming attack packets, but it might not even have the resources to log these packets. An IDS at this location can see the packet and log it.

We can also deploy the IDS at a subnet boundary. Compared with the IDS at the perimeter, which has to examine all traffic, an IDS at this location can perform a more detailed analysis of traffic data since it has less traffic to monitor. Additionally, a subnet IDS can detect intrusions from inside the network.

As well as protecting a subnet, an IDS can also protect the workstations or networks of important personnel or departments. An IDS at this location focuses on targeted attacks, such as attacks aimed at financial transaction systems.

Compared with an IDS at the network perimeter, which must examine traffic across the whole network, an IDS at this location can instead focus exclusively on traffic to high-value systems.



### 11.55 NIDS Sensor Deployment Quiz



#### NIDS Sensor Deployment Quiz

When using sensors which of the following is considered good practice? Check all the true statements:

- Set the IDS level to the highest sensitivity to detect every attack
- Monitor both outbound and inbound traffic
- Use a shared network resource to gather NIDS data
- NIDS sensors are not turnkey solutions, system administrators must interpret alerts.



### 11.56 NIDS Sensor Deployment Quiz Solution



#### NIDS Sensor Deployment Quiz

When using sensors which of the following is considered good practice? Check all the true statements:

- Set the IDS level to the highest sensitivity to detect every attack
- Monitor both outbound and inbound traffic
- Use a shared network resource to gather NIDS data
- NIDS sensors are not turnkey solutions, system administrators must interpret alerts.

### 11.57 Snort

**Snort** is an open-source, easily configurable, lightweight network IDS that can be easily deployed on most nodes of a network, including end hosts, servers, and even routers.

Snort can perform real-time packet capture, protocol analysis, and packet content-searching, and it consumes only a small amount of memory and processor time to perform its tasks.

Snort can detect a variety of intrusions based on the rules configured by a sysadmin. Helpfully, there is a community that maintains an extensive set of Snort rules that can be further configured by sysadmins for the needs of their networks.

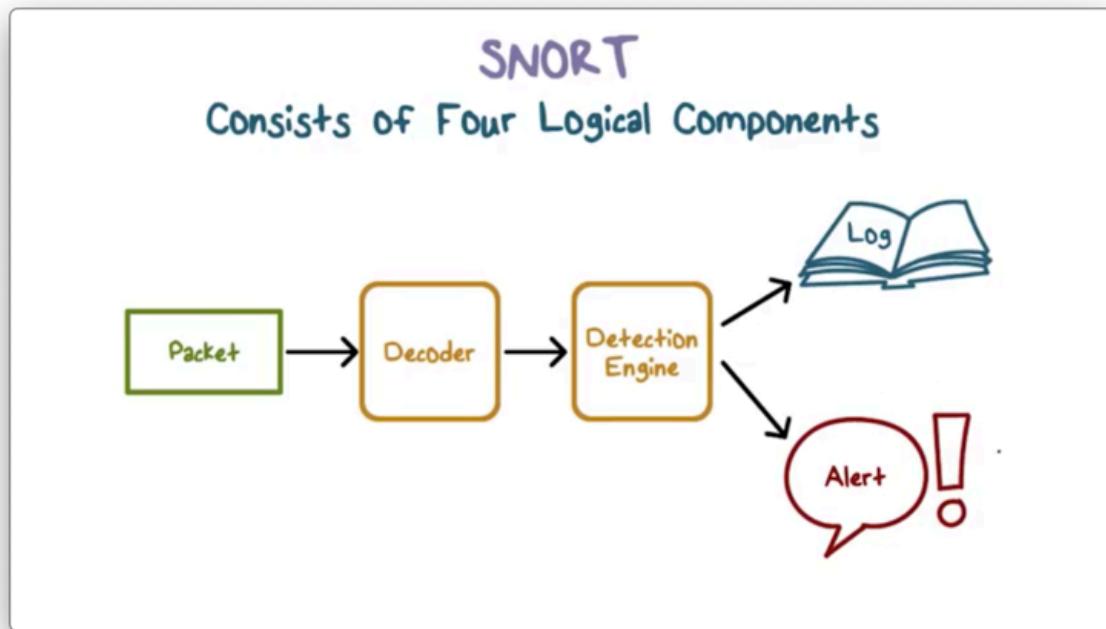
### 11.58 Snort Components

The Snort system has four logical components.

The **packet decoder** processes each captured packet to identify the protocol headers at the data link, network, transport, and application layers.

The **detection engine** performs the actual work of intrusion detection and checks each packet against a set of rules to determine if the packet matches a characteristic of a rule. If the engine finds a match, it triggers the action specified by the matching rule or rules; otherwise, it discards the packet.

Each rule specifies what **logging** or **alerting** steps the system should take. The logger stores the detected packet in a human-readable format. A sysadmin can then use the log files for later analysis. Alternatively, the system can send alerts to a file, database, or email, among other options.

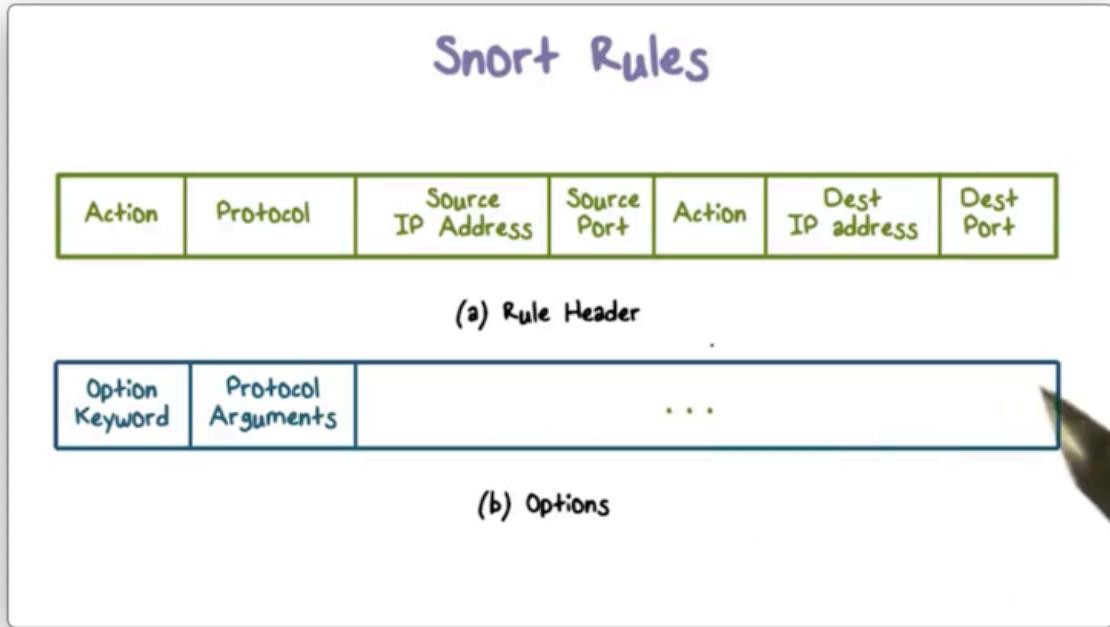


### 11.59 Snort Configuration

Snort can be configured as inline or passive. In passive mode, it merely copies and monitors traffic, and the traffic does not pass through it directly. In other words, in passive mode, Snort is configured for intrusion detection only.

### 11.60 Snort Rules

Snort uses a simple and flexible rule definition language. Each rule consists of a row header and a number of options.



## 11.61 Snort Rules Options

We specify our intrusion detection logic in the rule options, of which there are four main categories.

**Metadata** options allow us to provide information about the rule but do not have any effect on intrusion detection.

We specify the logic to examine the packet payload in the **payload** options, and we specify the logic to examine the packet headers in the **non-payload** options.

**Post-detection** options allow us to specify actions that the system will fire after a match has occurred, such as storing the packet information in a table so that we can correlate it with other packets.

## 11.62 Snort Rule Actions

The most important field in the Snort rule header is the **action** field, which tells Snort what to do when a packet matches a rule.

The following table describes the possible actions, the last three of which are only available in inline mode.

## Snort Rule Actions

Action	Description	Inline Mode Only
alert	Generate an alert using the selected alert method, and then log the packet.	
log	Log the packet.	
pass	Ignore the packet.	
activate	Alert and then turn on another dynamic rule.	
dynamic	Remain idle until activated by an activate rule, then act as a log rule.	
drop	Make iptables drop the packet and log the packet.	X
reject	Make iptables drop the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.	X
sdrop	Make iptables drop the packet but does not log it.	X

To summarize, a Snort rule includes a header and multiple options. Each option consists of an option keyword, which defines the option, followed by arguments, which specify the details of the option.

## Snort Rule Actions

Action	Protocol	Source IP Address	Source Port	Action	Dest IP address	Dest Port
--------	----------	-------------------	-------------	--------	-----------------	-----------

(a) Rule Header

Option Keyword	Protocol Arguments	...
----------------	--------------------	-----

(b) Options

### 11.63 Snort Rule Example

Typically, the root user account is only used for specific privileged operations, such as backing up filesystems and setting up subnetworks. A root account rarely sends email, and such an event should trigger an alert.

Here is an example of how Snort can capture and respond to this event.

**Snort Rule Example:**



```
alert tcp any any -> 192.168.1.0/24 25  
(content: "mail from: root"; msg: "root  
users attempts to send an email";)
```

It looks for traffic to the SMTP port on any host in the /24 subnet and checks if the content of the email contains “mail from root”, which indicates that a root user is attempting to send email. Snort then sends an alert with the following message: “root user attempts to send an email”.

The content keyword used above is one of the more important features of Snort. It allows sysadmins to set rules that search for specific content in the packet payload and then trigger responses based on that data.

### 11.64 Snort Quiz

 **SNORT Quiz**

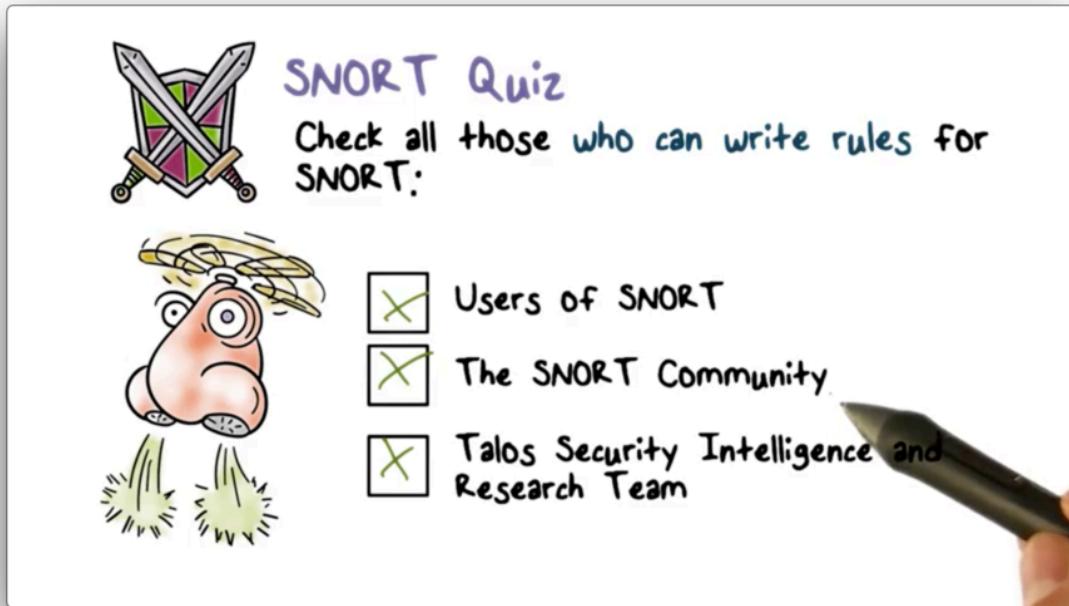
Check all those who can write rules for SNORT:



- Users of SNORT
- The SNORT Community
- Talos Security Intelligence and Research Team



### 11.65 Snort Quiz Solution



### 11.66 Honeypots

Honeypots are another component of the intrusion detection architecture. **Honeypots** are decoy systems designed to lure attackers away from critical systems.

By diverting attackers from valuable systems to honeypots, we can observe what they are trying to do to our systems and networks. Based on that information, we can develop strategies to respond to their attacks.

A honeypot system gives the appearance that any attack against it is successful. This apparent ease of compromise attracts attackers to a honeypot and keeps them there, which allows us to gather more information about their attacks.

Typically, a honeypot is filled with fabricated information to make it appear as if it is a valuable system on the network. Most importantly, a honeypot is not a real system used by any real user.

A honeypot system is instrumented with monitors and event loggers so that any access to or activity on the honeypot system is logged.

Since a honeypot is not a real system and has no production value, any access to it is not legitimate. Most likely, any inbound connection to a honeypot is a network scan or a direct attack. Any outbound

traffic from the honeypot indicates that the system is most likely compromised.

### **11.67 Honeypots Classification**

A **low interaction** honeypot typically emulates some network services, such as the web server, but does not provide a full version of the service. For example, an emulated web server may be able to “speak” HTTP but might not contain all the web contents and server-side programs.

A low interaction honeypot is typically sufficient to detect basic network scans and probes and warn of imminent attacks. On the other hand, these weakly emulated services may not fool a sophisticated attacker, who might realize early on that these services are not real.

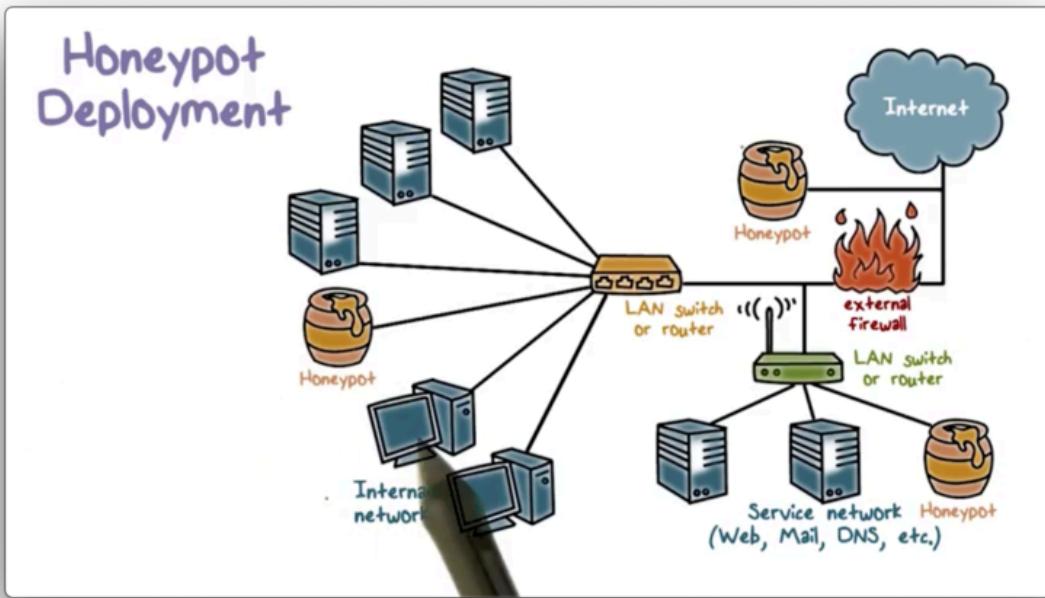
A **high interaction** honeypot essentially replicates a real server or workstation in terms of operating systems, services, and applications. They look realistic, and they can be deployed alongside the real servers and workstations.

Since a high interaction honeypot mimics a system, an attacker might attack it for an extended time without realizing that it is a honeypot. This large attack window gives us more time to learn about the attacks.

On the other hand, making a honeypot look like a real server or workstation is quite challenging. For example, we must emulate user activities and network traffic on the honeypot, which requires a significant amount of programming effort and data storage.

### **11.68 Honeypot Deployment**

We can deploy honeypots to a variety of locations within a network.



A honeypot placed outside of the external firewall is useful for checking attempts to scan or attack the internal network. The main advantages of placing the honeypot at this location are that first, it has no side effects, and second, it reduces the amount of attack traffic the firewall has to process. However, a honeypot at this location cannot trap internal attackers.

We can also place a honeypot in the DMZ, alongside public-facing servers, to trap attacks targeted at these services. A honeypot at this location may not be able to trap interesting attacks, because a DMZ is typically not accessible outside of a well-defined set of public-facing services. As a result, the external firewall can block virtually all attack traffic destined for the DMZ.

Finally, we can also place a honeypot in the internal network alongside servers and workstations. The main advantage of this configuration is that it can catch internal attacks. Additionally, it can detect a misconfigured firewall that forwards non-permissible traffic from the Internet to the internal network.

On the other hand, unless we can trap the attacker entirely in the honeypot, the attacker may be able to reach other internal systems.

As well, if we want to continue to attract and trap an attacker, we must continue to allow attack traffic from the Internet to reach the honeypot. This step might involve opening up the external firewall to allow this attack traffic to enter, which carries a considerable security risk.

### 11.69 Honeypot Quiz



#### Honeypot Quiz

Put True (T) next to each true statement and False (F) next to each false statement.



- A common location for a NIDS sensor is just inside the external firewall
- A Honeypot can be a workstation that a user uses for work
- There is no benefit of deploying a NIDS or Honeypot outside of the external firewall

### 11.70 Honeypot Quiz Solution



### Honeypot Quiz

Put True (T) next to each true statement and False (F) next to each false statement.

- T A common location for a NIDS sensor is just inside the external firewall
- F A Honeypot can be a workstation that a user uses for work
- F There is no benefit of deploying a NIDS or Honeypot outside of the external firewall



### 11.71 Evaluating IDS

We typically use accuracy metrics to evaluate the detection algorithm. We use **detection rate** or **true positive rate** to measure how well an IDS can detect an intrusion. More specifically, the detection rate measures how likely the IDS outputs an alert, given that an intrusion is present. We can also use the **false negative rate**, the probability that, given an intrusion, an alert is not triggered.

Alternatively, we can look at the **false alarm rate**. This rate measures the probability that the IDS generates an alert, given that there is no intrusion. We can also use the **true negative rate**, the probability that an alert is not triggered given an intrusion is not present.

We might also want to know the **Bayesian detection rate** of an IDS; that is, the probability that an intrusion has occurred, given that an alert is raised.

We can more formally summarize these metrics, using the probability  $A$  of an alarm being raised and the probability  $I$  of an intrusion occurring.

## Evaluating IDS

### Algorithm

- Alarm/positive: A; Intrusion: I
- Detection (true positive) rate:  $P(A|I)$ 
  - False negative rate  $P(\neg A|I)$
- False alarm rate:  $P(A|\neg I)$ 
  - True negative rate  $P(\neg A|\neg I)$
- Bayesian detection rate:  $P(I|A)$



The detection rate is the probability of A, given I. The false negative rate is the probability of not A, given I. The false alarm rate is the probability of A, given not I. The true negative rate is the probability of not A, given not I. The Bayesian detection rate is the probability of I given A.

In addition to the detection algorithms, we can also evaluate IDS's in terms of their system architecture. We want a scalable IDS that can function in high-speed networks. Additionally, we want a resilient IDS that is not easily disabled by attacks that target the IDS.

### 11.72 Bayesian Detection Rate

The following formula defines the Bayesian detection rate, given probability A of an alert and prior probability I of an intrusion.

## Bayesian Detection Rate



$$P(I|A) = \frac{P(I)P(A|I)}{P(I)P(A|I) + P(\neg I)P(A|\neg I)}$$

**P(I)** is prior probability of attacks: this is the probability of intrusion evidences in the data.

We can calculate  $I$  by counting the number of packets that flow through our network during a particular time period and taking the percentage of packets that contain evidence of intrusion activities.

There is an interesting phenomenon involving the Bayesian detection rate, called the base-rate fallacy. The **base-rate fallacy** states that even if the false alarm rate  $P(A|\neg I)$  is very low, the Bayesian detection rate  $P(I|A)$  is still low if the base-rate  $P(I)$  is low.

For example, given a detection rate  $P(A|I) = 1$ , a false alarm rate  $P(A|\neg I) = 10^{-5}$ , and a base rate of intrusion  $P(I) = 2 * 10^{-5}$ , the Bayesian detection rate  $P(I|A)$  is only 66%.

This low base rate may or may not be realistic, depending on where you measure it. For example, tens of millions to hundreds of millions of packets may pass through the entire network, and only a few hundred might contain intrusion activity. Thus, measured from the perspective of the entire network, this base rate can be quite realistic.

One way we can mitigate the base rate fallacy is to reduce the false alarm rate to zero, or as close to zero as possible; indeed, this is one of the main goals of modern IDS vendors.

Alternatively, we can deploy the IDS to the appropriate layer such that, at that layer, the base rate is sufficiently high. Modern-day IDS's use a hierarchical architecture to achieve this high base rate.

We can also use multiple independent models. This approach is similar to medical diagnosis, where multiple tests are used to reduce the overall false positive rate and increase the Bayesian detection rate.

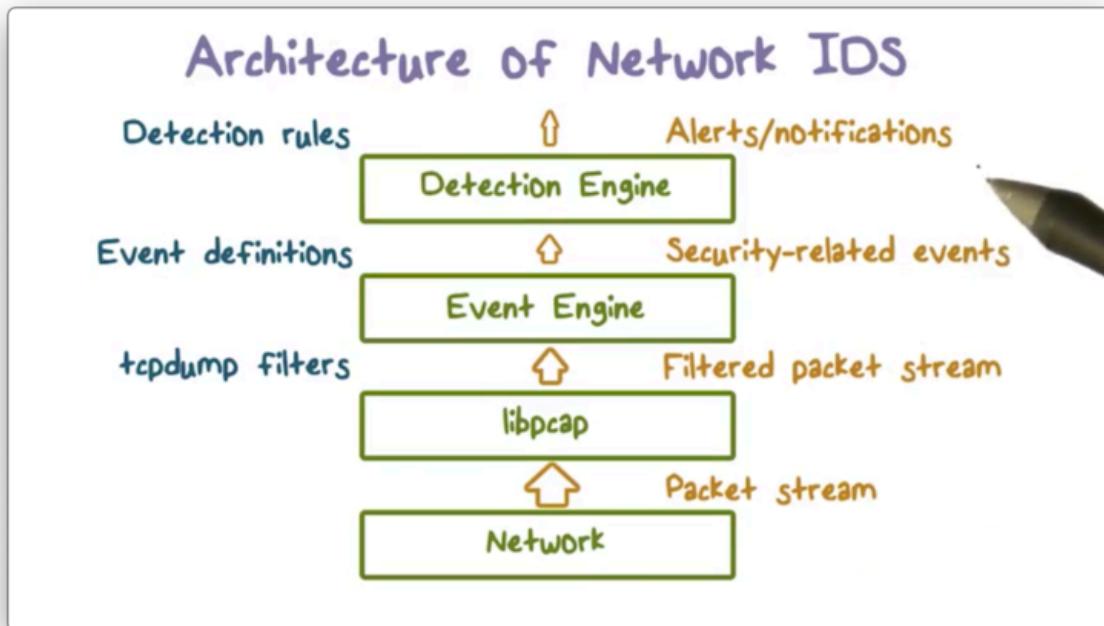
### 11.73 Architecture of Network IDS

With the Bayesian detection rate and the base-rate fallacy in mind, let's discuss the system architecture of a network IDS.

Typically, the volume of packet data in a network is enormous - on the order of tens of millions of packets per day - and only a few might involve any activity related to intrusion. Therefore, according to the base-rate fallacy, if we apply detection algorithms at the packet level, we will see meager Bayesian detection rates.

Instead, we should apply detection models to a set of data that has a higher base rate, which requires that we filter out packet data unlikely to be related to intrusion activity.

First, we can apply filters to the packet data by, for example, instructing `libpcap` to only capture packets destined for certain services. Second, an event engine analyzes the filtered packet data and summarizes them into security-related events, such as failed logins. Finally, the system applies detection models to the data contained in each security-related event.



The system decreases the volume of packet data at each step. Therefore, as long as we capture the intrusion evidence in the event data, the base rate is going to be much higher because the total number of packets is going to be much lower. As a result, the IDS model applied to the event data will yield a higher Bayesian detection rate.

### 11.74 IDS Quiz



#### IDS Quiz

Check any item that is true. To improve detection performance, an IDS should:

- Reduce false alarm rate while detecting as many intrusions as possible
- Apply detection models at all unfiltered packet data directly
- Apply detection models at processed event data that has higher base rate



### 11.75 IDS Quiz Solution



#### IDS Quiz

Check any item that is true. To improve detection performance, an IDS should:

- T Reduce false alarm rate while detecting as many intrusions as possible
- F Apply detection models at all unfiltered packet data directly
- T Apply detection models at processed event data that has higher base rate



### 11.76 Eluding Network IDS

Now let's talk about how an attacker can evade an IDS so that their attack can proceed undetected.

Recall that a network IDS performs passive monitoring; that is, it takes a copy of the network traffic destined for the end host and analyzes it. Therefore, for the IDS to detect the intrusion happening at the end host, it must see the same traffic as the end host.

However, this parity is not always present, and an attacker can exploit this to evade the IDS. The reason that the IDS and the end host see different traffic is because they are using two different operating systems that process traffic in different ways.

In particular, TCP/IP protocol specifications have ambiguities that lead to different implementations in different operating systems. As a result, an IDS running on Unix and an end host running on Windows may not process certain packets in precisely the same fashion.

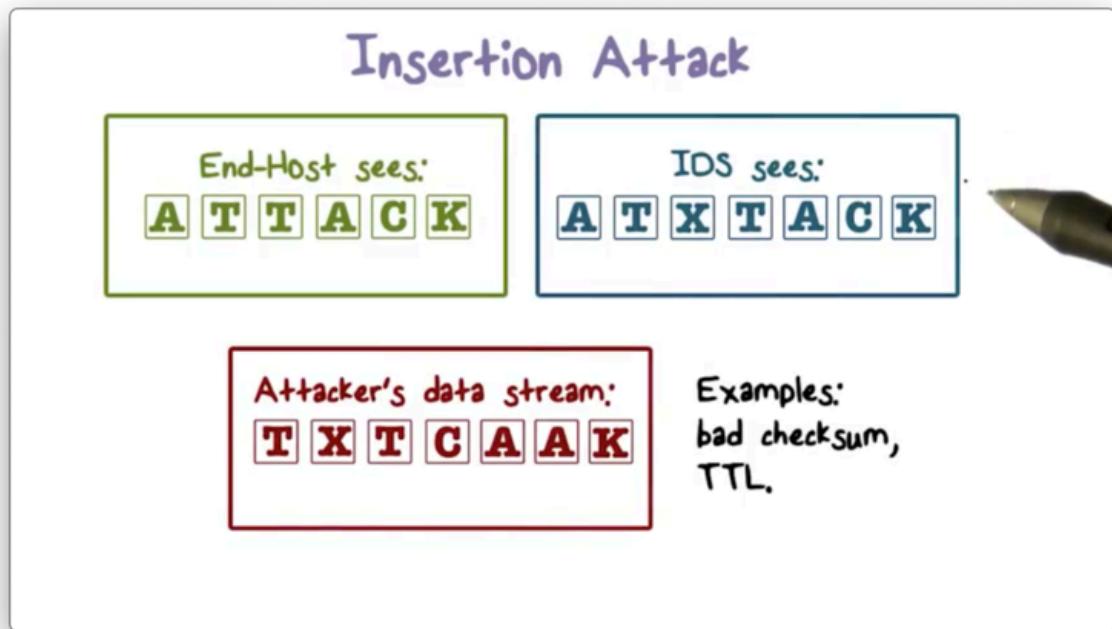
For example, options such as time-to-live (TTL), or error conditions associated with fragments and checksums are handled in different ways in different operating systems.

By exploiting these differences, the attacker can slip attack traffic past the IDS undetected while still impacting the destination end host.

### 11.77 Insertion Attack

An attacker can insert misleading data into the packet stream to cause the IDS to miss an attack. For example, an attacker might include a packet with a bad checksum value into an otherwise malicious stream of packets. The IDS may accept this packet, but the end host might reject it. As a result, the end host receives the attack, and yet the IDS doesn't detect it.

Here is an illustration of this **insertion attack**.

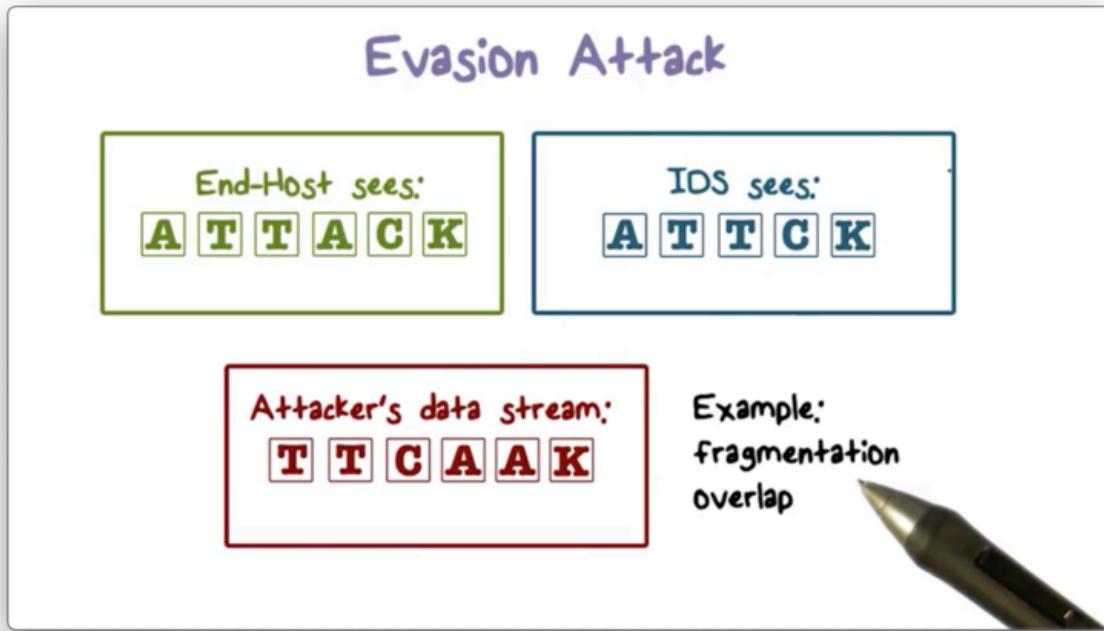


The attacker sends the out-of-order packets that both the IDS and the end host reassemble according to sequence numbers. One of the packets **X** has a bad checksum value. The IDS accepts **X**, and sees the packet stream **ATXTACK**, which looks benign. On the other hand, the end host rejects **X** and, as a result, sees the **ATTACK**.

### 11.78 Evasion Attack

The attacker can also hide part of the attack to evade IDS detection. An IDS may discard a packet fragment that overlaps with a previous fragment, but an end host may accept both fragments. Therefore, an attacker can send an attack embedded in overlapping packet fragments through the IDS to the end host.

Here is an illustration of this **evasion attack**.



The attacker sends the out-of-order packets that both the IDS and the end host reassemble according to sequence numbers. In this scenario, the two A's contain overlapping fragments; as a result, the IDS drops the second A, and so it only sees the stream ATTCK. On the other hand, the end host accepts both fragments, even though they overlap. Therefore, the end host sees the ATTACK.

### 11.79 DOS Attacks on Network IDS

Attackers can also use denial of service (DOS) attacks to disrupt the network intrusion detection process. Similar to DOSing a network server, an attacker can send a lot of traffic to an IDS to process, which results in the exhaustion of resources such as CPU, memory, and network bandwidth.

As a result, the IDS may not be able to analyze subsequent traffic, and such traffic may contain actual attacks. Knowing this, the attacker can first disable the IDS through a DOS attack and then launch the real attack.

Another attack approach involves abusing the reactive nature of intrusion detection. When the IDS outputs an alert, a security admin must analyze the alert. Therefore, the attacker can overload the system by sending a lot of traffic that triggers alerts; for example, purposely crafting and sending packets that contain signatures of attacks.

The goal with this attack is first to overwhelm the response system and the security admins and then send the real attack traffic. The attack traffic will generate an alert but, because the security admins

are busy analyzing alerts of fake attacks, the alert will not be analyzed until it is too late.

### 11.80 Intrusion Prevention Systems (IPS)

In addition to intrusion detection systems, there are also **intrusion prevention systems** (IPS) on the market. Instead of merely sending alerts like an IDS, an IPS tries to block an attack when it detects malicious activity.

Similar to an IDS, an IPS can be deployed at the end host, the network perimeter, or some combination of different locations depending on network needs.

An IPS also uses similar detection algorithms as an IDS. For example, it can use anomaly detection algorithms to detect abnormal behavior and then stop such behavior.

The main difference between an IPS and a firewall is that while a firewall typically only uses simple signatures of attacks to block traffic, an IPS can use very sophisticated detection algorithms.

### 11.81 IDS Attack Quiz



### IDS Attack Quiz

Check any item that is true. To defeat an IDS, attackers can:

- Send a huge amount of traffic
- Embed attack in packets that cause non-uniform processing by different operating systems, e.g., bad checksum, overlapping fragments
- Send traffic that purposely matches detection rules
- Send a packet that would trigger a buffer-overflow in the IDS code



### 11.82 IDS Attack Quiz Solution



### IDS Attack Quiz

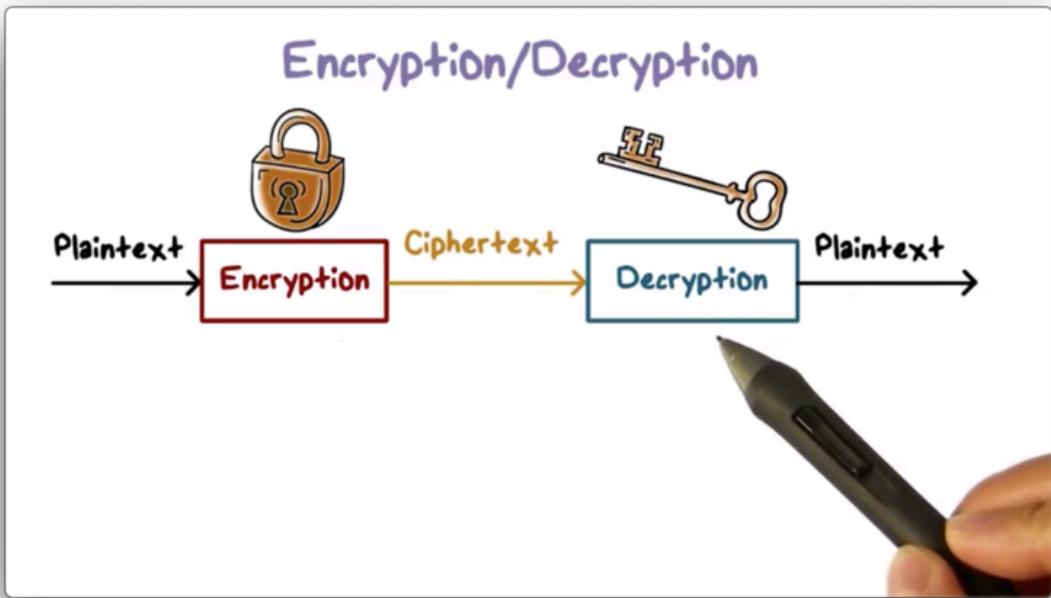
Check any item that is true. To defeat an IDS, attackers can:

- Send a huge amount of traffic
- Embed attack in packets that cause non-uniform processing by different operating systems, e.g., bad checksum, overlapping fragments
- Send traffic that purposely matches detection rules
- Send a packet that would trigger a buffer-overflow in the IDS code

## 12 Introduction to Cryptography

### 12.1 Decryption

**Encryption** is the process of converting data into a form that is unintelligible to the unintended or unauthorized party. We call the original data **plaintext**, and the unintelligible data **ciphertext**. An authorized party can reverse this process; that is, they can **decrypt** the ciphertext to reveal the plaintext.



The decryption process always recovers the original plaintext because there is a one-to-one mapping between plaintext and ciphertext.

Encryption protects data confidentiality because only the authorized party with the proper **secret key** can read the data.

Encryption also provides other security services. We can use encryption for integrity checking to ensure that no one has tampered with a message. We can also use encryption to verify the authorship of a message. Encryption also serves as a method by which one user can authenticate themselves with another.

## 12.2 Encryption Basics

Encryption has been used for thousands of years. There is evidence that ancient Egyptians used **ciphers** - encryption schemes - and one of the most famous ciphers is the letter-based [Caesar cipher](#) from Roman times.

The ancient ciphers were all *symmetric ciphers*, an encryption scheme that still enjoys widespread use today. *Asymmetric ciphers* are much newer, only having been invented in the late 1970s.

Most modern security protocols use both types of encryption. Parties wishing to communicate typically use asymmetric ciphers to authenticate themselves with one another and to establish and ex-

change a symmetric encryption key. Each party encrypts and decrypts subsequent communications using this key. Senders may also use asymmetric ciphers to sign the data to prove authenticity.

### **12.3 Attacks on Encryption**

Given that encryption plays such an essential role in security, we can expect that attackers are always trying to break our ciphers. The goal of these attackers is to recover the plaintext from the ciphertext or to uncover the encryption key.

An attacker can quickly get their hands on our ciphertext. For example, they may use a packet sniffing tool to capture encrypted packets you send over the Internet, and they might try to recover the plaintext from these packets. We should always assume that an attacker can capture the ciphertext we transmit over the Internet.

Instead of uncovering one piece of plaintext from one piece of ciphertext, the attacker may try to discover the encryption key so that they can decrypt all subsequent communications encrypted using that key. There are several attack methods to achieve this goal.

#### **12.3.1 Brute-force**

The simplest, most inefficient strategy is the **brute-force** method. In this attack, the attacker tries all possible keys one by one until they find the one that decrypts the ciphertext properly into plaintext.

How does the attacker know that the decryption has worked properly? Typically the attacker knows what the plaintext should resemble. For example, if the plaintext is an English sentence, then only the correct key can decrypt the ciphertext into text that reads like English.

This method is very inefficient because the number of possible keys can be huge. As a result, brute-force often requires an unfeasible amount of time to be successful.

#### **12.3.2 Cryptanalysis**

In cryptanalysis, the attacker knows the encryption algorithm and perhaps some characteristics of the data, such as the distribution of certain letters or words. An attacker can perform a much more directed search of the keyspace with cryptanalysis as opposed to brute-force.

#### **12.3.3 Implementation attacks**

The attacker can also exploit implementation or systems issues. For example, researchers were able to deduce the values of certain bits in an encryption key by observing the power consumption in a

cryptosystem. As a result, they were able to reduce the size of the keyspace significantly.

#### 12.3.4 Social-engineering attacks

The weakest link in a security system is often the naive users who can be exploited using social engineering tricks. For example, an attacker can call an unsuspecting user and pretend to be a sysadmin who has forgotten their key.

#### 12.4 Encryption Attack Quiz



### Encryption Attack Quiz

If the only form of attack that could be made on an encryption algorithm is brute-force, then the way to counter such attacks would be to...

- use a longer key length
- use a shorter key length
- use a more complex algorithm
- use a harder to guess key



## 12.5 Encryption Attack Quiz Solution



### Encryption Attack Quiz

If the only form of attack that could be made on an encryption algorithm is brute-force, then the way to counter such attacks would be to...

- use a longer key length
- use a shorter key length
- use a more complex algorithm
- use a harder to guess key



In a brute-force attack, the attacker must try all potential keys. The only way to make this task more difficult is to increase the length of the key, thus increasing the size of the keyspace.

## 12.6 Simple Ciphers Quiz



### Simple Ciphers Quiz

Use Caesar's cipher to decode the message:

LQIRUPDWLRQ VHFXULWB



Enter your answer in the text box:

## 12.7 Simple Ciphers Quiz Solution



### Simple Ciphers Quiz

Use Caesar's cipher to decode the message:

LQIRUPDWLRQ VHFXULWB

Enter your answer in the text box:

Information Security



Since “A” maps to “D”, “B” maps to “E”, and so forth, we can just “rewind” each letter in the ciphertext by three to obtain the plaintext.

## 12.8 Simple Ciphers

Caesar’s cipher - an example of a **shift cipher** - maps each letter to another letter by shifting it a fixed amount. If we represent each letter in the alphabet as a number - A = 1, B = 2, …, Z = 26 - then we can represent this encryption scheme for a given `letter` and a given `shift` as

```
1 (letter + shift) mod 26
```

The secret key in this scheme is the value of `shift`. Since there are only 26 possible keys (any shift higher than 26 wraps back around), an attacker only needs to try at most 26 keys before being able to decrypt any message encrypted with Caesar’s cipher.

A generalization of this scheme allows arbitrary mapping of one letter to another; that is, this scheme no longer requires shifting each plaintext letter by the same fixed amount. These generalized ciphers are referred to as **substitution ciphers** or **monoalphabetic ciphers**.

The substitution alphabet is the secret key in this scheme. The number of possible alphabets is vast. If

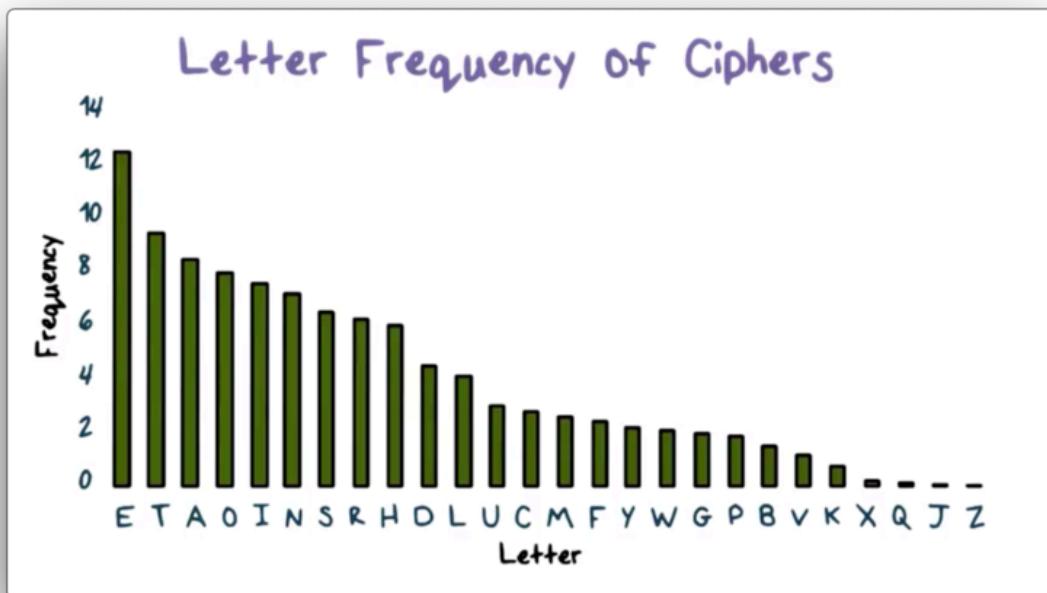
I wanted to create such an alphabet, I would have 26 options for mapping “A”, 25 options for mapping “B”, and so forth. Put another way, there are  $26!$  ( $2^{88}$ ) possible alphabets. An attacker cannot brute-force this scheme.

Instead of attempting to try every key, an attacker can analyze the statistical frequency of letters in the ciphertext. For example, in English, the most frequently used letter is “E”. If the most common letter in the ciphertext is “X”, then there is a high probability that “E” maps to “X”.

An attacker can use statistical analysis to decrypt the message directly or at least drastically reduce the size of the keyspace over which they subsequently run a brute-force attack.

## 12.9 Letter Frequency of Ciphers

Here is the frequency distribution of English letters.



Using this frequency distribution, we should be able to make some educated guesses to decipher the following ciphertext:

1	IQ	IFCC	VQQR	FB	RDQ	VFLLCQ	NA	RDQ
2	CFJWHWZ	HR	BNNB	HCC	HWWHBSQVQBRE			
3	HWQ	VHLQ						

To start, we notice that the letter “Q” is the most frequent in our ciphertext. In English, the most common letter is “E”, so there is a high probability that “Q” is the plaintext letter “E”.

Then we can look at the three-letter words, [RDQ](#) and [HWQ](#), which both end in “E”. It is likely that one of these words is “ARE”, while the other is “THE”.

We also have the word [HR](#), which can be “AT”. If so, then [HWQ](#) is “ARE” and [RDQ](#) is “THE”. If “H” is “A”, then the word [HCC](#) is likely “ALL”.

If we continue with this process, using both the frequency distribution of the letters in the English language along with our knowledge of English words to help us, we can uncover the following plaintext:

```
1 WE WILL MEET IN THE MIDDLE OF THE  
2 LIBRARY AT NOON ALL ARRANGEMENTS  
3 ARE MADE
```

In practice, we might also look at the frequency distribution of letter pairs and even triples in addition to single letters.

## 12.10 Monoalphabetic Cipher Quiz



**Monoalphabetic Cipher Quiz**

Try to decipher this method using the Monoalphabetic Cipher:

WAIT IT WAS SOD

Enter your answer in the text box:

### 12.11 Monoalphabetic Cipher Quiz Solution



#### Monoalphabetic Cipher Quiz

Try to decipher this method using the Monoalphabetic Cipher:

WAIT IT WAS SOD

Enter your answer in the text box:

This is the end

### 12.12 Vigenere Cipher

While a substitution cipher uses a single alphabet, a **polyalphabetic cipher** uses multiple substitution alphabets. The Vigenere cipher is the most well-known polyalphabetic cipher.

We can represent the Vigenere translation with the following matrix,  $M$ , whereby the ciphertext for a given plaintext  $P$  and key  $K$  resides at column  $P$ , row  $K$  in  $M$ .

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	D
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	C
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	B
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Suppose we want to encrypt the plaintext "ATTACKATDAWN" using the Vigenère cipher with "LEMON" as the key. Since Vigenère encryption proceeds characterwise, the key must be as long as the plaintext. We generate the **keystream** "LEMONLEMONLE" by repeating the key until the required number of characters are present.

To encrypt "A", we look at column "A" and row "L" of the matrix. The first character of the ciphertext is "L". To encrypt "T", we look at column "T" and row "E" of the matrix. The second character of the ciphertext is "X". To encrypt "T", we look at column "T" and row "M" of the matrix. The third character of the ciphertext is "F".

We can continue in this fashion to transform "ATTACKATDAWN" into "LXFOPVEFRNHR".

### 12.13 Vigenere Cipher Quiz



#### Vigenere Cipher Quiz

What weaknesses can be exploited in the Vigenere Cipher?



- It uses a repeating key letters
- It requires security for the key, not the message
- The length of the key can be determined using frequency analysis

### 12.14 Vigenere Cipher Quiz Solution



#### Vigenere Cipher Quiz

What weaknesses can be exploited in the Vigenere Cipher?

- It uses a repeating key letters
- It requires security for the key, not the message
- The length of the key can be determined using frequency analysis



### 12.15 What Should be Kept Secret?

While we should always keep our encryption keys a secret, our encryption algorithms should be public.

According to [Kerckhoff's principle](#):

A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

In general, we keep our encryption algorithms open so that they can be analyzed and improved by the broader community. More importantly, we don't have to rely on the secrecy of the algorithm for its security - the so-called "security by obscurity" principle. In practice, we should exclusively use the widely-known algorithms and standards.

### 12.16 Types of Cryptography

**Secret key cryptography** uses a single key for encryption and decryption, which means that the sender and the receiver must possess the same key.

**Public-key cryptography** uses two keys - a public and a private key - that are linked together mathematically. A user keeps their private key secret and announces their public key to other users.

The public key is used for encryption, and the private key is used for decryption. For example, Alice can use Bob's public key to encrypt a message that only Bob can decrypt because only Bob has the corresponding private key that can decrypt the message correctly.

The private key is also used for signing a message, and the public key can be used to verify the signature. For example, Alice can sign a message using her private key, and anyone knowing her public key can verify that only Alice can produce this signature.

## 12.17 Hash Functions

Hash functions are a third class of cryptographic functions. A **hash function** computes a fixed-length output - typically in the range of 128-512 bits - from a message of any size. This output is commonly referred to as a *hash* or *message digest*. Hash functions do not use keys.

We typically use hash functions for authentication and message authenticity/integrity. To provide these services, hash functions must satisfy the following properties.

### 12.17.1 Efficiency

A hash function should be able to compute a hash efficiently enough to make software and hardware implementations of the function practical.

### 12.17.2 One-way function

For a given hash function  $H$  and a given hash value  $v$ , finding the original message  $m$ , such that  $H(m) = v$  should be computationally infeasible.

Suppose Alice wants to authenticate herself to Bob. She can hash a secret  $S$  that she shares with Bob and send the hashed value to him. Bob can verify that Alice sent the message by hashing their shared secret himself and comparing it with the value he received.

Because Alice transmits the hash value over the Internet, we must assume that an attacker can intercept the message. If the hash function does not satisfy the one-way property, the attacker can reverse the function and recover  $S$  from its hash.

If the attacker knows  $S$ , they can impersonate Bob to Alice or Alice to Bob.

### 12.17.3 Weak collision resistance

Given a hash function  $H$  and a message  $m_1$ , it should be computationally infeasible to find another message  $m_2$  such that  $H(m_2) = H(m_1)$ .

This property is essential for message authenticity and integrity protection.

Suppose Alice sends a message  $M$  to Bob, and she wants to make sure Bob knows that she is the real author of  $M$ . Alice can do this by sending  $M$  along with a signed hash of  $M$ .

If the weak collision-resistant property is not present, an attacker may be able to craft a different message  $M'$ , such that the hash of  $M'$  is the same as the hash of  $M$ .

If Bob receives  $M'$  and the original signed hash of  $M$ , Bob has no way of knowing that the message is not from Alice.

### 12.17.4 Strong collision-resistant

Given a hash function  $H$ , it should be computationally infeasible to find two messages  $m_1$  and  $m_2$  such that  $H(m_2) = H(m_1)$ .

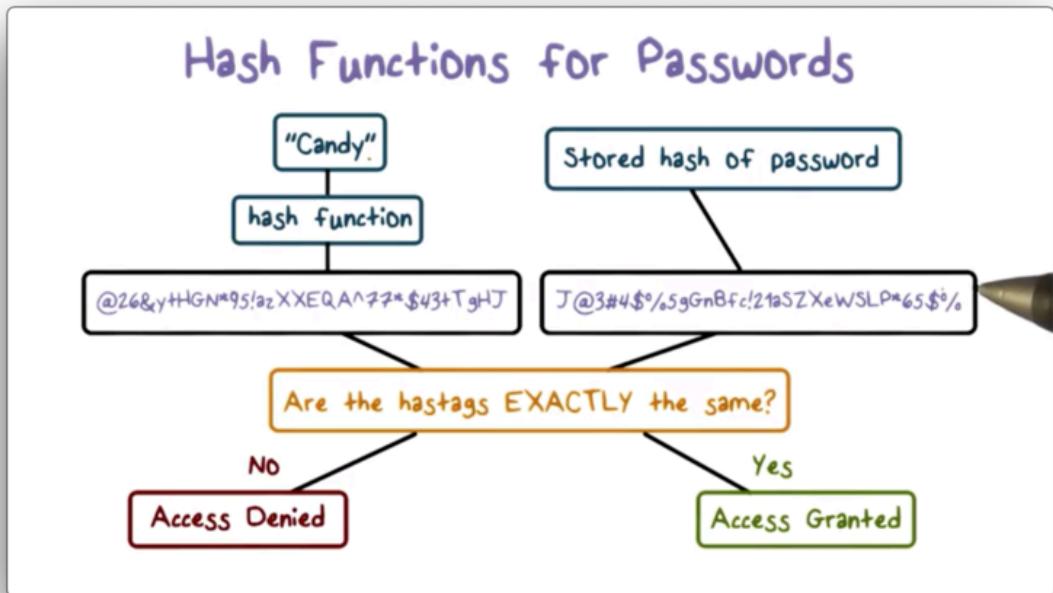
This property is stronger because it prevents an attacker from coming up with any two different messages that have the same hash value.

Suppose Alice owes Bob money. Bob can write an IOU message that Alice can hash and sign using her private key. If Bob can find two different messages that have the same hash value, one that requires Alice to pay a small amount, and another that requires her to pay a large amount, Bob may send Alice the message for the smaller amount and later claim that Alice owes him the larger amount.

## 12.18 Hash Functions for Passwords

The one-way property of hash functions - a hash function should be easy to compute, but impossible to invert - makes them a great candidate for password verification.

When a user authenticates themselves to a system, they often supply a password. The system hashes the supplied password and compares this hash against the saved hash of the password for that user. The system grants access if the two values match, denying access otherwise.



The advantage of this scheme is that the system only has to store the hashed passwords. If the system chose to store the cleartext passwords, then anyone who has access to the system can impersonate anyone else registered on the system after reading the password file.

If an attacker gains access to a system, they can steal the hashed passwords and compare the hashes against a dictionary of commonly-used passwords and their hashes to find matches. We must avoid using common passwords like “password” and “123456”.

### 12.19 Hash Function Quiz



#### Hash Function Quiz

Which of the following characteristics would improve password security?



- Use a one-way hash function
- Should not use the avalanche effect
- Should only check to see that the hash function output is the same stored output

## 12.20 Hash Function Quiz Solution



### Hash Function Quiz

Which of the following characteristics would improve password security?



Use a one-way hash function



Should not use the avalanche effect



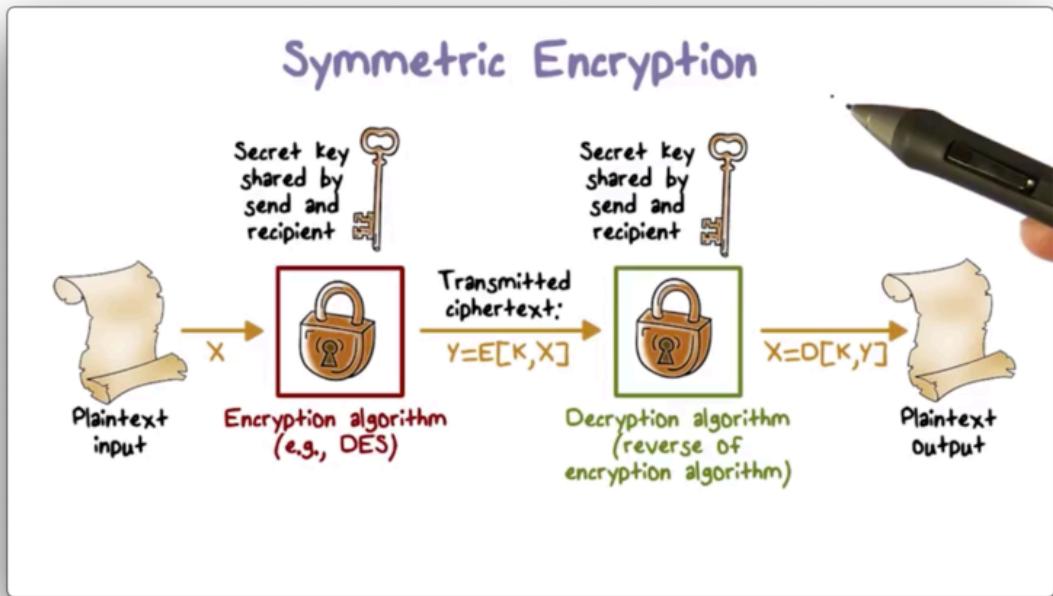
Should only check to see that the hash function output is the same stored output



The **avalanche effect** states that a small change in the input to a hash function causes a large change to the output. We want this in place as a way to obscure similar passwords. Without the avalanche effect, an attacker may be able to deduce password [A](#) from its hash value if he knows that the hash of a string [B](#) is similar to [A](#)'s hash.

## 12.21 Symmetric Encryption

Symmetric encryption uses the same key for both encryption and decryption. The encryption algorithm takes the plaintext and the key as input and produces the ciphertext using substitution and permutation. The decryption algorithm reverses the encryption process, reproducing the original plaintext from the key and the ciphertext.



## 12.22 Comparison of Encryption Algorithms

The most important symmetric algorithms are the [Data Encryption Standard](#) (DES) block cipher and the [Advanced Encryption Standard](#) (AES) block cipher. A **block cipher** encrypts plaintext in fixed-size blocks and produces ciphertext blocks of equal size. DES and AES use different key sizes and block lengths.

## Comparison of Encryption Algorithms

	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

DES = Data Encryption Standard

AES = Advanced Encryption Standard



Since a longer key indicates a larger keyspace, DES and AES can be distinguished by the amount of time necessary to successfully conduct a brute-force attack. The following table shows how much time is required to brute-force various ciphers with different key lengths.

## Comparison of Encryption Algorithms

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at $10^9$ descryptions/s	Time Required at $10^{13}$ descryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	$2^{55} \text{ ns} = 1.125 \text{ years}$	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	$2^{127} \text{ ns} = 5.3 \times 10^{21} \text{ years}$	$5.3 \times 10^{17} \text{ years}$
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	$2^{167} \text{ ns} = 5.3 \times 10^{33} \text{ years}$	$5.8 \times 10^{29} \text{ years}$
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	$2^{191} \text{ ns} = 5.3 \times 10^{40} \text{ years}$	$9.8 \times 10^{36} \text{ years}$
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	$2^{255} \text{ ns} = 5.3 \times 10^{60} \text{ years}$	$1.8 \times 10^{56} \text{ years}$

Note that DES - now seen as insecure - is breakable within an hour using a supercomputer. Ciphers with key lengths of 128 bits or greater are effectively unbreakable using modern processing power.

### 12.23 Symmetric Encryption Quiz

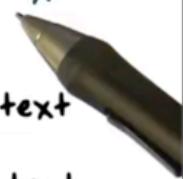


#### Symmetric Encryption Quiz

Select the correct definition for each type of attack:

- A. A method to determine the encryption function by analyzing known phrases and their encryption
- B. Analyzing the effect of changes in input on the encrypted output
- C. Compare the ciphertexts with its known plaintext
- D. A method where a specific known plaintext is compared to its ciphertext

- known-Plaintext attacks
- chosen-Plaintext attacks
- differential cryptanalysis
- linear cryptanalysis



### 12.24 Symmetric Encryption Quiz Solution

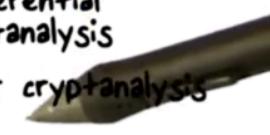


### Symmetric Encryption Quiz

Select the correct definition for each type of attack:

A. A method to determine the encryption function by analyzing known phrases and their encryption  
B. Analyzing the effect of changes in input on the encrypted output  
C. Compare the ciphertexts with its known plaintext  
D. A method where a specific known plaintext is compared to its ciphertext

d known-Plaintext attacks  
 c chosen-Plaintext attacks  
 b differential cryptanalysis  
 a linear cryptanalysis

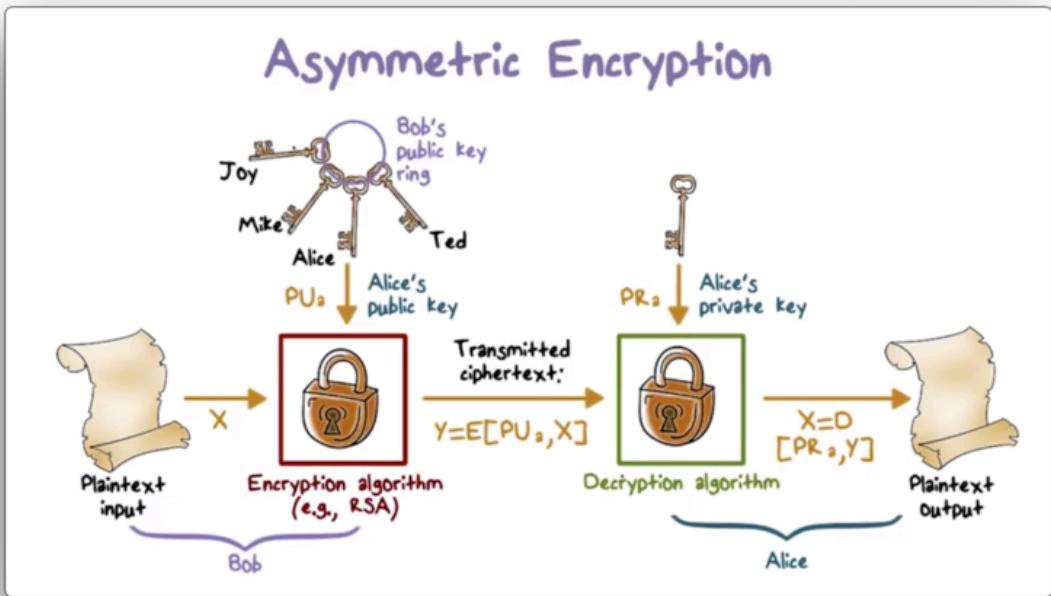


### 12.25 Asymmetric Encryption

While symmetric encryption uses the same key for encryption and decryption, **asymmetric encryption** uses two keys: one for encryption and the other for decryption. The two keys are paired together mathematically such that if one key encrypts a message, only the other key can decrypt it.

To communicate using asymmetric encryption, a user must first generate a pair of keys. The **public key** can be sent to other users, while the **private key** must be known only to the user. A user might have a collection of public keys, one for each user with whom they communicate regularly.

If Bob wants to send a private message to Alice, Bob must first obtain Alice's public key. With Alice's public key in hand, Bob encrypts his message to Alice using an asymmetric encryption algorithm such as **RSA**. Bob then transmits the ciphertext to Alice, which she can decrypt using her private key.

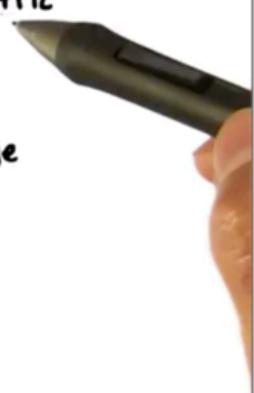


### 12.26 Asymmetric Encryption Quiz



#### Asymmetric Encryption Quiz

Check all tasks for which asymmetric encryption is better:



- provide confidentiality of a message
- securely distribute a session key
- scalability

### 12.27 Asymmetric Encryption Quiz Solution



#### Asymmetric Encryption Quiz

Check all tasks for which asymmetric encryption is better:

- provide confidentiality of a message
- securely distribute a session key
- scalability

### 12.28 Digital Signatures

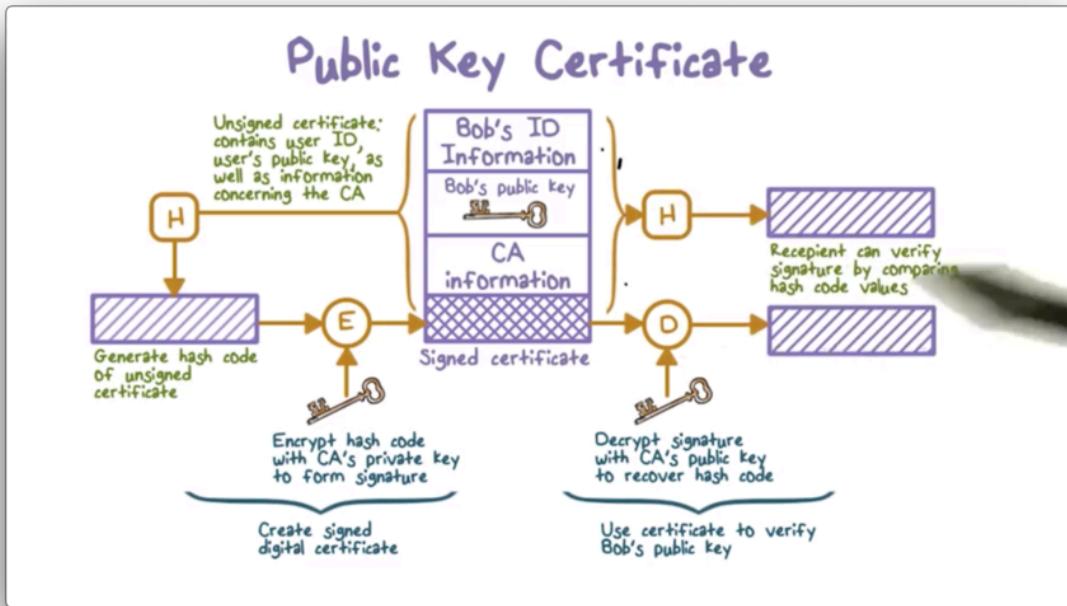
The public key in public-key encryption is genuinely public. Any user can broadcast their public key to the world for any other user to consume. Although this approach is very convenient, it has a significant weakness: any user can forge such a public announcement.

For example, an attacker can pretend to be Bob and send a public key to Alice. When she goes to send a message to the real Bob using the imposter's key, the attacker can intercept and decrypt the message. If Bob discovers that an attacker is forging his identity, he can send Alice his real public key, but a better solution to the problem is one that prevents the forgery in the first place.

Bob can prevent the forgery by obtaining a **public key certificate** from a certification authority (CA). Bob can contact the CA and provide them with his authentication information, such as his driver's license or his user id, as well as his public key.

The CA then constructs Bob's certificate using his identification, his public key, and some other information, such as the period of validity for the certificate. Finally, the CA hashes and signs this information with their private key and then appends this hash to the certificate.

Now Bob can send his public key certificate to Alice. When Alice receives the certificate, she can extract the key as well as the information about Bob and the certificate itself. Alice can then hash the certificate and use the CA's public key to decrypt the signed, appended hash. If the two hashes match, she can be sure that no one has tampered with the certificate.



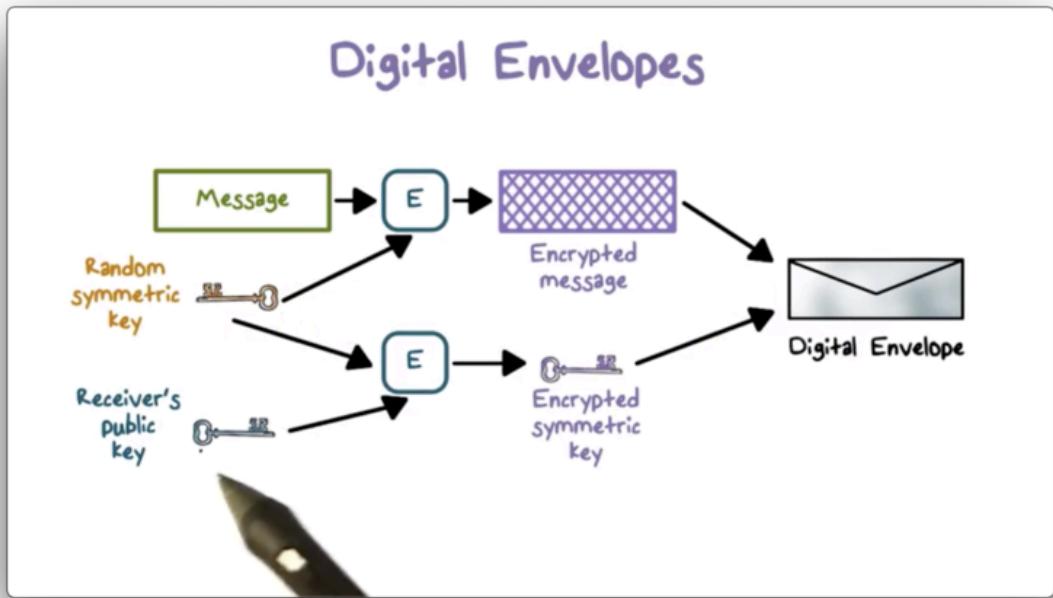
The underlying assumption is that the CA is a trusted party by everyone involved. In practice, the CA is a well-known company, such as VeriSign, Microsoft, Google, or Apple. Their public keys are already in your software - such as your browser - so they can automatically validate certificates that they have signed.

## 12.29 Digital Envelopes

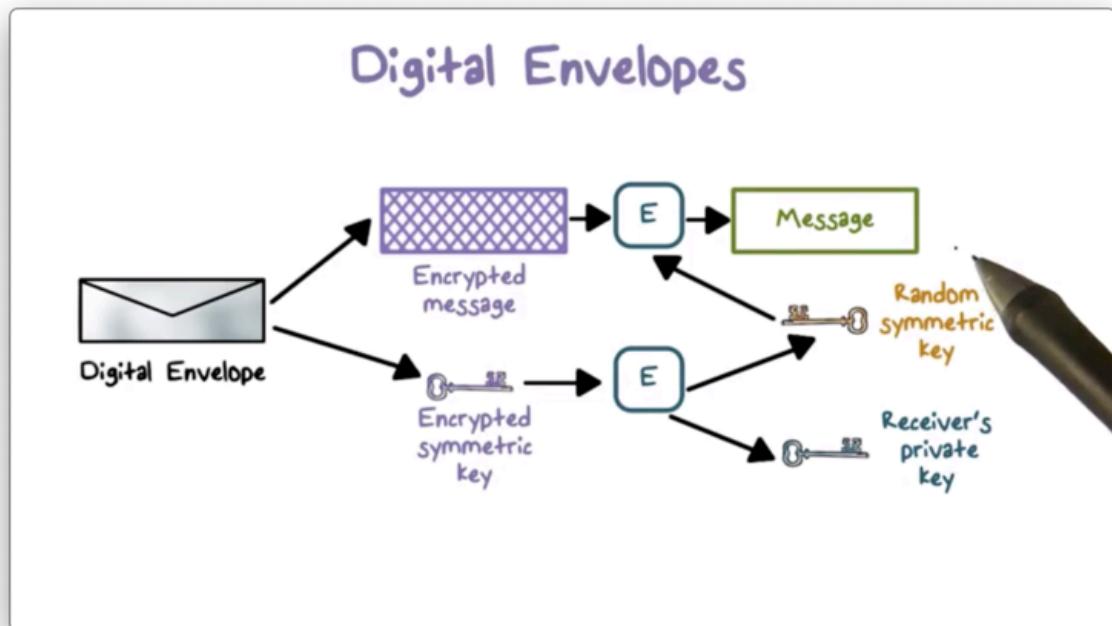
Public-key encryption is typically used to establish a symmetric key, which is then used to encrypt subsequent communications amongst parties.

Suppose Alice and Bob wish to communicate securely.

Alice first creates a random symmetric key that she wants to share with Bob and encrypts her message to Bob using this key. She then encrypts the symmetric key with Bob's public key and transmits a **digital envelope** containing both the message and the encrypted key to Bob.



When Bob receives the envelope, he can use his private key to decrypt the encrypted symmetric key. With the decrypted key in hand, he can decrypt the actual message from Alice.



### 12.30 Encryption Quiz



#### Encryption Quiz

Mark each of the statements either T for True or F for False:



- Symmetric encryption can only be used to provide confidentiality
- Public-key encryption can be used to create digital signatures
- Cryptanalytic attacks try every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained
- The secret key is input to the encryption algorithm

### 12.31 Encryption Quiz Solution



#### Encryption Quiz

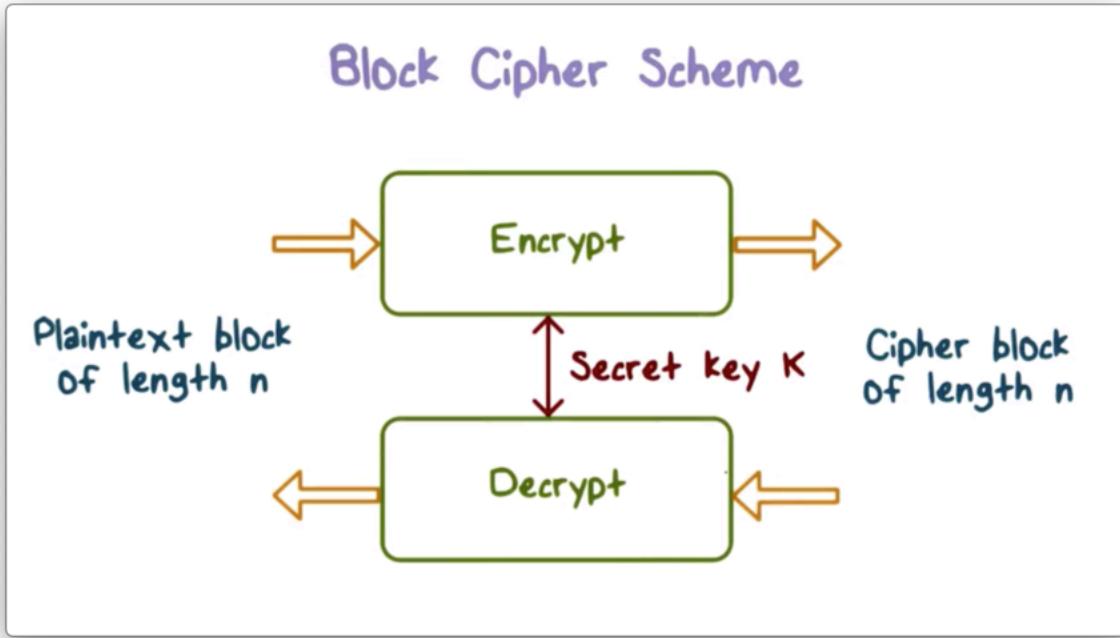
Mark each of the statements either T for True or F for False:

- F Symmetric encryption can only be used to provide confidentiality
- T Public-key encryption can be used to create digital signatures
- F Cryptanalytic attacks try every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained
- T The secret key is input to the encryption algorithm

## 13 Symmetric Encryption

### 13.1 Block Cipher Scheme

Most symmetric encryption schemes are block ciphers. A **block cipher** encrypts a plaintext block of length  $n$  into a ciphertext block of length  $n$  using a secret key  $k$  and decrypts the ciphertext using the same  $k$ .



### 13.2 Block Cipher Primitives

The goal of encryption is to transform plaintext into an unintelligible form. Since we assume that an attacker can obtain the ciphertext, we don't want the ciphertext to convey any information about the key or the plaintext.

**Confusion** obscures the relationship between the key and the ciphertext and is typically achieved with *substitution*. Through confusion, the attacker cannot determine the key, even if they obtain the ciphertext.

Confusion alone is not sufficient. Even when a letter can be mapped to any other letter, an attacker can perform a statistical analysis of letter frequencies to break the scheme. For example, the most common letter in English is 'E'. If the most common letter in the ciphertext is 'Q', we can be confident that 'E' maps to 'Q'.

The second principle that we need is diffusion. **Diffusion** spreads the influence of one plaintext bit over many ciphertext bits to hide the statistical properties of the plaintext. We can achieve diffusion with *permutation*.

For example, instead of mapping an English letter to another English letter, we can map a letter to parts of many 8-bit letters. This approach renders the frequency distribution of English letters less useless.

We need this combination - confusion and diffusion - to affect every bit in the ciphertext, so a block cipher typically runs for multiple rounds. The initial round affects some parts of the ciphertext, and subsequent rounds further propagate these effects into other parts of the ciphertext. Eventually, all bits of ciphertext are affected.

### 13.3 Block Cipher Quiz



#### Block Cipher Quiz

Select all correct answers to complete that statement.

A block cipher should...

- Use substitution to achieve confusion
- Use permutation to achieve diffusion
- Use a few rounds, each with a combination of substitution and permutation
- Keep the algorithm secret

### 13.4 Block Cipher Quiz Solution



#### Block Cipher Quiz

Select all correct answers to complete that statement.

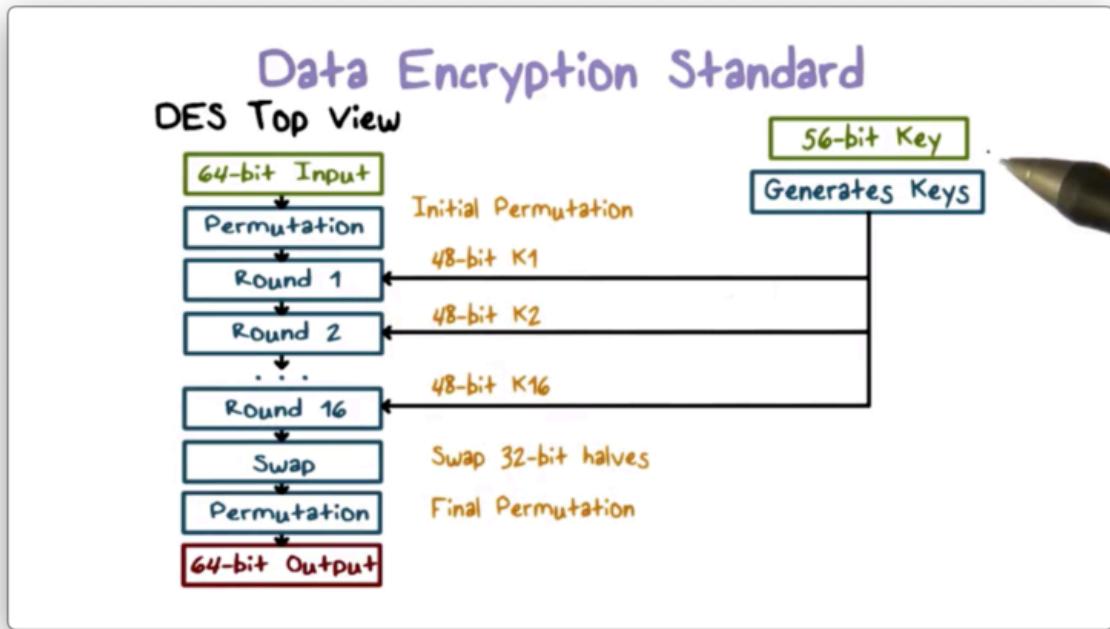
A block cipher should...

- Use substitution to achieve confusion
- Use permutation to achieve diffusion
- Use a few rounds, each with a combination of substitution and permutation
- Keep the algorithm secret



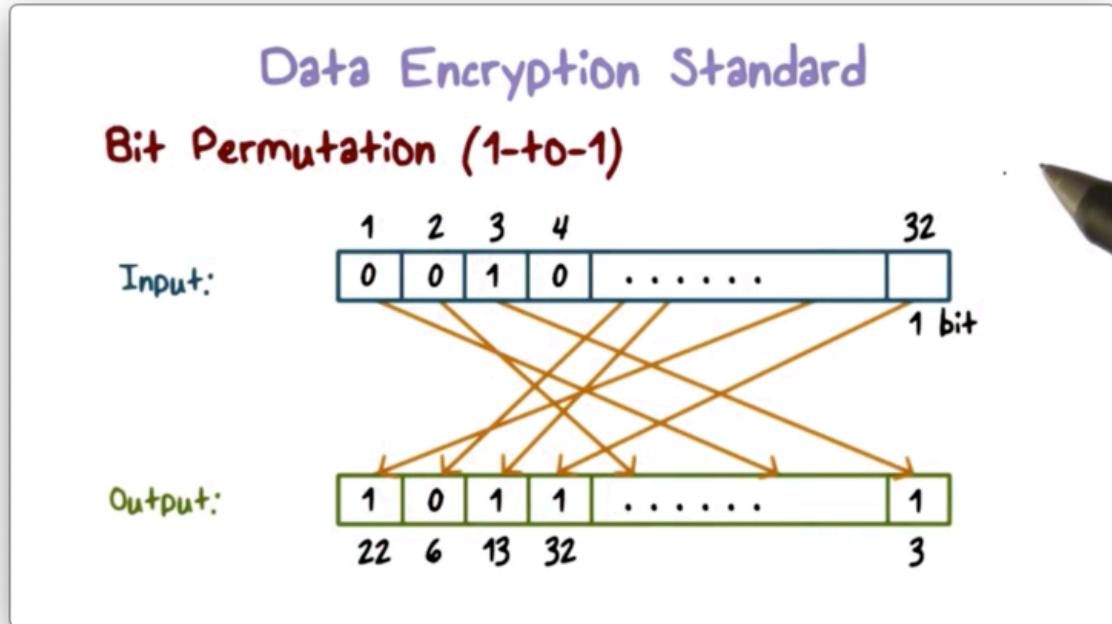
### 13.5 Data Encryption Standard

A widely used symmetric encryption scheme is based on the **Data Encryption Standard** (DES), which was established in 1977 and standardized in 1979.



In DES, the key is 64 bits (8 bytes) long. For each byte, there is one parity bit, so the actual value of the key is only 56 bits. DES receives a 64-bit plaintext block as input and produces a 64-bit ciphertext block.

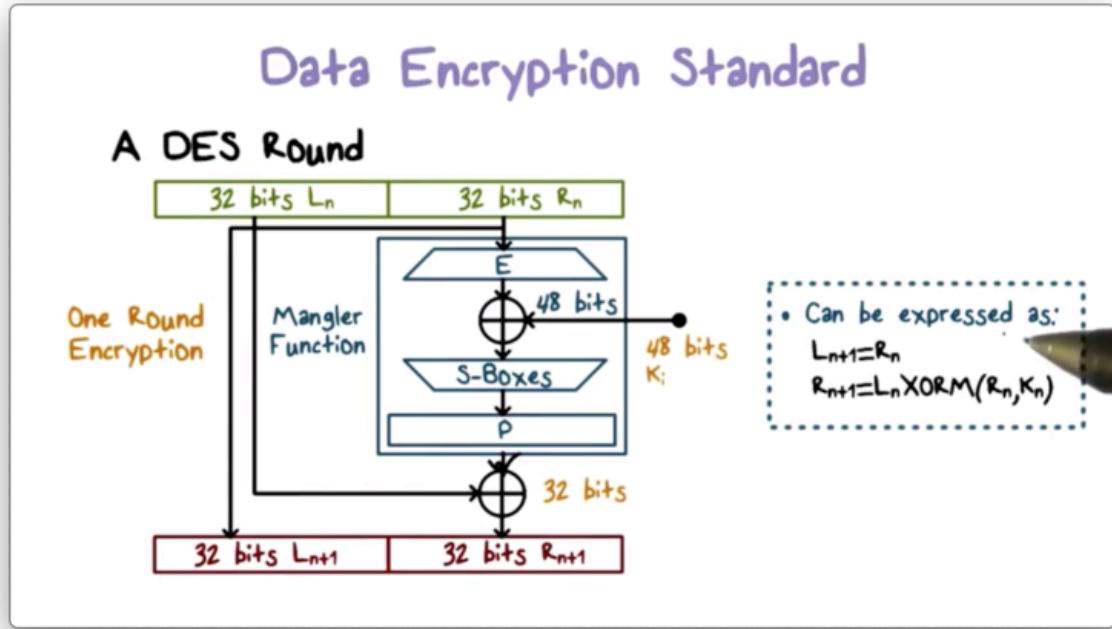
DES contains an initial and final permutation step that remaps the positions of the bits to achieve diffusion.



In between the permutation steps, DES performs 16 rounds of operations using 16 48-bit subkeys generated from the original 56-bit key. Each round receives as input the ciphertext produced by the previous round and outputs the ciphertext used as input by the next round.

A round proceeds as follows. First, the 64-bit input is divided into two 32-bit halves. The output left half is simply the input right half. The output right half is the XOR of the input left half, and the output of a mangler function applied to the input right half.

The mangler function receives a 32-bit input, expands it to 48 bits, XORs it with the 48-bit per-round key, and then passes it to an S-box to substitute the 48-bit value back into a 32-bit value.



The decryption process performs the same operations as the encryption process but uses the keys in reverse order. That is, the first decryption step uses the sixteenth key, and the final step uses the first key.

### 13.6 Decryption

DES is a [Fiestel cipher](#), which means that encryption and decryption differ only in key schedule. In decryption, we apply the same operations as we do in encryption, but with a reverse key sequence. Formally, round  $l$  of decryption uses subkey  $16 - l + 1$ .

Since each encryption round swaps the right and left halves of the input, the input to each decryption round is the swap of the input. However, since each decryption round performs the same operations as encryption, the decryption input is swapped to recover the original encryption input.

After 16 rounds of decryption, the algorithm has recovered the right and left halves of the original plaintext, and the final swap arranges the halves in the correct order.

### 13.7 XOR Quiz



#### XOR Quiz

Use the XOR function and the given key to encrypt the word "Hi".

key = FA F2

Hi =

FA F2 =

Hi encrypted =

### 13.8 XOR Quiz Solution



**XOR Quiz**

Use the XOR function and the given key to encrypt the word "Hi".

key = FA F2

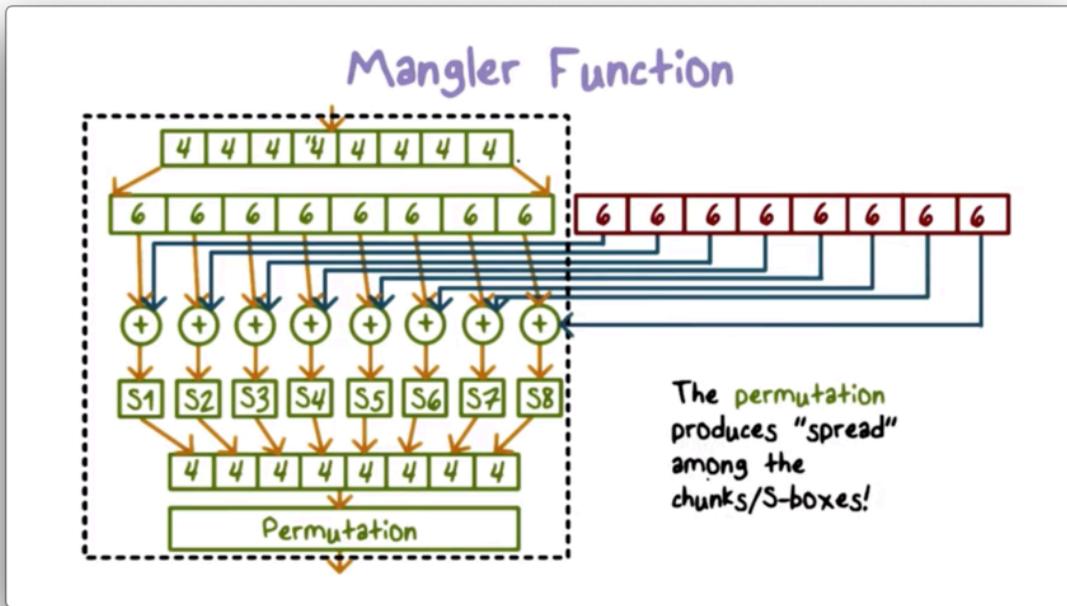
H:	=	0	1	0	0	1	0	0	1	0	0	1				
FA F2	=	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0
Hi encrypted	=	1	0	1	1	0	0	0	1	1	0	0	1	1	0	1

"H" has an ASCII code of 72, which maps to `0b01001000`, and "i" has an ASCII code of 105, which maps to `0b01101001`. "F" maps to 15 (`0b1111`) and "A" maps to 11 (`0b1001`), so "FA" maps to `0b11111001` and "F2" maps to `0b11111001`.

We XOR two numbers bit-by-bit, and we return 0 when the bits match and 1 otherwise. Therefore `0b0100100001101001` XOR `0b1111100111110010` is `0b1011000110011011`.

### 13.9 Mangler Function

The **mangler function** performs the bulk of the processing in a DES round. It expands the right half of the input from 32 bits to 48 bits and XORs it with the per-round key. The result is substituted back into a 32-bit value, which the function then permutes.

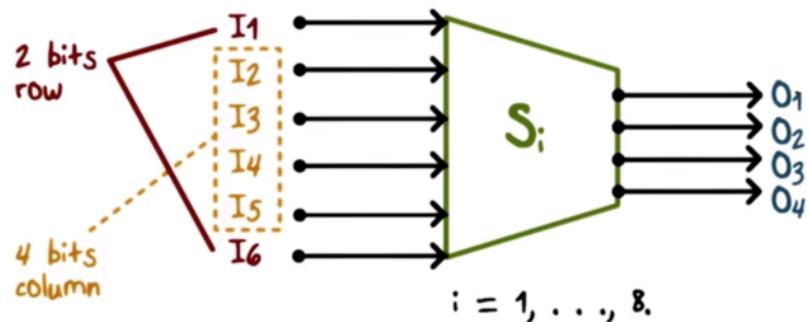


### 13.10 S Box

An **S-box** substitutes a 6-bit value with a 4-bit value using a predefined lookup table. DES uses 8 such S-boxes to substitute a 48-bit (68) *value with a 32-bit (48) value*. We can perform the substitution of a 6-bit value by using the outer two bits to look up the row of the S-box and the inner four bits to look up the column.

## S-Box (Substitute and Shrink)

- 48 bits  $\Rightarrow$  32 bits. ( $8 \times 6 \Rightarrow 8 \times 4$ )
- 2 bits used to select amongst 4 substitutions for the rest of the 4-bit quantity



### 13.11 S Box Quiz



#### S-Box Quiz

For the given input, determine the output.

S <sub>5</sub>		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Input: 011011

Output:

### 13.12 S Box Quiz Solution



**S-Box Quiz**

For the given input, determine the output.

		Middle 4 bits of input															
S <sub>5</sub>		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Input: 011011      Output: 1001



### 13.13 Security of DES

The key size in DES is 56 bits, so there are only  $2^{56}$  possible keys. This keyspace is too small, and an attacker can use brute-force to find the correct key relatively easily using today's computers.

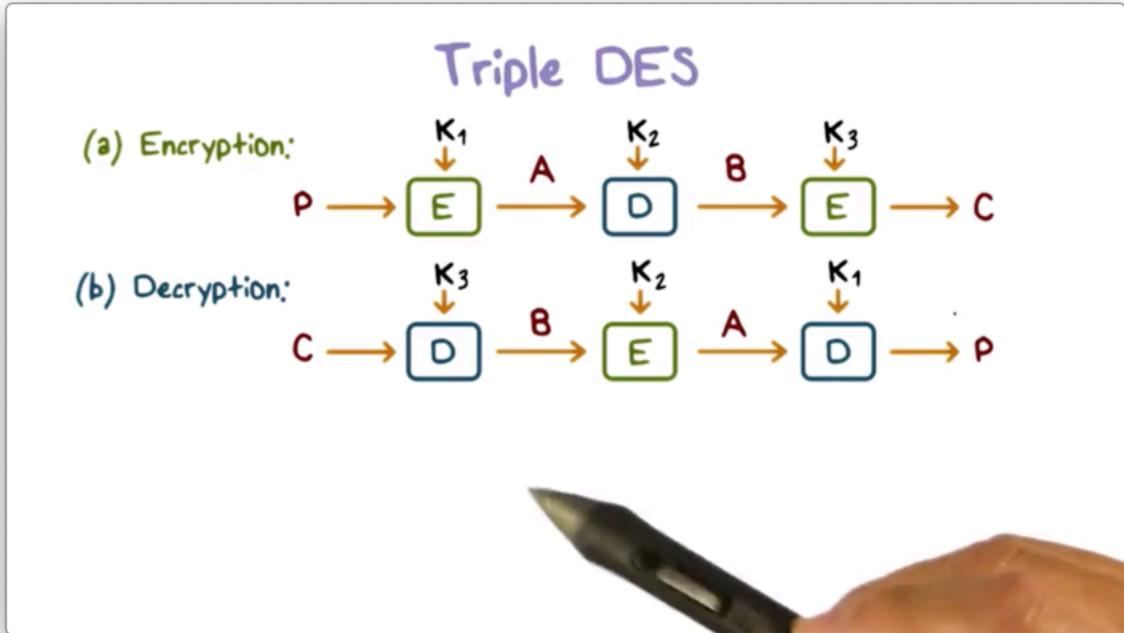
Another issue with DES is that the design criteria for the S-boxes have been kept secret. On the one hand, the S-boxes are resistant to attacks that were published years after DES was published, which indicates the security of the design. On the other hand, because the design criteria have been kept secret, one might wonder if the designer of DES knew about these attacks years before they were published.

One could argue that the designer should have published the design criteria so that researchers could review them and improve upon them.

### 13.14 Triple DES

To overcome the small keyspace that renders DES insecure, we can run DES multiple times, using a different key each time. The standard is to run it three times, and the resulting scheme is called **Triple DES**.

To encrypt plaintext using triple DES, we first run the encryption process with key  $k_1$ , followed by the decryption process with a second key  $k_2$ , followed by the encryption process with a third key  $k_3$ . To decrypt a ciphertext, we run the process in reverse, decrypting with  $k_3$ , encrypting with  $k_2$ , and decrypting again with  $k_1$ .



This order of operations is advantageous because it supports multiple key lengths. For example, if  $k_1$  and  $k_3$  are equal, the result is 112-bit DES. If all three keys are different, the effective key length is 168 bits.

Additionally, if we set  $k_2$  equal to  $k_1$ , then triple DES has essentially become single DES using  $k_3$ . Using this configuration, we can allow DES and triple DES to communicate with one another.

### 13.15 DES Quiz



#### DES Quiz

Check all the statements that are true:

- To decrypt using DES, same algorithm is used, but with per-round keys used in the reversed order
- With Triple DES the effective key length can be 56, 112, and 168
- Each round of DES contains both substitution and permutation operations
- The logics behind the S-boxes are well-known and verified



### 13.16 DES Quiz Solution



### DES Quiz

Check all the statements that are true:

- To decrypt using DES, same algorithm is used, but with per-round keys used in the reversed order
- With Triple DES the effective key length can be 56, 112, and 168
- Each round of DES contains both substitution and permutation operations
- The logics behind the S-boxes are well-known and verified

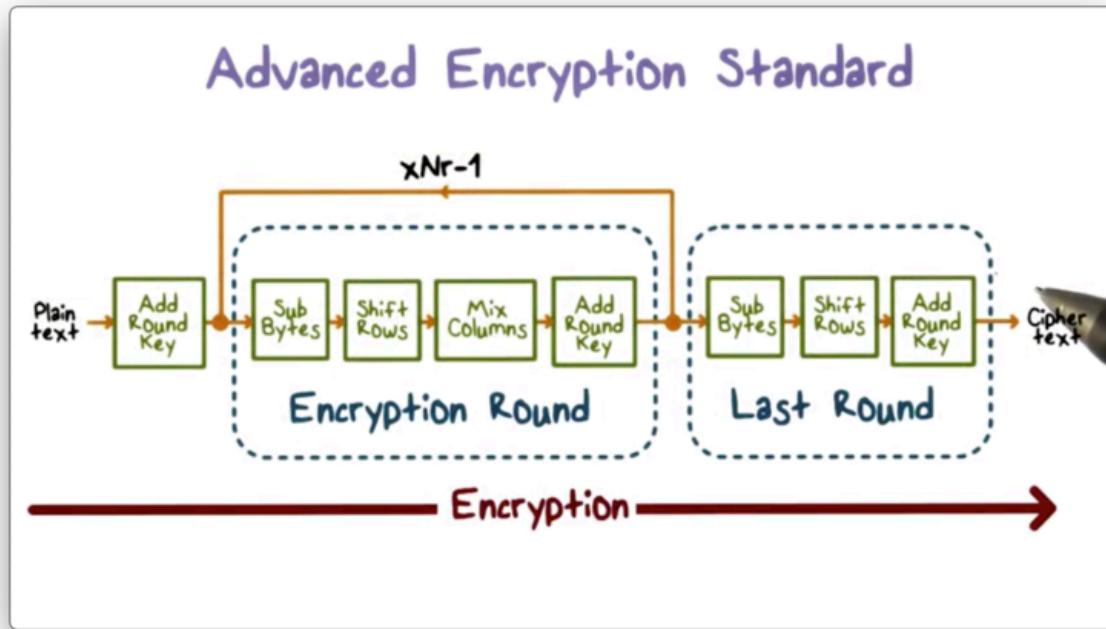


### 13.17 Advanced Encryption Standard

Key length is a significant shortcoming of DES. At 56 bits long, the keyspace for DES is too small, and an attacker can use brute force to find a key with the power of modern computers. While we can use triple DES to increase the key length, running DES three times is not an efficient approach.

The replacement for DES is the **Advanced Encryption Standard** (AES). Like DES, AES is a block cipher. Whereas in DES, the input plaintext block is 64-bit, it is 128-bit in AES. In DES, the key length is only 56-bit, but in AES, the key can be 128, 192, or 256 bits long. Researchers consider these key lengths long enough to defeat brute force attempts.

The following diagram presents a high-level overview of AES.



Each block of plaintext that AES operates on is represented as a square matrix called the *state array*. This array is first XORed with a per-round key before going through multiple rounds of encryption.

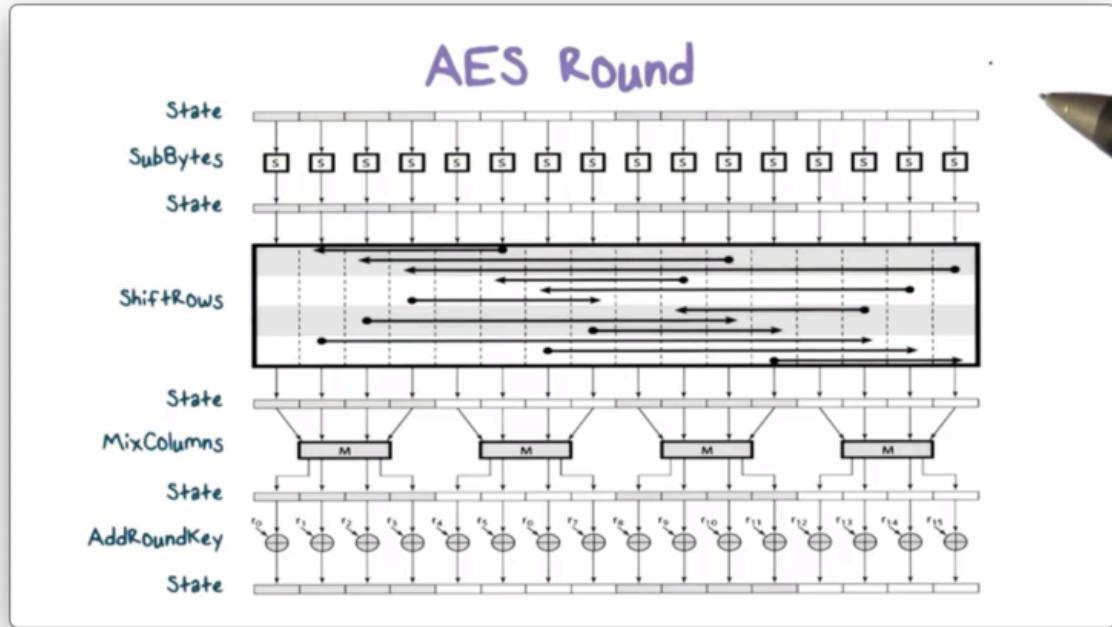
In each round, the state array passes through several operations - `SubBytes`, `ShiftRows`, and `MixColumns` - which represent substitution and permutation. The transformed value of the state array is XORed with the per-round key and then passed as input to the next round. The final round, which excludes the `MixColumns` step, produces the ciphertext.

In AES, the decryption process runs the encryption process in the reverse direction, which means that each of the applied operations must be reversible.

Adding the per-round key involves the XOR operation, which by itself is reversible. The decryption process uses an inverse function to reverse the other operations: `SubBytes`, `ShiftRows`, and `MixColumns`. Since each operation is reversible, the entire process is reversible, and we can recover the plaintext from the ciphertext.

### 13.18 AES Round

Let's take a closer look at each round of AES and the transformations the algorithm performs on the state matrix `S`. Each operation updates the value of `S` directly.



First, [SubBytes](#) uses S-boxes to perform a byte-by-byte substitution. Next, [ShiftRows](#) permutes [S](#) by shifting the bytes in each row of [S](#) a specified amount. Then, [MixColumns](#) substitutes each byte in a column as a function of all the bytes in the column. Finally, [AddRoundKey](#) XORs [S](#) with the per-round key, passing the final value of [S](#) to the next encryption round.

### 13.19 AES Encryption Quiz



#### AES Encryption Quiz

Check all the statements that are true:

- To decrypt using AES, just run the same algorithm in the same order of operations
- Each operation or stage in AES is reversible
- AES can support key length of 128, 192, 256
- AES is much more efficient than Triple DES



### 13.20 AES Encryption Quiz Solution



### AES Encryption Quiz

Check all the statements that are true:

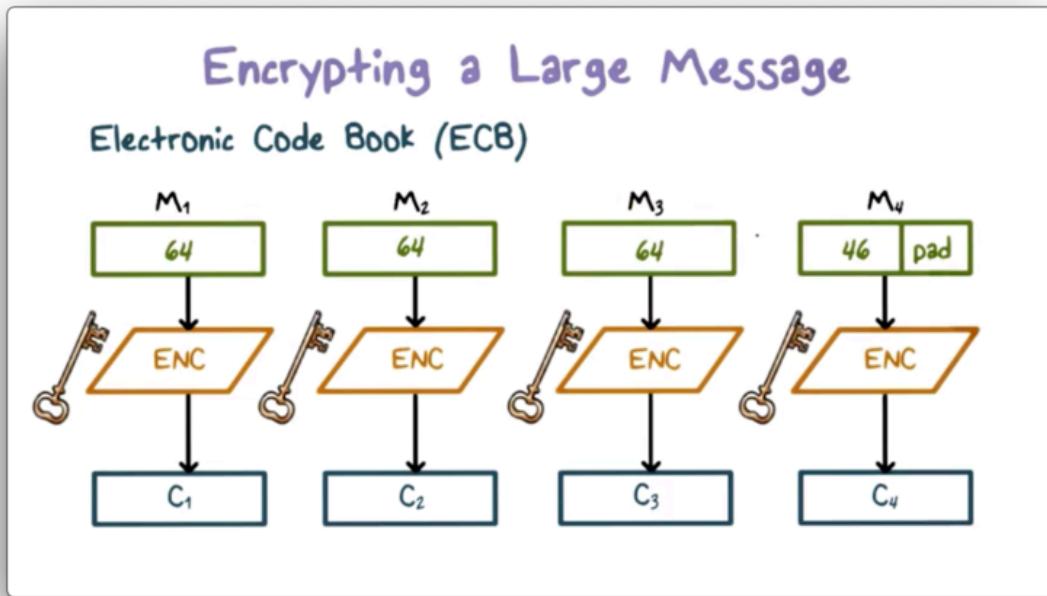
- To decrypt using AES, just run the same algorithm in the same order of operations
- Each operation or stage in AES is reversible
- AES can support key length of 128, 192, 256
- AES is much more efficient than Triple DES



### 13.21 Encrypting a Large Message

A block cipher takes in a fixed-length data block as input: 64 bits in DES and 128 bits in AES. If we want to encrypt a much bigger message, the solution seems obvious: break the message into fixed-size blocks, apply the cipher to each block, and combine the resulting ciphertexts.

The simplest method for encrypting large messages is to use an **electronic codebook** (ECB).



First, the original large message is broken down into fixed-size blocks. The final block is potentially padded if it is smaller than the block size. Each plaintext block is then encrypted using the same key, and the collection of ciphertext blocks is the ciphertext of the original message.

For a given key, there is a unique ciphertext block for every plaintext block. We can construct a large codebook - hence the name - in which there is an entry mapping every possible plaintext block to its ciphertext. In practice, the codebook only contains entries for the plaintext blocks used by the application.

### **13.22 ECB Problem #1**

There are significant shortcomings with the ECB approach, which make it a weak candidate for ensuring confidentiality.

For a given key, two identical plaintext blocks produce identical ciphertext blocks. As long as the same key is in use, this parity exists within a message and across distinct messages.

Cryptanalysts can exploit this consistency. For example, if an analyst knows that a message always starts with specific predefined fields, then they might immediately have several known plaintext/ciphertext pairs to aid their analysis.

Cryptanalysts can also leverage repetitive elements in a message. Whenever they see two identical

ciphertext blocks, they know that the corresponding plaintext blocks are the same.

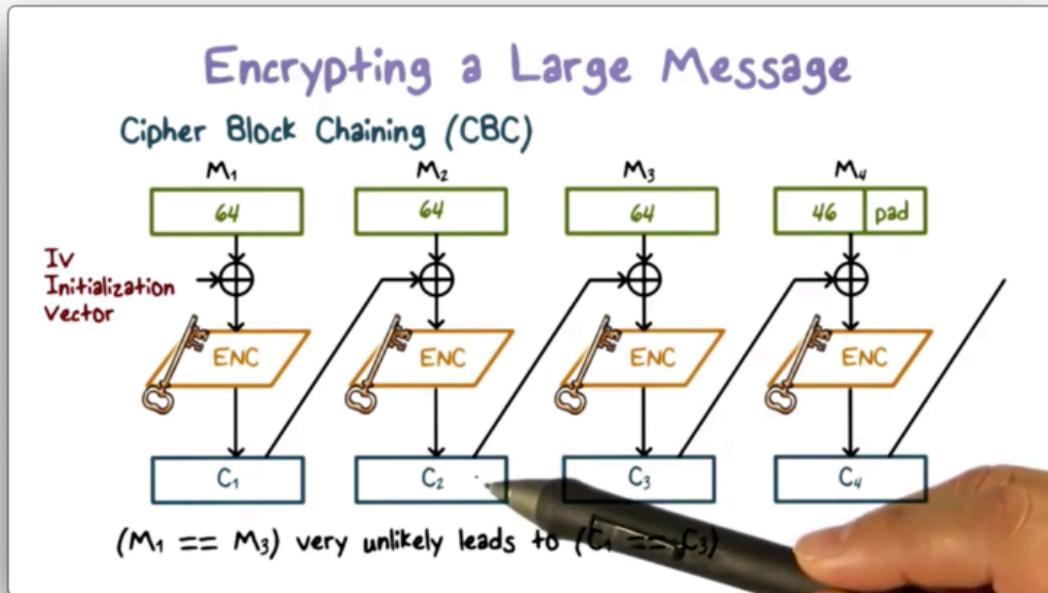
If they obtain a plaintext block for a ciphertext block, they can be sure that if they see the same ciphertext block again - assuming the key hasn't changed - that it decrypts to the plaintext block in their possession.

### 13.23 ECB Problem #2

Another problem with ECB is that the plaintext blocks are independently encrypted. An attacker can potentially rearrange inflight ciphertext blocks or substitute in a previously captured block for one currently being transmitted. Either way, an attacker can compromise message integrity as a result of the independent encryption scheme used by ECB.

### 13.24 Cipher Block Chaining

The most widely-used approach for encrypting a large message is **cipher block chaining** (CBC).

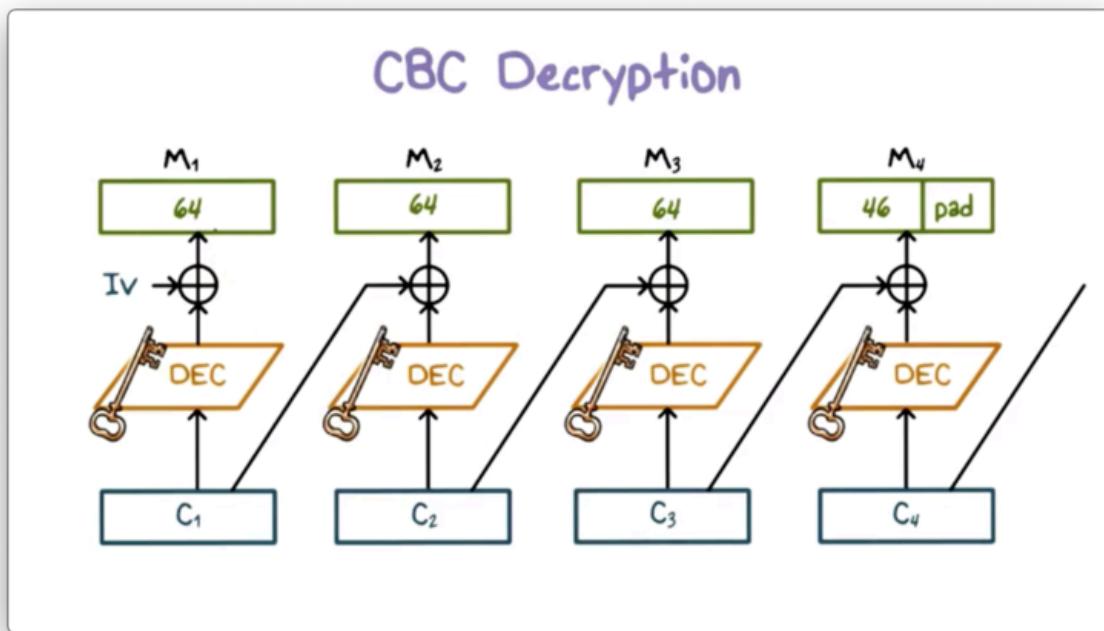


In CBC, the input to the encryption algorithm is the result of XORing the previous ciphertext block with the current plaintext block. To encrypt the first plaintext block, we XOR the block with an **initialization vector** (IV), which we then encrypt to produce the first ciphertext block.

This process is structured such that each ciphertext block incorporates information from all the previous ciphertext blocks. As a result, determining the plaintext block just by looking at the corresponding ciphertext block becomes very difficult.

More importantly, if two plaintext blocks are the same, the ciphertext blocks are not likely to be the same. Additionally, if an attacker tries to rearrange the ciphertext blocks or substitute one block for another, the ciphertext will not decrypt correctly into the plaintext.

Decryption works similarly. CBC decrypts a ciphertext block and then XORs the result with the previous ciphertext block to produce the current plaintext block.



The first decrypted ciphertext block must be XORed with the IV to produce the first plaintext block, which means that the IV must be known to both parties.

### 13.25 Protecting Message Integrity

While we usually discuss encryption as a method for providing confidentiality, we can also use encryption to ensure message integrity. In other words, encryption can surface unauthorized modifications to messages.

One approach for providing message integrity is to send the last block of CBC - the CBC *residue* - along with the plaintext. If an attacker intercepts the message and modifies the plaintext, they cannot recompute the CBC residue since they do not have the encryption key. Thus, they are forced to send the

modified plaintext with the original residue.

When the authorized recipient receives the message, they will run the message through CBC to produce the residue. Since the plaintext has changed inflight, but the residue has not, the recipient's computed residue will not match the received residue. Therefore, the receiver will know that an unauthorized party has modified the plaintext.

### 13.26 Protecting Message Confidentiality and Integrity

To protect both message confidentiality and integrity, we should use two separate keys and two encryption rounds: one key produces the ciphertext, and the other key produces the CBC residue. Alternatively, we can first compute a hash of the message, append the hash to the message, and then encrypt the entire entity.

### 13.27 CBC Quiz



### CBC Quiz

Put a check next to the statements that are true:

CBC is more secure than ECB

We can have both confidentiality and integrity protection with CBC by using just one key



### 13.28 CBC Quiz Solution



### CBC Quiz

Put a check next to the statements that are true:

CBC is more secure than ECB

We can have both confidentiality and integrity protection with CBC by using just one key



## 14 Public-Key Cryptography

### 14.1 Modular Arithmetic

Both RSA and Diffie-Hellman - the most widely-used public-key algorithms - are based on number theory and use modular arithmetic - modular addition, multiplication, and exponentiation. Before we dive into the details of the algorithms themselves, let's review the basics of modular arithmetic.

Given a modulus  $M$ ,  $x + y \pmod{M}$  is equal to the remainder of  $(x + y) \div M$ . For example,  $2 + 8 \pmod{10} = 0$ , because  $10 \div 10$  divides evenly whereas  $3 + 8 \pmod{10} = 1$  because  $11 \div 10$  yields a remainder of 1.

In modular addition, a number  $k$  has an inverse  $k'$  such that  $k + k' \pmod{M} = 0$ . For example, for  $M = 10$  and  $k = 2$ ,  $k' = 8$  because  $2 + 8 \pmod{10} = 0$ . Every number has an inverse under modular addition.

The presence of an additive inverse means that modular addition is reversible. That is, given  $c = a + b \pmod{M}$ ,  $a = c + b' \pmod{M}$  and  $b = c + a' \pmod{M}$ . This reversibility is very convenient

for encryption because we want the decryption process ideally to be the reverse of the encryption process.

Suppose we have plaintext  $p = 3$ , key  $k = 2$  and encryption algorithm  $p + k \pmod{10}$ . Thus, ciphertext  $c = 3 + 2 \pmod{10} = 5 \pmod{10} = 5$ . We can decrypt  $c$  using the inverse of  $k$ :  $k'$ . Since  $k' = 8$ ,  $c + k' \pmod{10} = 5 + 8 \pmod{10} = 13 \pmod{10} = 3 = p$ .

## 14.2 Additive Inverse Quiz



### Additive Inverse Quiz

What is the additive inverse of 8 MOD 20?

Enter an answer in the textbox:

### 14.3 Additive Inverse Quiz Solution



### Additive Inverse Quiz

What is the additive inverse of 8 MOD 20?

Enter an answer in the textbox:



In modular addition, a number  $k$  has an inverse  $k'$  such that  $k + k' \pmod{M} = 0$ . In this case,  $M = 20$  and  $k = 8$ . Therefore,  $k' = 12$  because  $8 + 12 \pmod{20} = 0$ .

### 14.4 Modular Multiplication

Given a modulus  $M$ ,  $x * y \pmod{M}$  is equal to the remainder of  $(x * y) \div M$ . For example,  $5 * 8 \pmod{10} = 0$ , because  $40 \div 10$  divides evenly whereas  $4 * 8 \pmod{10} = 2$  because  $32 \div 10$  yields a remainder of 2.

In modular multiplication, a number  $k$ , has an inverse  $k'$  such that  $k * k' \pmod{M} = 1$ . For example, for  $M = 10$  and  $k = 3$ ,  $k' = 7$  because  $3 * 7 \pmod{10} = 21 \pmod{10} = 1$ .

Not all numbers have multiplicative inverses for a given  $M$ . For  $M = 10$ , for example, 2, 5, 6, and 8 do not have multiplicative inverses.

### 14.5 Modular Multiplication Quiz



### Modular Multiplication Quiz

What is multiplicative inverse of 3 MOD 17?

Enter an answer in the textbox:

## 14.6 Modular Multiplication Quiz Solution



### Modular Multiplication Quiz

What is multiplicative inverse of 3 MOD 17?

Enter an answer in the textbox:

A hand holding a pen is shown writing the number 6 into the text box.

In modular multiplication, a number  $k$  has an inverse  $k'$  such that  $k * k' \pmod{M} = 1$ . In this case,  $M = 17$  and  $k = 3$ . Therefore,  $k' = 6$  because  $3 * 6 \pmod{17} = 18 \pmod{17} = 1$ .

## 14.7 Totient Function

Given a modulus  $M$ , only the numbers that are relatively prime to  $M$  have multiplicative inverses in  $(\pmod{M})$ .

Two numbers  $x$  and  $y$  are *relatively prime* to one another if they share no common factor other than 1. For example, 3 and 10 are relatively prime, whereas 2 and 10 - which share 2 as a common factor - are not.

Given  $n$ , we can calculate how many integers are relatively prime to  $n$  using the **totient function**  $\phi(n)$ . If  $n$  is prime,  $\phi(n) = n - 1$ , because every number smaller than  $n$  is relatively prime to  $n$  because  $n$  itself is prime. If  $n = p * q$  and  $p$  and  $q$  are prime, then  $\phi(n) = (p - 1) * (q - 1)$ . If  $n = p * q$  and  $p$  and  $q$  are relatively prime,  $\phi(n) = \phi(p) * \phi(q)$ .

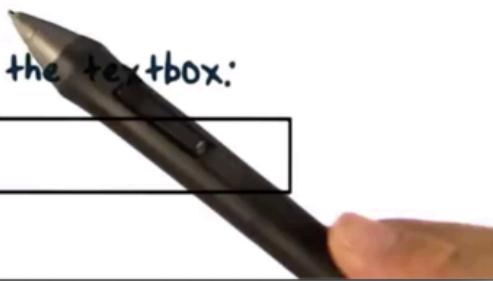
### 14.8 Totient Quiz



### Totient Quiz

If  $n = 21$ , what is  $\phi(n)$ ?

Enter an answer in the textbox:



### 14.9 Totient Quiz Solution



### Totient Quiz

If  $n = 21$ , what is  $\phi(n)$ ?

Enter an answer in the textbox:

A digital pen is shown pointing towards the text box.

If  $n = p * q$  and  $p$  and  $q$  are prime, then  $\phi(n) = (p - 1) * (q - 1)$ . For  $n = 21$ ,  $p = 3$  and  $q = 7$ ,  $\phi(n) = (3 - 1) * (7 - 1) = 2 * 6 = 12$ .

### 14.10 Modular Exponentiation

In modular exponentiation,  $x^y \pmod{n} = x^{y \pmod{\phi(n)}} \pmod{n}$ . If  $y \equiv 1 \pmod{\phi(n)}$ , then  $x^y \pmod{n} = x \pmod{n}$ .

### 14.11 Modular Exponentiation Quiz



#### Modular Exponentiation Quiz

Use the totient technique to find  $c$ .  
Write your answer in the textbox:

$$c = 7^{27} \bmod 30$$

$c =$

### 14.12 Modular Exponentiation Quiz Solution



## Modular Exponentiation Quiz

Use the totient technique to find  $c$ .  
Write your answer in the textbox:

$$c = 7^{27} \text{ mod } 30 = 7^{27 \text{ mod } \phi(30)} \text{ mod } 30$$

$c =$

We know that  $x^y \pmod{n} = x^{y \pmod{\phi(n)}} \pmod{n}$ . For  $x = 7, y = 27$  and  $n = 30$ ,  $7^{27} \pmod{30} = 7^{27 \pmod{\phi(30)}} \pmod{30}$ . We can calculate  $\phi(30)$  as follows:  $\phi(30) = \phi(3) * \phi(10) = \phi(3) * \phi(2) * \phi(5) = 2 * 1 * 4 = 8$ . Thus,  $7^{27} \pmod{30} = 7^{27 \pmod{8}} \pmod{30}$ . If we divide 27 by 8, we are left with a remainder of 3, so  $7^{27} \pmod{30} = 7^3 \pmod{30}$ .  $7^3 = 343$ , which yields a remainder of 13 when divided by 30.

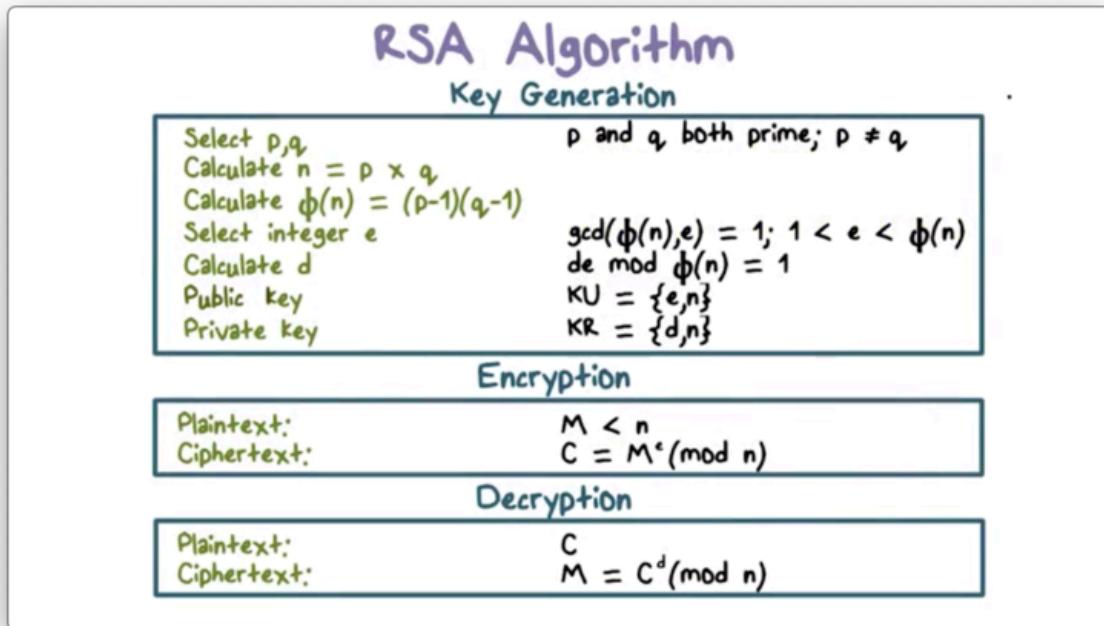
### 14.13 RSA (Rivest, Shamir, Adleman)

RSA - named after its three creators - is the most widely used public-key algorithm. RSA supports both public-key encryption and digital signature.

RSA bases the strength of its security on the hypothesis that **factoring** a massive number into two primes is computationally infeasible problem to solve on a reasonable timescale using modern computers.

RSA supports variable key lengths, and in practice, most people use a 1024-, 2048-, or 4096-bit key. The plaintext block size can also be variable, but the value of the block - represented as a binary integer - must be smaller than the value of the key. The length of the ciphertext block is the same as the key length.

Here is a summary of RSA, including the key generation, encryption, and decryption steps.



The first step is key generation. First, we select two primes  $p$  and  $q$  that are at least 512 bits in size. Next, we compute  $n = p * q$ , and  $\phi(n) = (p - 1) * (q - 1)$ . Then, we select an integer  $e$  that is smaller than  $\phi(n)$  and relatively prime to  $\phi(n)$ . Finally, we calculate  $d$ : the multiplicative inverse of  $e$ ,  $(\text{mod } \phi(n))$ .

The public key is  $e, n$ . The private key is  $d, n$ .

Suppose Bob wishes to send a message  $m$  to Alice that only she can read. Bob can encrypt  $m$  using Alice's public key,  $e, n$  by computing  $m^e \pmod{n}$ . On receipt of ciphertext  $C$ , Alice can use her private key,  $d, n$ , and compute  $C^d \pmod{n}$  to recover  $m$ .

RSA guarantees that only Alice can decrypt  $m$  because only she has the private key that pairs with the public key used to encrypt the message.

Digital signature creation and verification work in a similar fashion as encryption and decryption.

## How Does RSA Work?



- Given  $KU = \langle e, n \rangle$  and  $KR = \langle d, n \rangle$

- encryption:**  $c = m^e \pmod{n}$ ,  $m < n$
- decryption:**  $m = c^d \pmod{n}$
- signature:**  $s = m^d \pmod{n}$ ,  $m < n$
- verification:**  $m = s^e \pmod{n}$

To create a signature  $s$  for a message  $m$ , Alice uses her private key,  $d, n$  to compute  $m^d \pmod{n}$ . Bob can verify Alice's signature by using her public key,  $e, n$  to compute  $s^e \pmod{n}$ , which is equivalent to the original message  $m$ .

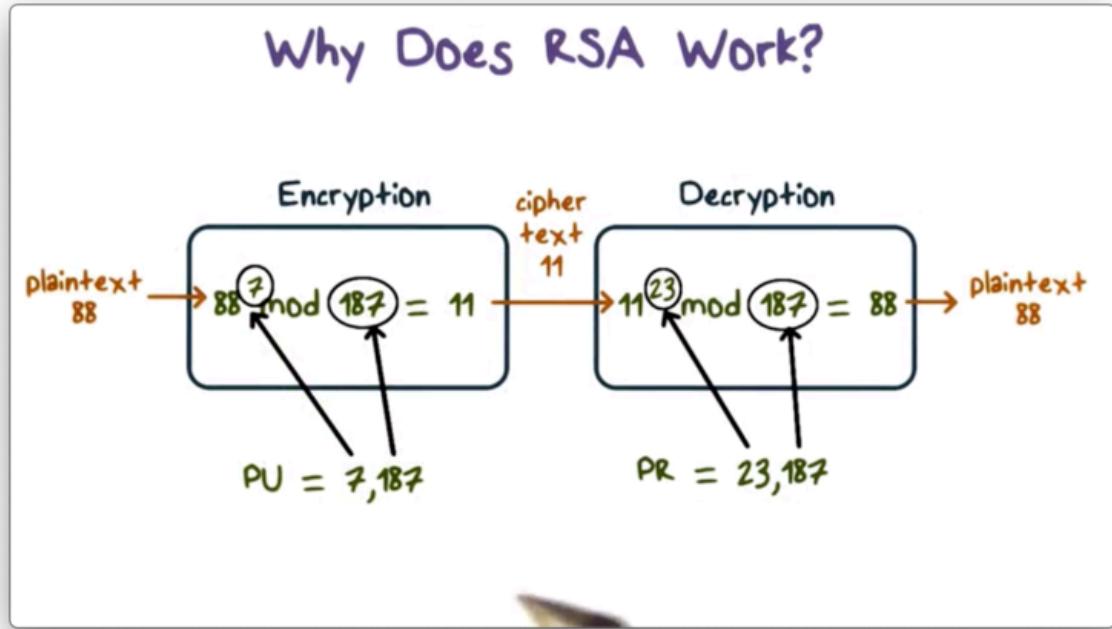
### 14.14 Why Does RSA Work

Remember from the rules of modular exponentiation that, for a base  $x$ , a power  $y$ , a modulus  $n$ ,  $x^y \pmod{n} = x^{y \pmod{\phi(n)}} \pmod{n}$ . If  $y = 1 \pmod{\phi(n)}$ , then  $x^y \pmod{n} = x \pmod{n}$ .

Also recall that, for a given public key,  $e, n$  and its private key  $d, n$ ,  $d * e = 1 \pmod{\phi(n)}$ . Thus,  $x^{e*d} \pmod{n} = x \pmod{n}$ .

To encrypt a message  $m$ , we compute  $c = m^e \pmod{n}$ . To decrypt  $c$ , we compute  $m = c^d \pmod{n}$ . We can substitute the first expression in for  $c$  in the second to get  $m = (m^e \pmod{n})^d \pmod{n}$ . The right-hand side of the equation simplifies to  $m^{ed} \pmod{n}$ , which is equivalent to  $m \pmod{n}$ , which equals  $m$  since  $m < n$ .

Here is an example of RSA in action.



The key generation step proceeds as follows. First, we select two prime numbers,  $p = 17$  and  $q = 11$ . From these values, we can compute  $n$  and  $\phi(n)$ , as  $p * q = 187$  and  $(p-1)*(q-1) = 160$ , respectively. We select 7 as our public key  $e$ , as 7 and 160 are relatively prime. The private key is the multiplicative inverse of  $e$ ,  $(\text{mod } \phi(n))$ . The multiplicative inverse of 7  $(\text{mod } 160)$  is 23, which is our private key  $d$ .

To encrypt a plaintext  $m = 88$ , we compute ciphertext  $C = m^e \pmod{n} = 88^7 \pmod{187} = 11$ . To decrypt  $C$ , we perform  $C^d \pmod{n} = 11^{23} \pmod{187} = 88$ , which returns  $m$ .

### 14.15 RSA Quiz

 **RSA Quiz**

Fill in the text boxes:

Given  $p = 3$  and  $q = 11$

Compute  $n$ :

$n = \boxed{\hspace{2cm}}$

Compute  $\phi(n)$ :

$\phi(n) = \boxed{\hspace{2cm}}$

Assume  $e = 7$

Compute the value of  $d$ :

What is the public key  $(e, n) = (\boxed{\hspace{2cm}}, \boxed{\hspace{2cm}})$

What is the private key  $(d, n) = (\boxed{\hspace{2cm}}, \boxed{\hspace{2cm}})$



### 14.16 RSA Quiz Solution

 **RSA Quiz**

Fill in the text boxes:

Given  $p = 3$  and  $q = 11$

Compute  $n$ :

$$n = \boxed{33}$$

Compute  $\phi(n)$ :

$$\phi(n) = \boxed{20}$$

Assume  $e = 7$

Compute the value of  $d$ :

$$\boxed{3}$$

What is the public key  $(e, n) = (\boxed{7}, \boxed{33})$

What is the private key  $(d, n) = (\boxed{3}, \boxed{33})$

$n = p * q = 11 * 3 = 33$  and  $\phi(n) = (p - 1) * (q - 1) = 2 * 10 = 20$ .  $e$  and  $d$  must be multiplicative inverses  $(\text{mod } \phi(n))$ , so for  $e = 7$ ,  $d = 3$ , since  $21 \pmod{20} = 1$ . Finally, public key  $e, n$  is equal to 7, 33, and private key,  $d, n$  is equal to 3, 33.

### 14.17 RSA Encryption Quiz

#### RSA Encryption Quiz



Given:

Public key is  $(e, n) \Rightarrow (7, 33)$

Private key is  $(d, n) \Rightarrow (3, 33)$

Message  $m = 2$



What is the encryption of  $m$ :

What formula is used to decrypt  $m$ ?

(Use  $**$  for denoting an exponent)

### 14.18 RSA Encryption Quiz Solution

**RSA Encryption Quiz**



Given:  
Public key is  $(e, n) \Rightarrow (7, 33)$   
Private key is  $(d, n) \Rightarrow (3, 33)$   
Message  $m = 2$

What is the encryption of  $m$ :  $2^{**7 \% 33 = 29}$

What formula is used to decrypt  $m$ ?  $29^{**3 \% 33 = 2}$

(Use  $**$  for denoting an exponent)

Encrypting message  $m$  involves computing  $m^e \pmod{n}$ , which is equivalent to  $2^7 \pmod{33} = 128 \pmod{33} = 29$ . Decrypting ciphertext  $C$  involves computing  $C^d \pmod{n}$ , which is equivalent to  $29^3 \pmod{33} = 24389 \pmod{33} = 3$ .

### 14.19 Why is RSA Secure?

RSA bases its security on the hypothesis that factoring a very large number - at least 512 bits, but often in the range of 1024-4096 bits - takes an inordinately long amount of time using modern computers.

If someone finds an efficient way to factor a large number into primes, then the security of RSA is effectively broken. Given public key,  $e, n$ , an efficient factorization of  $n$  into  $p$  and  $q$  allows an attacker to compute  $\phi(n)$  and the multiplicative inverse of  $e$ ,  $(\text{mod } \phi(n))$ . This inverse is  $d$ , the private key.

An attacker can read confidential messages and forge digital signatures against any public key if they can compute a private key in a reasonable amount of time.

## 14.20 Issues with Schoolbook RSA

One issue with RSA is that the algorithm is deterministic. For a given key, the same plaintext message always encrypts to the same ciphertext. Additionally, for specific plaintext values such as 0, 1, or -1, the ciphertext is always equivalent to the plaintext, regardless of the key used.

Another issue is that RSA is malleable. An attacker can induce predictable transformations in plaintext by modifying ciphertext in specific ways.

For example, suppose Bob sends Alice an encrypted message  $c = m^e \pmod{n}$  using Alice's public key,  $e, n$ . The attacker intercepts  $c$  and performs the transformation  $c' = s^e * c$ . When Alice receives this ciphertext, the decrypted result is  $m' = s * m$ .

In practice, the standard is to prepend  $m$  with [padding](#), a step which addresses the issues described above.

## 14.21 RSA in Practice Quiz



### RSA in Practice Quiz

Select the best answer.

When implementing RSA, it is best to use:

- Your own custom software, to ensure a secure system
- Use the standard libraries for RSA

### 14.22 RSA in Practice Quiz Solution



**RSA in Practice Quiz**

Select the best answer.

When implementing RSA, it is best to use:

- Your own custom software, to ensure a secure system
- Use the standard libraries for RSA



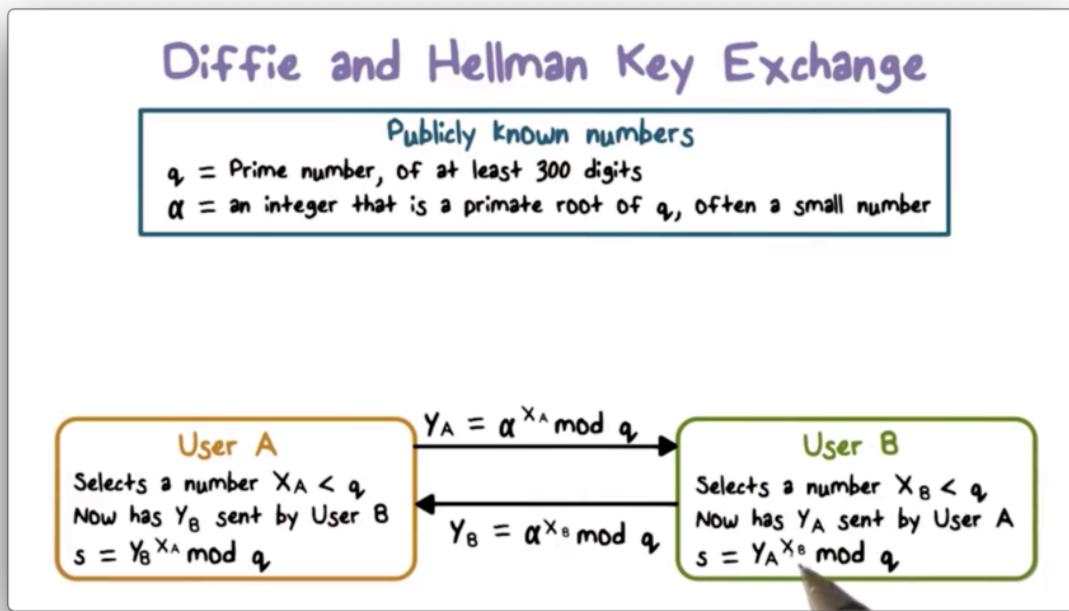
Always use standard libraries, as they have been reviewed and tested by experts in the field.

### 14.23 Diffie and Hellman Key Exchange

In the Diffie-Hellman key exchange algorithm, there are two publicly known numbers  $q$  and  $\alpha$ .  $q$  is a large prime number - at least 300 digits long - and  $\alpha$  is a small [primitive root](#) of  $q$ .

Suppose two users  $A$  and  $B$  wish to exchange a key using Diffie-Hellman.  $A$  selects a random integer  $X_A < q$  and computes  $Y_A = \alpha^{X_A} \pmod{q}$ . Likewise,  $B$  selects a random integer  $X_B < q$  and then computes  $Y_B = \alpha^{X_B} \pmod{q}$ . Each side keeps its  $X_*$  value private and sends their  $Y_*$  value to the other side.

Upon receiving  $Y_B$  from  $B$ ,  $A$  computes the key  $k = Y_B^{X_A} \pmod{q}$ . Similarly,  $B$  computes the key  $k = Y_A^{X_B} \pmod{q}$  upon receiving  $Y_A$  from  $A$ . The expressions used to compute  $k$  by each party are equivalent; that is,  $Y_B^{X_A} \pmod{q} = Y_A^{X_B} \pmod{q}$ . As a result of this exchange, both sides now share a secret encryption key.



The values of  $X_A$  and  $X_B$  are private while  $\alpha$ ,  $q$ ,  $Y_A$ , and  $Y_B$  are public. If an attacker  $C$  wants to compute the shared key, they must first compute either  $X_B$  or  $X_A$ . The attacker can compute  $X_B$ , for example, by computing  $dlog(\alpha, q)(Y_B)$ , where  $dlog$  is the [discrete logarithm](#). If  $C$  can retrieve  $X_B$ , they can compute the shared key using  $Y_A$  and  $q$ .

The security of Diffie-Hellman lies in the fact that it is infeasible to compute discrete logarithms for large primes such as  $q$  using modern computers.

### 14.24 Diffie-Hellman Example

Let's walk through the Diffie-Hellman key exchange using  $q = 353$  and  $\alpha = 3$ .

First, user  $A$  selects a random number,  $X_A < q = 97$ , which they keep secret, and they compute  $Y_A = 3^{97} \pmod{353} = 40$ . Similarly, user  $B$  selects a random number  $X_B < q = 233$ , which they also keep secret, and they compute  $Y_B = 3^{233} \pmod{353} = 248$ . Next,  $A$  sends 40 to  $B$  and  $B$  sends 248 to  $A$ .

Upon receiving  $Y_A$ ,  $B$  computes  $k = Y_A^{X_B} \pmod{353} = 40^{233} \pmod{353} = 160$ . Similarly,  $A$  computes  $k = Y_B^{X_A} \pmod{353} = 248^{97} \pmod{353} = 160$ . Through this exchange, both  $A$  and  $B$  have computed the same secret value, which they can now use to encrypt their communications.

## Diffie-Hellman Example

Have

- Prime number  $q = 353$
- Prime number  $\alpha = 3$

A and B each compute their public keys

- A computes  $Y_A = 3^{97} \bmod 353 = 40$
- B computes  $Y_B = 3^{233} \bmod 353 = 248$

Then exchange and compute secret key:

- For A:  $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$
- For B:  $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$

We assume that an attacker can access  $Y_A$ ,  $Y_B$ ,  $q$ , and  $\alpha$ . Because the numbers in this example are quite small, an attacker can feasibly compute the discrete log to retrieve either  $X_B$  or  $X_A$ . However, as we said before, this computation becomes infeasible for the large values of  $q$  used in practice.

### 14.25 Diffie-Hellman Quiz



#### Diffie-Hellman Quiz

Alice and Bob agree to use prime  $q = 23$  and primitive root  $\alpha = 5$

Alice chooses secret  $a = 6$

Bob chooses secret  $b = 15$

What number does Alice send Bob?

What number does Bob send Alice?



### 14.26 Diffie-Hellman Quiz Solution



#### Diffie-Hellman Quiz

Alice and Bob agree to use prime  $q = 23$  and primitive root  $\alpha = 5$

Alice chooses secret  $a = 6$   
Bob chooses secret  $b = 15$

What number does Alice send Bob?  $5^6 \text{ mod } 23 = 8$

What number does Bob send Alice?  $5^{15} \text{ mod } 23 = 19$

Alice sends  $\alpha^a \pmod{q}$  to Bob, which is equivalent to  $5^6 \pmod{23} = 8$ . Bob sends  $\alpha^b \pmod{q}$  to Alice, which is equivalent to  $5^{15} \pmod{23} = 19$ .

### 14.27 Diffie-Hellman Security

In Diffie-Hellman, neither party ever transmits the shared secret encryption key  $S$ , nor the locally generated secrets  $X_A$  and  $X_B$ . Since we assume that attackers can intercept any transmitted value, the lack of transmission of secret values adds to the security of the scheme.

We assume that an attacker can access  $Y_A$ ,  $Y_B$ ,  $q$ , and  $\alpha$ , since these values are transmitted. The value of local secret  $X_A$  is equal to the discrete logarithm  $dlog(\alpha, q)(Y_A)$ . The security assumption in Diffie-Hellman is that finding the discrete logarithm is infeasible given a very large, prime  $q$ .

Of course, if this conjecture is not valid, then an adversary knowing  $Y_A$ ,  $q$ , and  $\alpha$  can easily compute  $X_A$ . With  $X_A$  in hand, they can compute  $S$  and effectively eavesdrop on communication between  $A$  and  $B$ .

### 14.28 Diffie-Hellman Limitations

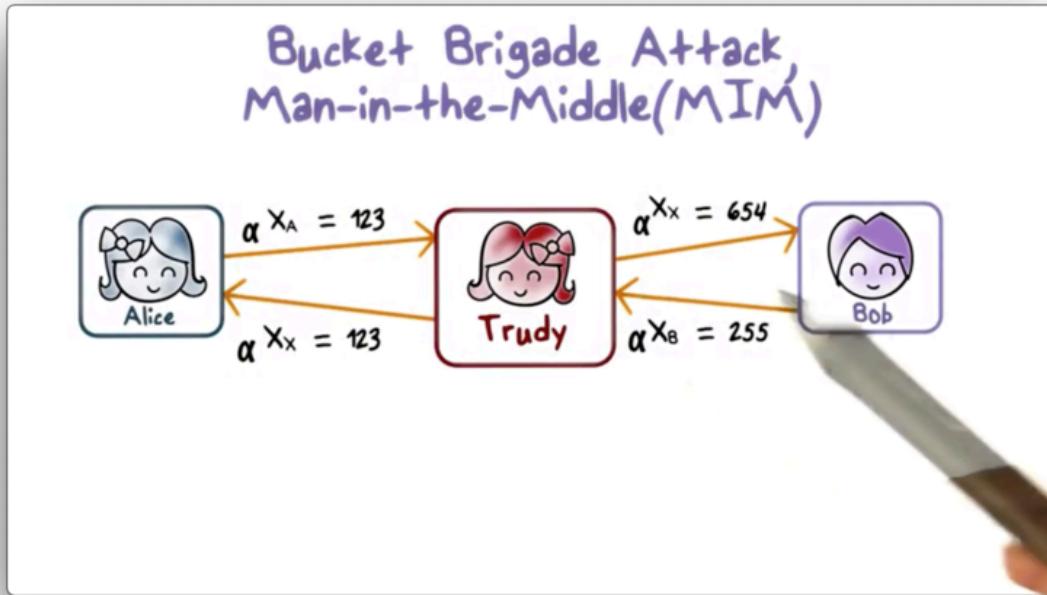
Suppose that Alice tells Bob to use Diffie-Hellman. The first thing that Bob has to do is compute  $Y_B$  from his local secret  $X_B$ , and this computation involves a very CPU-intensive exponentiation calculation.

If Alice is a malicious attacker, and Bob is a server, then Alice can request multiple simultaneous Diffie-Hellman sessions with Bob. These requests would cause Bob to waste many CPU cycles on exponentiation, which can result in denial of service.

Additionally, the sole purpose of Diffie-Hellman is key exchange. Unlike RSA, it does not offer encryption and cannot produce digital signatures.

### 14.29 Bucket Brigade Attack, Man in the Middle (MIM)

The biggest threat to the Diffie-Hellman key exchange is the **bucket brigade attack**, which is a type of [man-in-the-middle attack](#).



In this attack, Trudy intercepts the message  $Y_A$  that Alice sends to Bob, and instead sends her own  $Y_X$  to Bob, fooling Bob to accept this as  $Y_A$ . Likewise, Trudy intercepts  $Y_B$  that Bob sends to Alice and instead sends her own  $Y_X$  to Alice, fooling Alice to believe that  $Y_X$  is actually  $Y_B$ .

The result is that the shared key that Alice computes is the shared key between Alice and Trudy. Similarly, the shared key that Bob computes is the shared key between him and Trudy. In other words, Trudy plays Bob to Alice and Alice to Bob.

This man-in-the-middle attack is possible because the Diffie-Hellman key exchange protocol does not authenticate Alice or Bob. There is no way for Alice to know that the message that she has received is from Bob, and vice versa.

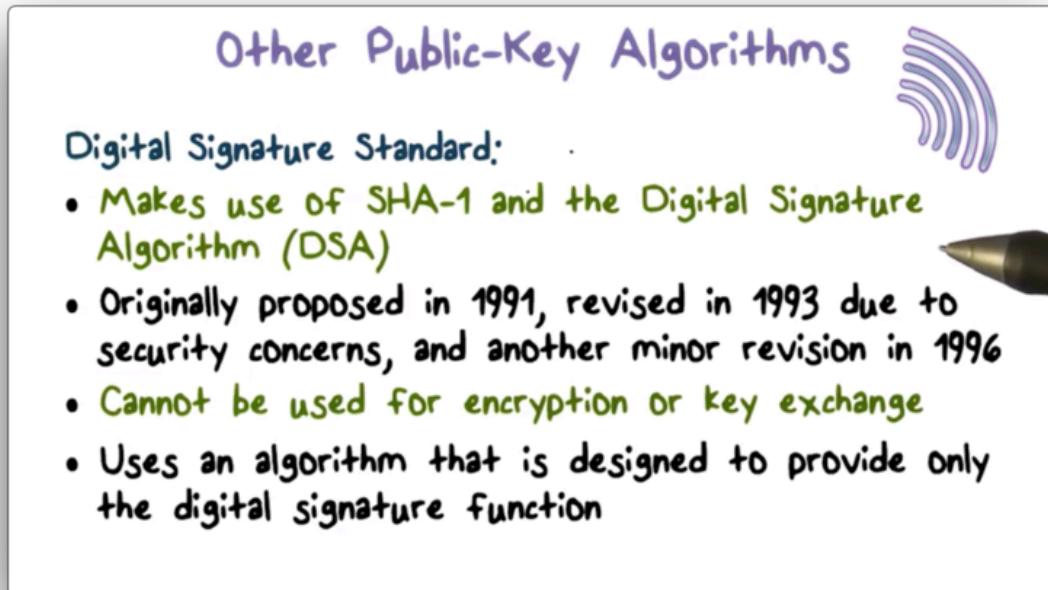
There are several ways to fix this problem. For example, everyone can publish their public key to a trusted, public repository instead of sending it and risking interception and forgery.

Alternatively, if Alice has already published her RSA public key, she can sign  $Y_A$  when she sends it to Bob. Bob can verify that  $Y_A$  is really from Alice using her RSA public key.

### 14.30 Other Public Key Algorithms

The **Digital Signature Standard** (DSS) is based on the secure hash function **SHA-1**. This algorithm is only used for digital signature, not encryption or key exchange.

## Other Public-Key Algorithms



**Digital Signature Standard:**

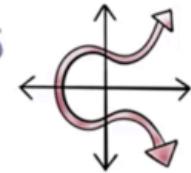
- Makes use of SHA-1 and the Digital Signature Algorithm (DSA)
- Originally proposed in 1991, revised in 1993 due to security concerns, and another minor revision in 1996
- Cannot be used for encryption or key exchange
- Uses an algorithm that is designed to provide only the digital signature function

While RSA is the most widely used public-key algorithm for encryption and digital signature, a competing system known as **Elliptic-Curve Cryptography** (ECC), has recently begun to challenge RSA. The main advantage of ECC over RSA is that it offers the same security with a far smaller bit size.

On the other hand, RSA has been subject to a lot of cryptanalysis work over the years. For ECC, the cryptanalysis work is still just beginning; therefore, we are not as confident in ECC as we are in RSA.

## Other Public-Key Algorithms

### Elliptic-Curve Cryptography (ECC):



- Equal security for smaller bit size than RSA
- Seen in standards such as IEEE P1363
- Confidence level in ECC is not yet as high as that in RSA
- Based on a mathematical construct known as the elliptic curve

### 14.31 RSA, Diffie-Hellman Quiz



#### RSA, Diffie-Hellman Quiz

Check the statements that are True:

- RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and  $n - 1$  for some  $n$
- If someone invents a very efficient method to factor large integers, then RSA becomes insecure
- The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms
- The Diffie-Hellman key exchange protocol is vulnerable to a man-in-the-middle attack because it does not authenticate the participants
- RSA and Diffie-Hellman are the only public-key algorithms



### 14.32 RSA, Diffie-Hellman Quiz Solution



### RSA, Diffie-Hellman Quiz

Check the statements that are True:

- RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and  $n - 1$  for some  $n$
- If someone invents a very efficient method to factor large integers, then RSA becomes insecure
- The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms
- The Diffie-Hellman key exchange protocol is vulnerable to a man-in-the-middle attack because it does not authenticate the participants
- RSA and Diffie-Hellman are the only public-key algorithms



## 15 Hashes

### 15.1 Hash Functions

A **hash function** can be applied to a block of data of any size and produces a fixed-size output, typically in the range of 128-512 bits. Given a message  $m$ , computing the hash  $H(m)$  should be very easy.

These properties make hash functions practical for security applications. We need our hash functions to be able to handle arbitrary input and compute their output efficiently.

The following properties are required for hash functions to be secure.

#### 15.1.1 One-Way Function

Given  $H(m)$ , it should be computationally infeasible to find  $m$ . We call such a non-invertible function a **one-way function**.

Alice can authenticate a message by hashing a secret value and appending it to the message. If an attacker intercepts this message and the hash function is not one-way, then the attacker can find the

input that computes the hash value. That is, the attacker can invert the hash function to obtain the secret value from its hash.

### 15.1.2 Weak Collision-Resistant Property

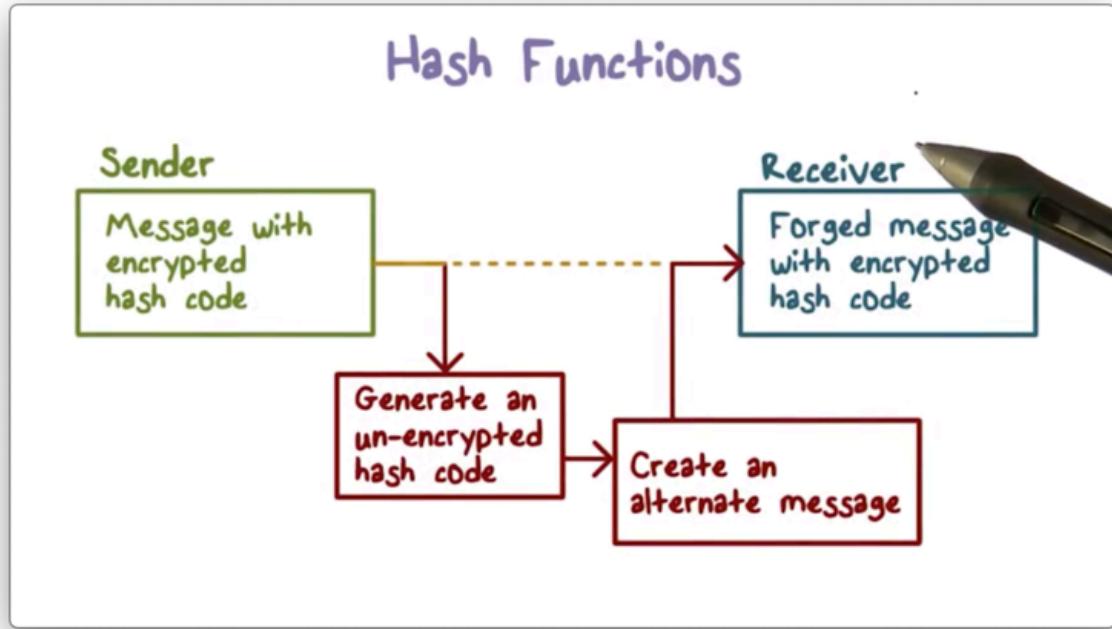
Given input data  $m_1$ , it should be computationally infeasible to find another input value  $m_2 \neq m_1$ , such that  $H(m_1) = H(m_2)$ . This property of hash functions is the **weak collision-resistant property**.

Suppose that Alice wants to send a message to Bob, and she wants to ensure that the message arrives to him unmodified. She can create an encrypted hash of the message and send both the message and the hash to Bob.

When the attacker intercepts the message and encrypted hash code, they can recompute the (unencrypted) hash code of the message. If the weak collision property is not present, then the attacker can find another message such that the hash value of that message is the same as the hash value of the original message.

Even though the hash code is encrypted, the attacker only needs to find a message that reproduces the unencrypted hash code, since the encryption of two identical hash codes produces the same result.

As a result, the receiver is unable to tell that an attacker has modified the message because the modified message has the same hash code as the original message, and thus the same encrypted hash code.



### 15.1.3 Strong Collision-Resistant Property

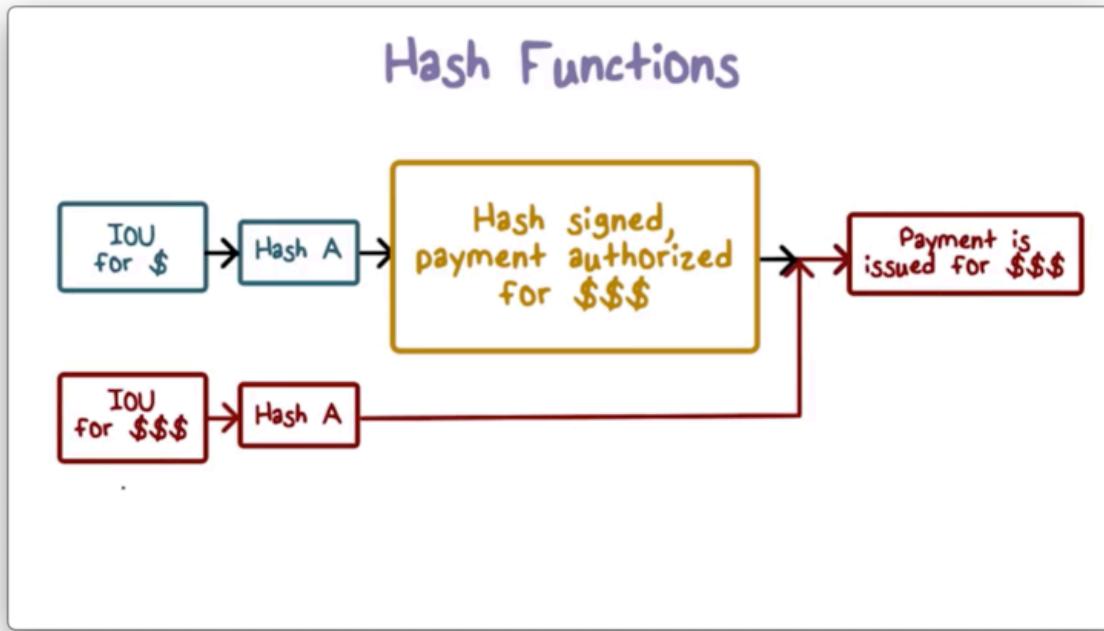
The **strong collision-resistant property** states that it should be computationally infeasible to find two different inputs  $m_1 \neq m_2$ , such that  $H(m_1) = H(m_2)$ .

The strong collision-resistant property implies the weak collision-resistant property; that is, while the latter only requires that a hash function is collision-resistant regarding a specific input message, the former requires collision resistance for any pair of given messages. Thus, a hash function cannot meet the strong collision-resistant property without also satisfying the weak collision-resistant property.

Suppose that Bob does some work for Alice, and he wants her to pay him later. He can draft an IOU message and ask Alice to digitally sign it as a way of agreeing to pay it. To sign the IOU, Alice first hashes it and then signs the hash using her private key. Later, Bob can present this IOU message, along with the signature, to Alice or Alice's bank to retrieve the payment.

If the strong collision-resistant property is not present, Bob can find two different messages that have the same hash code: one requesting Alice to pay a small amount  $X$ , and another requesting her to pay a more substantial amount  $XXX$ .

Bob can present the IOU for  $X$  to Alice and have Alice sign it. Later, Bob can present the signature with the message for  $XXX$  and ask Alice to pay it. Because the two messages have the same hash value, Alice cannot deny that she has signed the message agreeing to pay  $XXX$ .



## 15.2 Hash Function Weaknesses

To understand the constraints and potential weaknesses of hash functions, we need to understand two concepts: the pigeonhole principle, and the birthday paradox.

### 15.2.1 Pigeonhole Principle

Imagine we have nine pigeonholes. If we have nine pigeons, then we can place each pigeon in one hole. Generally, for  $n$  pigeons and  $m$  holes, each pigeon can occupy an unoccupied hole when  $m = n$ .

If we add another pigeon, then there must be one hole occupied by two pigeons. Generally, if  $n > m$ , then at least one hole must have more than one pigeon.

### 15.2.2 Birthday Paradox

How many people do you need in a room before you have a greater than 50% chance that two of them have the same birthday?

There are 365 birthdays, and we can think of these birthdays as pigeonholes. Then, the probability that two people in the room have the same birthday is the probability that two people occupy one pigeonhole.

Obviously, with 366 people, there is a 100% chance that two people have the same birthday.

If we only want a good chance, say 50%, that two people share a birthday, how do we calculate how many people we need in the room?

We can reframe the problem to ask instead: how many people do we need in a room before the probability  $p$  that everyone has a unique birthday drops below 50%?

When we place the first person  $f$  in the room,  $p = 1$ . Since there are no other people in the room,  $f$  is guaranteed to have a unique birthday.

When we place the second person  $s$  in the room, there are only 364 unique days left, since  $f$  has already taken one day. The probability that  $s$  doesn't share  $f$ 's birthday is  $364/365$ , and thus the overall probability  $p$  drops from 1 to  $364/365$ .

The third person,  $t$ , is in a similar situation. Since two people precede them, there are only 363 available birthdays left. As a result,  $p$  drops to  $364/365 * 363/365$ . We multiply the probabilities because we need to make sure that  $s$  doesn't violate the uniqueness constraint *and*  $t$  doesn't either. Technically, this is a conditional probability.

Generally, for  $k$  possible birthdays and  $n$  individuals in the room, we can calculate the  $p$  as:

$$1 \ p = (1 / k)^n * (k! / (k - n)!)$$

We can steadily increase  $n$  to determine when  $p$  dips below 0.5. For  $k = 365$  and  $n = 23$ , we see that:

$$1 \ p = (1 / 365)^{23} * (365! / (365 - 23)!) = 0.492703$$

Since  $p$  is the probability that no birthdays overlap, the probability that one or more birthdays overlap is  $1 - p = 0.507297$ .

Analytically, the probability of repetition given  $k$  possible events and  $n$  selected events, is approximately  $n^2/2k$ . This probability is 0.5 when  $n = \sqrt{k}$ . For example, if  $k = 365$ , then  $\sqrt{k} = 19$  approximately, which is close to the correct answer of 23.

### 15.2.3 Back to Hash Functions

Once we understand the pigeonhole principle and the birthday paradox, we see that some of the properties of hash functions seem to contradict each other.

Remember, a hash function produces a fixed-size output from an input of arbitrary size. Since there are an infinite number of arbitrarily-sized inputs, and a finite number of fixed-size hash codes, many inputs can be mapped to the same output hash value. In other words, we have many more pigeons than pigeonholes.

This setup seems to violate the property of collision resistance. However, the collision resistance properties only speak to computational infeasibility, not mathematical impossibility. From a security perspective, a collision that is infeasible to find is as good as one that doesn't exist.

The larger the number of output hash values, the harder it is to find a collision. We can reduce the probability of a collision by increasing the length of the hash code.

### 15.3 Determining Hash Length

To reduce the chance of collision, we need to use hash functions that produce longer hash values, but how many bits is enough?

Suppose a hash function produces hash codes that are  $l$  bits long. As a result, this function can produce  $2^l$  possible hash values. According to the birthday paradox, we have a 50% chance of finding a collision after processing  $\sqrt{2^l} = 2^{l/2}$  messages.

Therefore, if  $l = 64$ , then there are  $2^{64}$  hash values, and an attacker only needs to search  $2^{32}$  messages to find a collision. This search is entirely feasible with modern computing power. In practice, hash values are at least 128 bits long.

### 15.4 Hash Size Quiz



### Hash Size Quiz

Choose the correct answer:

If the length of hash is 128 bits, then how many messages does an attacker need to search in order to find two that share the same hash?

128        $2^{127}$   
  $2^{128}$         $2^{64}$

### 15.5 Hash Size Quiz Solution



**Hash Size Quiz**  
Choose the correct answer:

If the length of hash is 128 bits, then how many messages does an attacker need to search in order to find two that share the same hash?

128        $2^{127}$   
  $2^{128}$         $2^{64}$

Given a hash length  $n$ , an attacker needs to hash  $2^{n/2}$  messages to find a collision. For  $n = 128$ , an attacker needs to compute  $2^{64}$  hashes.

### 15.6 Secure Hash Algorithm

The original secure hash algorithm is SHA-1, which produces a hash value of 160 bits. The subsequent SHA-2 family of algorithms, which operate similarly to SHA-1, produce hashes of 256-, 384-, and 512-bits long.

The following table compares the SHA parameters.

## Comparison of SHA Parameters

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	80	80	80
Security	80	128	192	256

**Notes:**

1. All sizes are measured in bits.
2. Security refers to the fact that a birthday attack on a message digest of size  $n$  produces a collision with a work factor of approximately  $2^{n/2}$ .

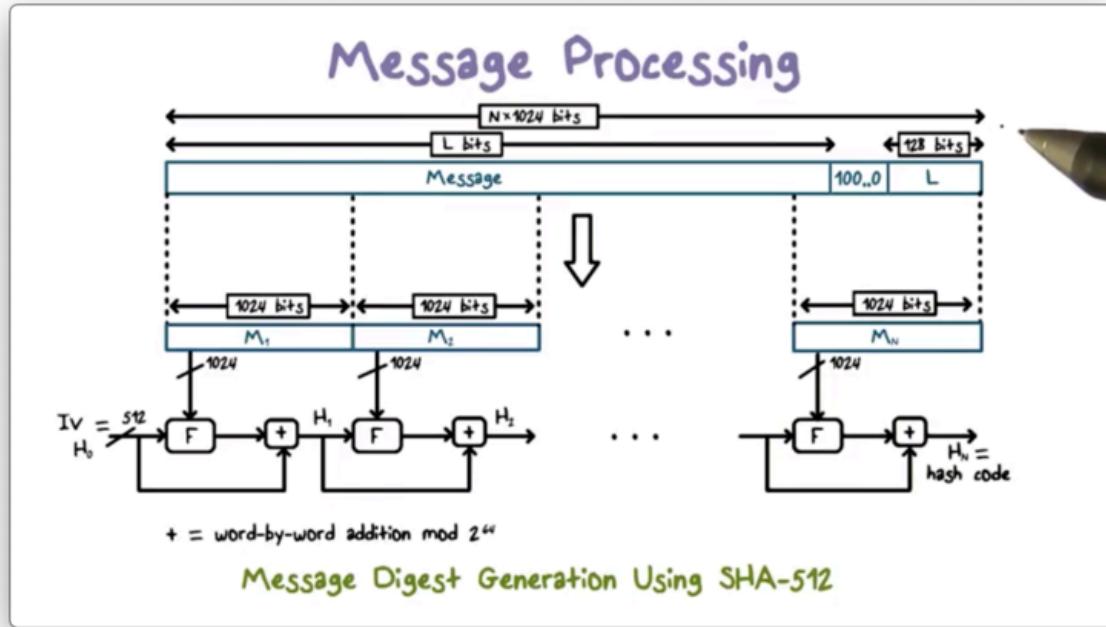
Message digest size refers to the length of the hash value. As we know, SHA-1 produces 160-bit hashes, and SHA-512 produces 512-bit hashes.

Message size is the size limit on the input. These limits do not have any effect in practice, because most, if not all, messages are much smaller. For example, SHA-1 caps the message size at  $2^{64}$  bits. Given that one terabyte is  $2^{43}$ , SHA-1 can hash a message that is over two million terabytes long.

As we move from left to right in the table, each subsequent algorithm produces a hash value with more bits. Consequently, each subsequent hash function is more secure than the last. For example, with a hash value size of 160 bits, the search space for an attacker to find a collision with SHA-1 is  $2^{80}$  messages. For SHA-512, that search space grows to  $2^{256}$  messages.

### 15.7 Message Processing

The following figure shows the overall processing of a message to produce a hash value.

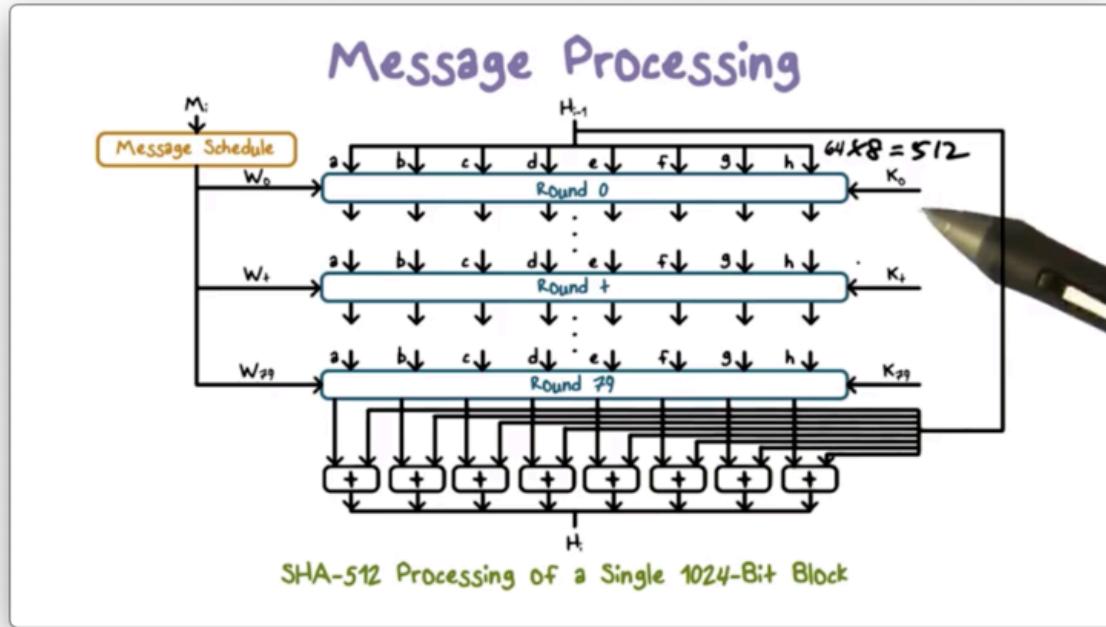


Before any processing can start, we must pad the message to a multiple of 1024 because the hash function processes the message in 1024-bit blocks. First, we store the length of the original message in the final 128 bits of the last block. Next, we fill in the space between the end of the original message and the last 128 bits with 1 followed by as many 0s as necessary.

After padding, the message is processed one 1024-bit block at a time. The output of processing the current block becomes the input to processing the next block. That is, when processing the second block, the input includes not only the second message block but also the output of the processing of the first message block.

For the first message block, the input includes an **initialization vector** (IV), a 512-bit value hardcoded in the algorithm.

The following figure shows the processing of a single message block.



Processing a single block involves 80 rounds of operations that operate on both the current message block  $m$  and the output of processing the previous block.

The inputs to each round include the result from the previous round, some constant  $k$ , and some words  $w$  derived from  $m$ .  $k$  provides randomized values to eliminate any regularities in  $m$ .

The operations at each round include **circular shifts** and primitive boolean functions based on AND, OR, NOT, and XOR.

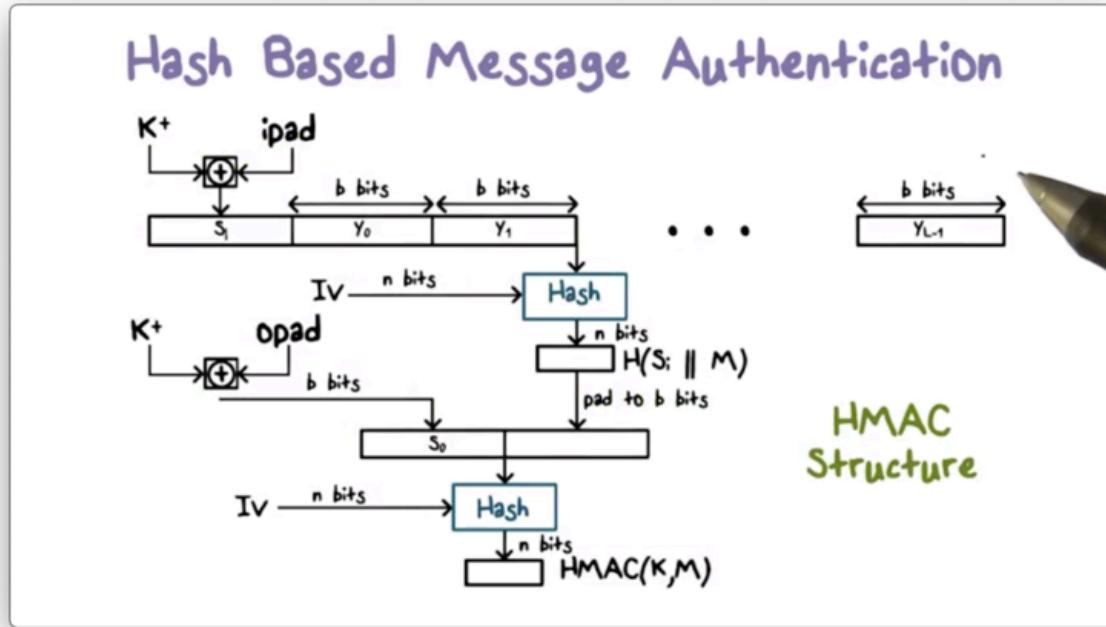
The result of processing the current block becomes input for processing the next block, and the result of processing the final block is the hash code for the entire message.

## 15.8 Hash Based Message Authentication

Using hash values for message authentication has several advantages. For example, hash functions are very efficient to compute, and libraries for hash functions are widely available. On the other hand, a hash function such as SHA cannot be used directly for message authentication because it does not rely on a secret.

There have been several proposals to incorporate a secret key into an existing hash function. HMAC has received the most support thus far and has been adopted for use in other protocols such as IPSec and TLS.

The following diagram illustrates how HMAC works.



HMAC involves a hash function  $H$  and a secret key  $k$ . The message  $m$  consists of multiple blocks of  $b$  bits. For example, in SHA-512, each block is 1024 bits, so  $b = 1024$ .

First,  $k$  is padded to  $b$  bits, which is accomplished by appending zeroes to  $k$ . The padded key is then XORed with  $ipad$ , a constant designed to eliminate any regularities in the key.

The result is a  $b$ -bit value  $s_1$ .  $s_1$  is prepended to  $m$ , and then the combination  $s_1 + m$  is hashed using  $H$  to produce an  $n$ -bit hash value. For example, if the hash function is SHA-512, then  $n = 512$ .

Next, the  $n$ -bit hash value is padded to  $b$  bits. The padded key is then XORed with  $opad$ , another constant designed to eliminate regularities in the key. The result is a  $b$ -bit value  $s_0$ .

The padded hash  $h$  is then appended to  $s_0$ , and the entire message  $s_0 + h$  is hashed. The  $n$ -bit result is the output of the HMAC procedure.

## 15.9 HMAC Security

The security of HMAC depends on the cryptographic strength of the underlying hash function. For HMAC to be secure, the underlying hash function must satisfy the properties of non-invertibility and collision resistance.

Furthermore, compared with a cryptographic hash function, it is much harder to launch a successful collision attack on HMAC because HMAC uses a secret key.

For example, suppose the attacker obtains the HMAC of message  $m_1$  and wants to find another message  $m_2$  such that  $HMAC(m_2) = HMAC(m_1)$ .

Without the secret key, there is no way the attacker can compute the correct HMAC value for  $m_2$ . Thus, the attacker can't determine whether  $m_1$  and  $m_2$  have collision in HMAC.

In summary, because of the use of the secret key, HMAC is much more secure than a cryptographic hash function alone.

### 15.10 Hash Function Quiz



#### Hash Function Quiz

Check the statements that are True:

- The one-way hash function is important not only in message authentication but also in digital signatures
- SHA processes the input one block at a time but each block goes through the same processing
- HMAC is secure provided that the embedded hash function has good cryptographic strengths such as one-way and collision resistant

### 15.11 Hash Function Quiz Solution



### Hash Function Quiz

Check the statements that are True:

The one-way hash function is important not only in message authentication but also in digital signatures

SHA processes the input one block at a time but each block goes through the same processing

HMAC is secure provided that the embedded hash function has good cryptographic strengths such as one-way and collision resistant



## 16 Security Protocols

### 16.1 Why Security Protocols

A network protocol defines the rules and conventions for communications between two parties. A security protocol defines the rules and conventions for secure communications between two parties.

Suppose Alice and Bob want to communicate securely over the Internet. They first need to authenticate each other, so that Alice knows she is communicating with Bob and Bob knows that he is communicating with Alice.

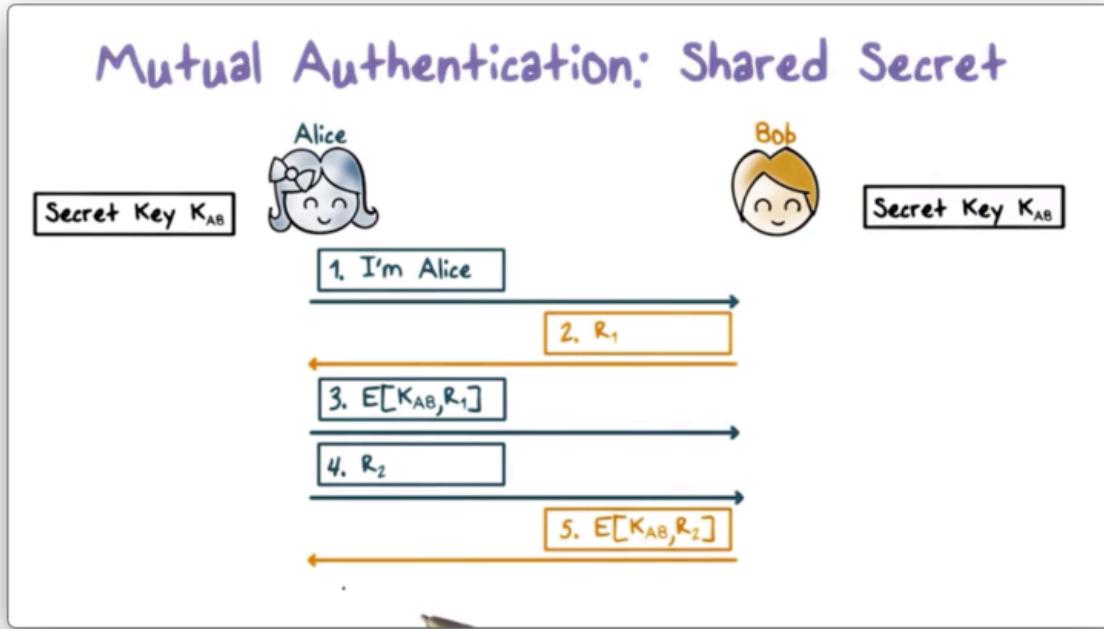
Further, if they want to communicate securely, they likely want to encrypt their messages. Therefore, they need to establish and exchange keys and agree on what cryptographic operations and algorithms to use.

The building blocks of these security protocols are the public-key and secret-key algorithms, as well as hash functions, all of which we have previously discussed.

## 16.2 Mutual Authentication: Shared Secret

In mutual authentication, Alice needs to prove to Bob that she is Alice, and Bob needs to prove to Alice that he is Bob.

Suppose that Alice and Bob share a secret key  $K_{AB}$ , which only they know. Using this secret key, we can envision the following authentication protocol.



First, Alice sends a message to Bob, claiming that she is Alice. Bob response with a random value  $r_1$ , referred to as a **challenge**. Alice encrypts  $r_1$  with  $K_{AB}$  and sends the ciphertext back to Bob as a **response** to the challenge. When Bob receives the response, he decrypts it with  $K_{AB}$  and sees if it matches the plaintext  $r_1$ .

If there is a match, Bob knows that he must be communicating with Alice, since she is the only other person with knowledge of  $K_{AB}$ . Without  $K_{AB}$ ,  $r_1$  cannot be encrypted in such a way that Bob can recover it with decryption using  $K_{AB}$ .

Next, Alice must authenticate Bob, and she starts by sending him a random value  $r_2$ . Bob encrypts  $r_2$  with  $K_{AB}$  and sends the ciphertext to Alice. Upon receiving the response from Bob, Alice decrypts the ciphertext and checks if the plaintext matches  $r_2$ . If so, she can be confident that she is communicating with Bob.

Note that there are cases where only one-way authentication is required. For example, if Alice is a client and Bob is a server, Alice may need to authenticate with Bob, but Bob might not authenticate

with Alice. In this case, the authentication process ends after the response to  $r_1$  is received.

If Alice is a human user, then typically  $K_{AB}$  can be derived from a password hash that is known to Bob.

The challenge values  $r_1$  and  $r_2$  should not be easily repeatable or predictable; otherwise, an intruder Trudy can record the challenge and response between Alice and Bob and replay them to impersonate Alice or Bob.

Suppose Trudy has spent time recording the authentication messages between Alice and Bob across multiple sessions, and one of the challenge-response pairs that she recorded is  $\{a, b\}$ . If Bob sends  $a$  again, Trudy can impersonate Alice by responding with  $b$ .

As another example, suppose that the challenge value always increases. Trudy can first impersonate Bob and send a large challenge value, say  $r_1 = 1000$ , and record the response from Alice. Meanwhile, the real Bob is using a smaller value, such as  $r_1 = 950$ . When the real Bob finally sends  $r_1 = 1000$  sometime in the future, Trudy can replay the response she received earlier from Alice.

We assume that Trudy can intercept any messages delivered over the Internet, and our goal is to prevent Trudy from replaying these messages and impersonating Alice or Bob. We can achieve this goal by avoiding repeatable or predictable values for  $r_1$  and  $r_2$ . Typically, we use large random values.

Additionally, we need to protect the shared secret key  $K_{AB}$ . If Trudy can steal a copy of  $K_{AB}$  from either Alice's or Bob's machine, then she can impersonate both Alice and Bob. The security of the endpoints is as important as the security of the communication between the two endpoints.

### 16.3 Mutual Authentication Quiz



#### Mutual Authentication Quiz

Mark T for True or F for False:



- The challenge values used in an authentication protocol can be repeatedly used in multiple sessions
- The authentication messages can be captured and replayed by an adversary
- Authentication can be one-way, e.g., only authenticating Alice to Bob

#### 16.4 Mutual Authentication Quiz Solution



### Mutual Authentication Quiz

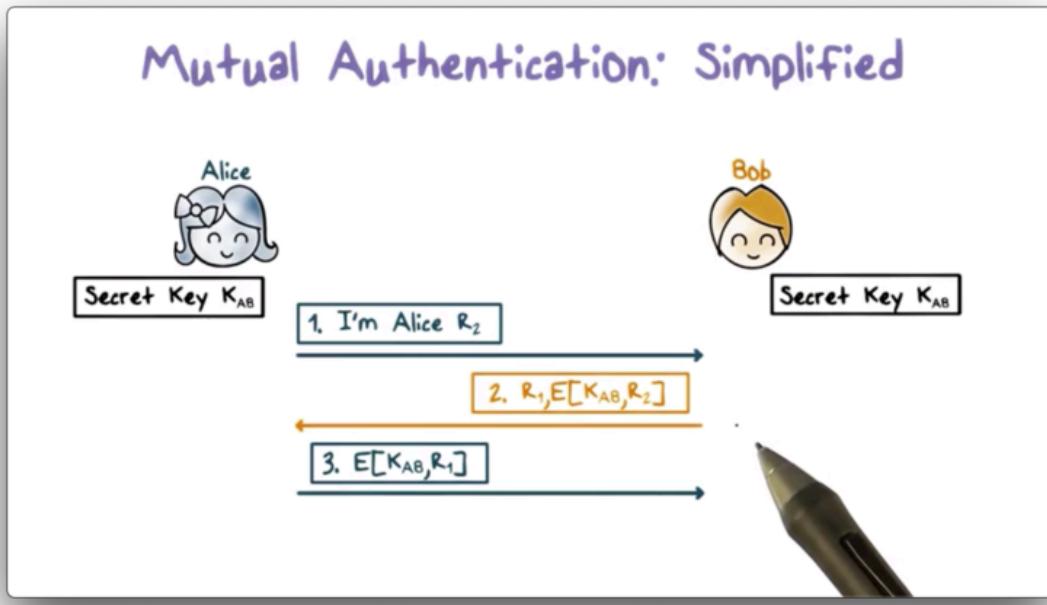
Mark T for True or F for False:

- F The challenge values used in an authentication protocol can be repeatedly used in multiple sessions
- T The authentication messages can be captured and replayed by an adversary
- T Authentication can be one-way, e.g., only authenticating Alice to Bob



#### 16.5 Mutual Authentication: Simplified

The mutual authentication protocol described above takes five steps. Can we shorten it to require only the following three steps?



First, Alice presents her identity to Bob and sends a challenge  $r_2$ . Bob responds with both the ciphertext of  $r_2$  generated from  $K_{AB}$  as well as his challenge  $r_1$ . Upon receiving the response, Alice decrypts the ciphertext using  $K_{AB}$  and validates that the plaintext matches  $r_2$ . Third, Alice sends Bob the ciphertext of  $r_1$ , which Bob decrypts and compares with plaintext  $r_1$  to authenticate Alice.

This protocol is susceptible to a **reflection attack** - a kind of **man in the middle attack** - that works as follows.

First, Trudy impersonates Alice and sends challenge  $r_2$  to Bob. According to the protocol, Trudy will be stuck at the final step of the exchange because she cannot encrypt the challenge  $r_1$  sent from Bob without  $K_{AB}$ .

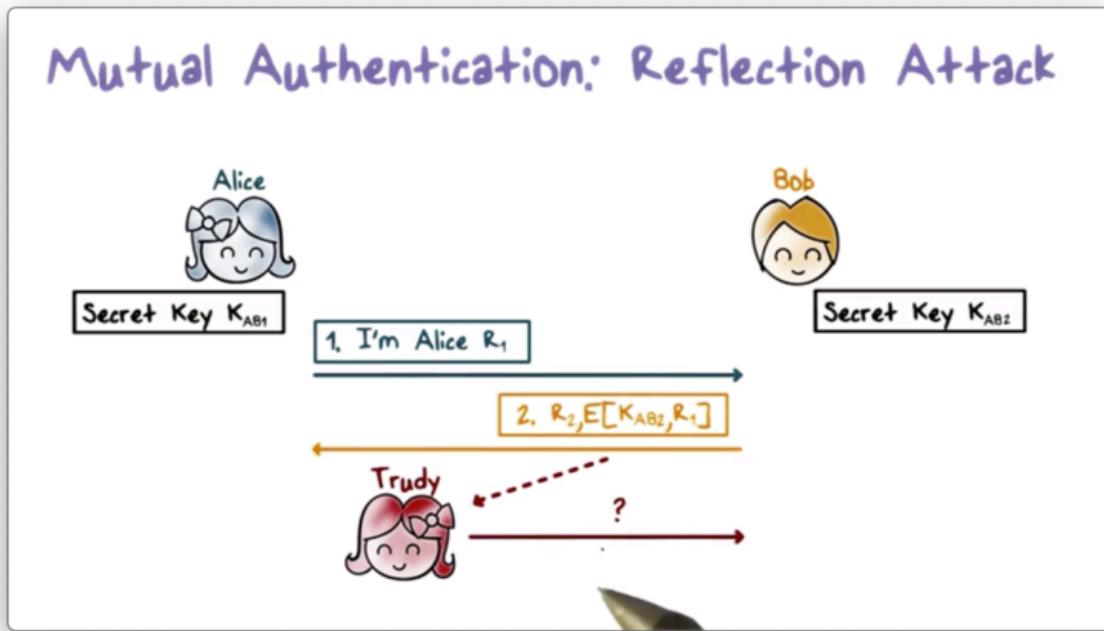
Next, Trudy opens another connection to Bob, again impersonating Alice. Trudy sends Bob the challenge  $r_1$  that he just sent her in the first connection. Bob responds with the ciphertext for  $r_1$  and a new challenge  $r_3$ .

Remember, Trudy needed the ciphertext for  $r_1$  to complete the authentication exchange in the first connection. She tricked Bob into responding to his own challenge in the second connection. Thus, Trudy can take the ciphertext from the second connection and send it back to Bob in the first connection.

This type of attack is referred to as a reflection attack because Trudy reflects Bob's challenge from the first connection back to him in the second connection.

### 16.5.1 Preventing Reflection Attacks

One strategy to defend against reflection attacks is to use two different shared keys: one for the initiator of the connection and one for the responder.



Suppose Alice uses  $K_{AB1}$ , and Bob uses  $K_{AB2}$ . In this setup, Bob sends ciphertext encrypted using  $K_{AB2}$  and expects to receive ciphertext encrypted with  $K_{AB1}$ . Even though Trudy can intercept the challenge sent from Bob, she cannot obtain the correct ciphertext through reflection.

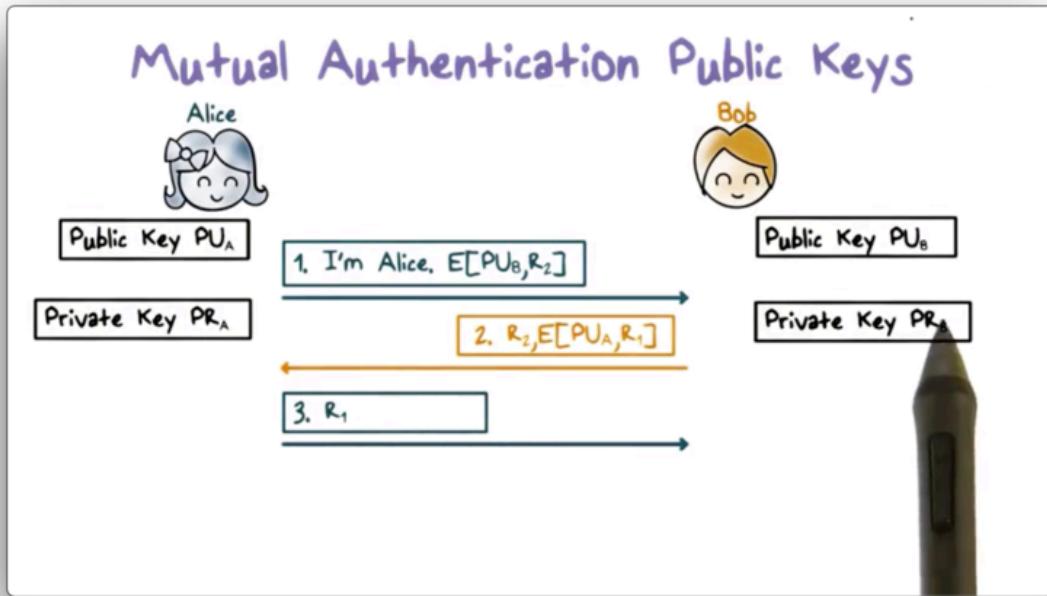
For example, if she sends him his own challenge, he will encrypt it with  $K_{AB2}$ . Unfortunately for Trudy, she needs the challenge encrypted with  $K_{AB1}$ , as this is the key used by Alice in responding to challenges from Bob. Since the challenge is encrypted with the wrong key, Trudy can't use it to impersonate Alice.

Another way to prevent reflection attacks is to use different challenges for the initiator and responder. For example, we can use even-numbered challenges for Alice and odd-numbered challenges from Bob.

Therefore, when Trudy receives a challenge  $r_1$  from Bob, the challenge is an odd number, and she cannot reflect this challenge back to Bob since he is expecting an even number.

## 16.6 Mutual Authentication Public Keys

If Bob and Alice have each other's public key, they can use public-key cryptography for mutual authentication.



First, Alice sends Bob a challenge  $r_2$  encrypted with Bob's public key. Upon receiving the challenge, Bob decrypts the ciphertext using his private key, and sends back the plaintext challenge  $r_2$  along with his own challenge  $r_1$ .  $r_1$  is encrypted using Alice's public key.

When Alice receives the response from Bob, the plaintext  $r_2$  lets Alice know that she is communicating with Bob because only he has the private key that pairs with the public key that encrypted  $r_2$ .

Alice also decrypts the ciphertext for  $r_1$  using her private key and sends the plaintext  $r_1$  to Bob. Bob knows that he is communicating with Alice because only Alice has the private key that pairs with the public key used to encrypt  $r_1$ .

This protocol can be modified to use signing with private keys instead of encrypting with public keys.

### 16.7 Security Protocols Quiz



#### Security Protocols Quiz

Mark T for True or F for False:

- A reflection attack is a form of man-in-the-middle attack
- To defeat a reflection attack, we can use an odd number as challenge from the initiator and even number from the responder
- We can use signing with public keys to achieve mutual authentication



## 16.8 Security Protocols Quiz Solution



### Security Protocols Quiz

Mark T for True or F for False:

- T A reflection attack is a form of man-in-the-middle attack
- T To defeat a reflection attack, we can use an odd number as challenge from the initiator and even number from the responder
- T We can use signing with public keys to achieve mutual authentication



## 16.9 Session Keys

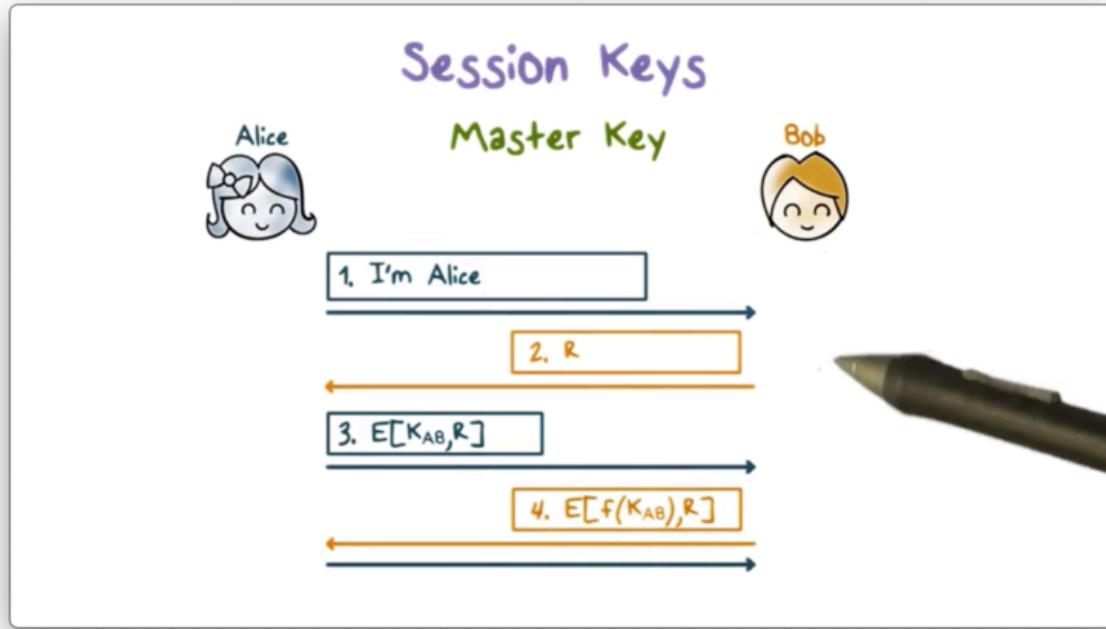
After authentication, Alice and Bob need to establish a shared secret key for their communication session so they can securely send messages to one another.

Typically, Alice and Bob share a long term secret key - called a *master key* - often derived from a password. Alice can use this master key  $K_{AB}$  to encrypt a new key  $K_S$ , which she and Bob can use for message encryption during their current communication session.

Alice and Bob should establish a new shared key for each session, even though they already share a master key. If the session key is leaked or broken, the impact is limited to the current session.

Intuitively, the more a secret is used, the higher the chance of a leak. Therefore, we should limit the use of the long-term master secret key, and only use it at the beginning of a session for authentication and establishing the session key.

Suppose Alice and Bob share a master key,  $K_{AB}$ . They can establish a session key as follows. Note that the first three steps are the same as before and serve for Bob to authenticate Alice.



After authentication, Bob and Alice compute the same session key based on both  $K_{AB}$  and something about the current session. Incorporating information about the current session into the generation of the session key helps to ensure that the key is unique to the session.

Bob then encrypts the challenge  $r$  using this new key  $K_S$  and sends it to Alice. If Alice can decrypt  $r$  using the version of  $K_S$  that she generated, then she knows that she and Bob have generated the same session key, and communication can proceed.

Alice and Bob can also use public-key cryptography to exchange a shared session key. For example, Alice can generate a key and encrypt it with Bob's public key so that only he can decrypt it. Alice then signs the encrypted key with her private key so that Bob can verify that she genuinely sent the key.

As a third alternative, Alice and Bob can use the Diffie-Hellman key exchange protocol to generate and exchange their session key.

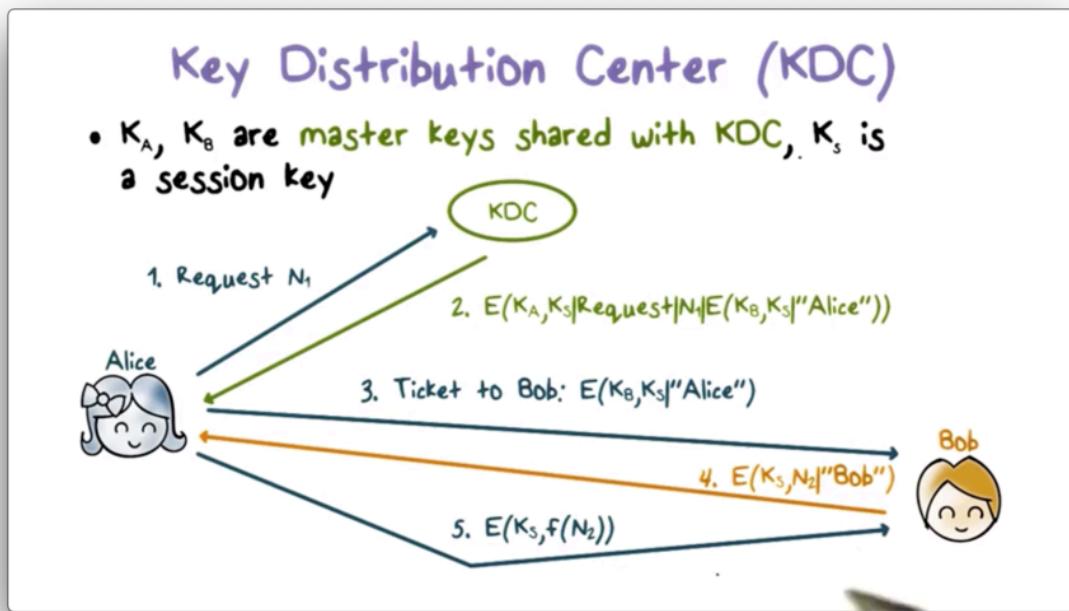
### 16.10 Key Distribution Center (KDC)

A major shortcoming of using pairwise key exchange based on a shared secret is that it doesn't scale well. For example, Alice needs to share a master key with Bob, Carol, and every other user with whom she wants to communicate.

A **key distribution center** (KDC) solves this scalability problem. In this setup, each party holds only one master key, which they share with the KDC, and the KDC holds master keys for each enrolled party.

For example, Alice may share  $K_A$  with the KDC, and Bob might share  $K_B$  with the KDC. Alice and Bob individually hold only one key, while the KDC holds both  $K_A$  and  $K_B$ .

If Alice and Bob want to have a secure session, they must first establish a session key  $K_S$ . The following diagram illustrates the steps involved in this operation.



First, Alice sends a message to the KDC containing both a request for a shared session key between her and Bob as well as a random **nonce** value  $n_1$ . The KDC sends a response encrypted using the master key  $K_A$  that it shares with Alice.

This response contains the session key  $K_S$  that the KDC just created for Alice and Bob to share. The response also contains the original request that Alice sent to KDC along with the nonce,  $n_1$ . Finally, the response contains a **ticket**: a message containing  $K_S$  and Alice's ID that is encrypted using the secret key  $K_B$  that Bob shares with the KDC.

When Alice receives this response, she can decrypt it because she holds  $K_A$ . She can verify that the message is not a replay by comparing the received request/nonce combination with the one she generated.

Alice then sends the ticket to Bob, which he can decrypt because he holds  $K_B$ . Bob can tell that the KDC generated the ticket because only the KDC can encrypt Alice's ID using  $K_B$  since it is the only other party in possession of that key.

Next, Bob creates a message containing a nonce  $n_2$  and his ID, encrypts it using  $K_S$ , and sends it to

Alice. When Alice receives this message and decrypts it successfully, she knows that she is communicating with Bob because only he can decrypt the ticket and retrieve  $K_S$ .

Finally, Alice performs an agreed-upon transformation of  $n_2$ , encrypts the result using  $K_S$ , and sends it back to Bob. This message proves to Bob that he is communicating with Alice because she is the only party that possesses  $K_S$  beside him.

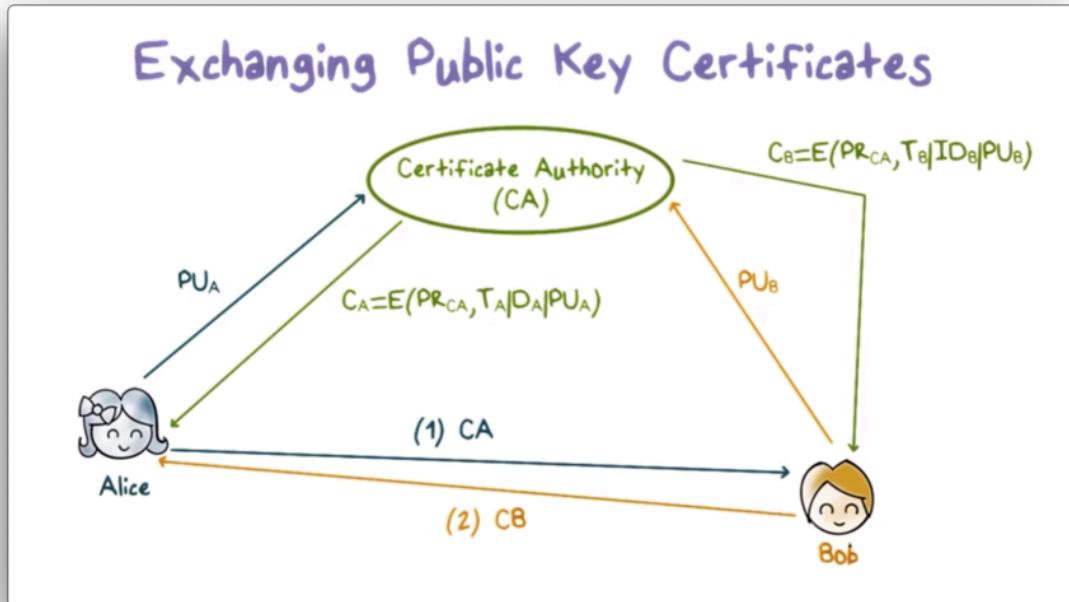
## 16.11 Exchanging Public Key Certificates

A **certificate authority** (CA) often manages the secure distribution of public keys.

First, Alice sends her public key to the CA, which verifies her identification and then sends her a certificate of her public key. The certificate is signed using the CA's private key and contains the certificate creation time and period of validity, as well as Alice's ID and public key.

Alice can then send this certificate to a user, such as Bob, or she can publish it so that any user can retrieve it. We assume that all users have the CA's public key, and therefore that Bob can use this public key to verify the certificate and obtain Alice's public key.

Likewise, Bob can obtain his certificate from the CA and send it to Alice so that Alice has Bob's public key as well.



### 16.12 Session Key Quiz



#### Session Key Quiz

Mark T for True or F for False:

- A session key should be a secret and unique to the session
- Authentication should be accomplished before key exchange
- A key benefit of using KDC is for scalability
- In order for Bob to verify Alice's public key, the certificate authority must be on-line
- Signing the message exchanges in Diffie-Hellman eliminates the man-in-the-middle attack



### 16.13 Session Key Quiz Solution



### Session Key Quiz

Mark T for True or F for False:

- T A session key should be a secret and unique to the session
- T Authentication should be accomplished before key exchange
- T A key benefit of using KDC is for scalability
- F In order for Bob to verify Alice's public key, the certificate authority must be on-line
- T Signing the message exchanges in Diffie-Hellman eliminates the man-in-the-middle attack

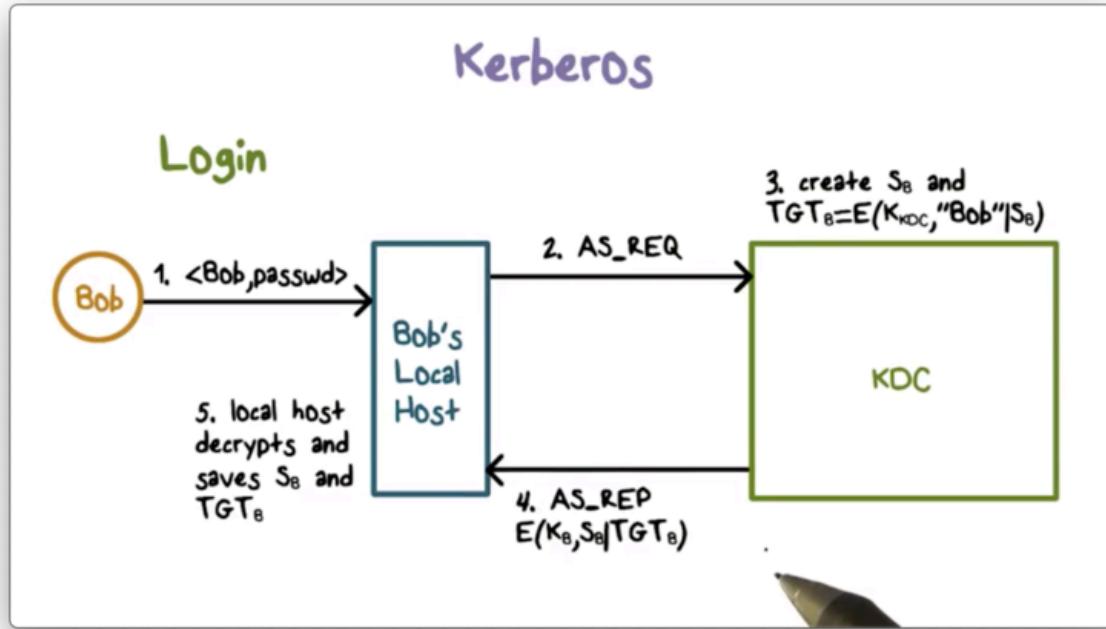
### 16.14 Kerberos

**Kerberos** is a standard protocol used to provide authentication and access control in a networked environment, such as an enterprise network.

Every entity in the network - every user and network resource - has a master key that it shares with the Kerberos servers, which perform both authentication and key distribution. That is, a Kerberos server functions as a KDC.

For human users, the master key is derived from his or her password, while for network devices, the key must be configured in. All the keys are stored securely in the KDC.

The following diagram summarizes the interactions that take place when Bob logs into a workstation backed by Kerberos.



When Bob logs in, his workstation first contacts the KDC with an authentication service request. The KDC generates a per-day session key,  $S_B$ , and a so-called **ticket-granting ticket** (TGT) that contains  $S_B$  and Bob's ID. This ticket is encrypted using the KDC's key.

The KDC then sends the authentication service response containing  $S_B$  and the TGT back to Bob's workstation. This message is encrypted using the master key  $K_B$  shared between Bob and the KDC.

Because  $K_B$  is shared only between Bob and the KDC, only Bob's local workstation can decrypt the authentication service response. After decryption, Bob's workstation stores  $S_B$  and the TGT.

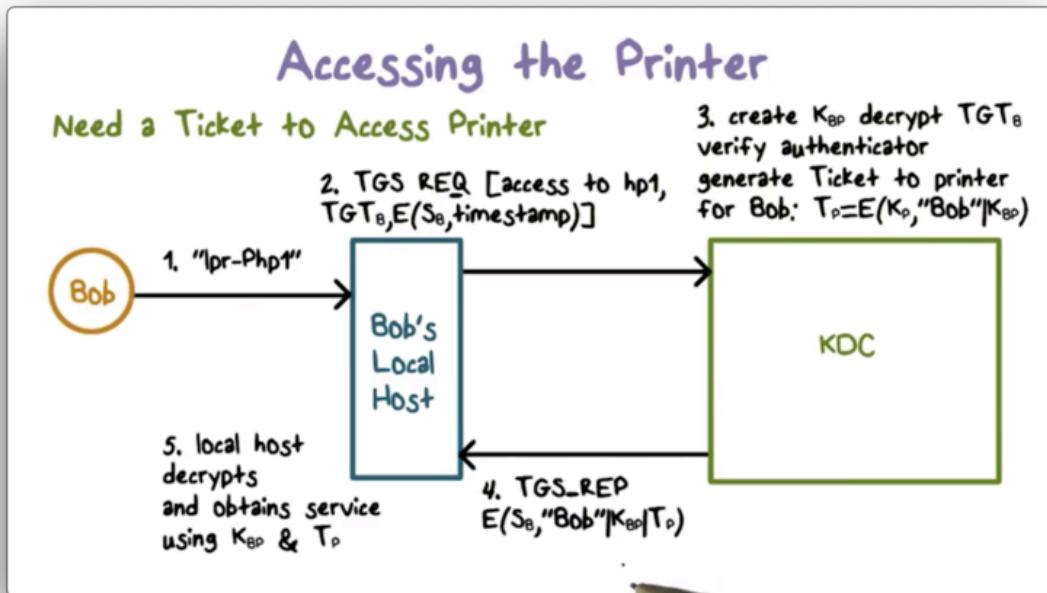
Bob's local workstation uses  $S_B$  for subsequent messages with the KDC and includes the TGT to convince the KDC to use  $S_B$ . Any new tickets from the KDC are encrypted using  $S_B$ .

There are several benefits to this setup. First, Bob's localhost does not need to store his password or password hash once Bob has logged in and obtained  $S_B$  from the KDC.

Second, the master key  $K_B$  that Bob shares with the KDC is only used once every day when Bob initially logs in. As a result, the exposure of  $K_B$ , which is derived from Bob's password and is subject to password-guessing attacks, is very limited.

## 16.15 Accessing the Printer

Suppose Bob wants to send a print job to a printer `hp1`.



His localhost sends a ticket-granting service request to the KDC. The request contains the TGT and an authenticator: the current timestamp encrypted using Bob's per-day session key  $S_B$ .

When the KDC receives the request, it decrypts the TGT using its private key  $K_{KDC}$ . The correct decryption of Bob's ID in the TGT verifies the validity of the TGT since only  $K_{KDC}$  could have encrypted it in the first place.

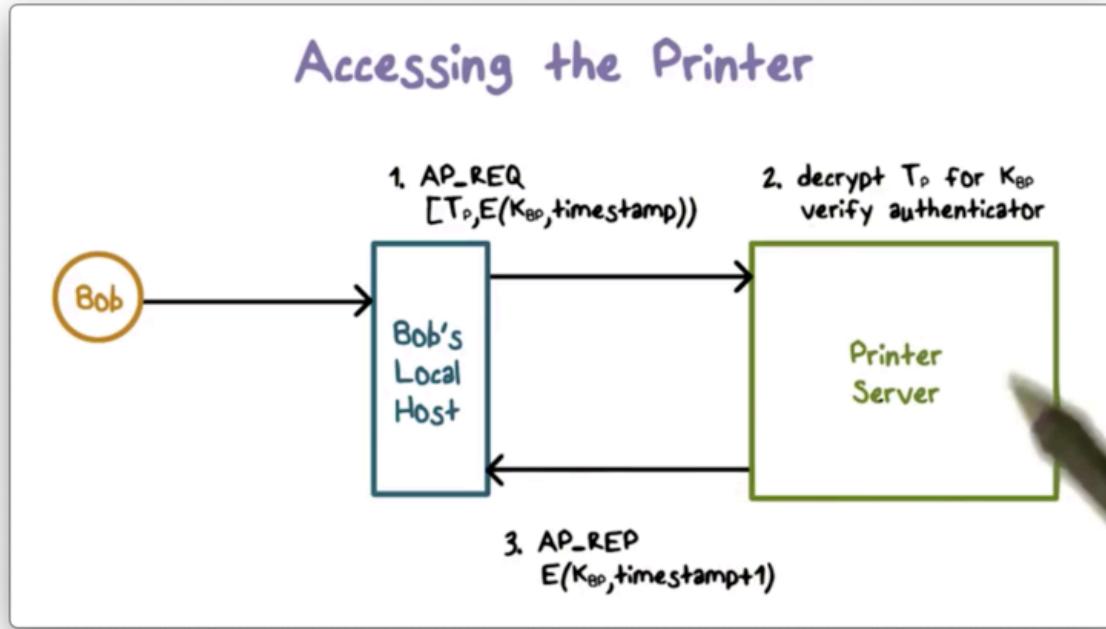
The KDC then uses  $S_B$  contained in the TGT to verify the authenticator by decrypting it and checking that the timestamp is current. This decryption verifies that Bob is the sender because only Bob has the key  $S_B$  that can encrypt the current timestamp properly.

The KDC then generates a ticket for Bob to communicate with the printer. This ticket contains a session key  $K_{BP}$  and Bob's ID, and is encrypted using the printer's master key  $K_P$ .

Note that a network device such as a printer has a long, random master key that is configured in, and is typically hard to guess or crack. Therefore, the tickets for these devices can be securely encrypted using their master keys.

The KDC sends a ticket-granting service response to Bob's localhost, which contains the session key  $K_{BP}$ , Bob's ID, and the ticket to the printer. The entire response is encrypted using  $S_B$ ; therefore, only Bob's local workstation can decrypt it.

Now Bob's localhost can authenticate itself to the printer.



First, it sends an authentication request to the printer containing the ticket and a new authenticator encrypted with  $K_{BP}$ .

The printer decrypts the received ticket using the master key  $K_P$  it shares with the KDC because the KDC encrypted the ticket with that key.

Since the printer can verify that the KDC created the ticket - only the KDC and the printer share  $K_P$  - the printer can verify that the KDC created the shared key  $K_{BP}$  contained in the ticket.

The print server uses  $K_{BP}$  to verify the authenticator by decrypting the ciphertext and verifying that the result matches the current timestamp.

The printer then sends a response to authenticate itself to Bob by, say, adding 1 to the current timestamp, and encrypting it with the shared key  $K_{BP}$ .

After these authentication steps, Bob's localhost can send the print job to the printer.

### 16.16 Kerberos Quiz



#### Kerberos Quiz

Mark T for True or F for False:

- Kerberos provides authentication and access control
- Kerberos also distributes session keys
- To avoid over-exposure of a user's master key, Kerberos uses a per-day key and a ticket-granting-ticket
- The authenticators used in requests to KDC and application servers can be omitted
- Access to any network resource requires a ticket issued by the KDC



### 16.17 Kerberos Quiz Solution



## Kerberos Quiz

Mark T for True or F for False:

- T Kerberos provides authentication and access control
- T Kerberos also distributes session keys
- T To avoid over-exposure of a user's master key, Kerberos uses a per-day key and a ticket-granting-ticket
- F The authenticators used in requests to KDC and application servers can be omitted
- T Access to any network resource requires a ticket issued by the KDC

## 17 IPSec and TLS

### 17.1 Goals of IPSec

If Alice receives a packet with Bob's source IP address, she cannot be sure that the packet is really from Bob. Since IPv4 does not enforce source IP address authentication, IP spoofing - forging a packet's source IP address - is a commonly used technique in cyber attacks.

For example, bots in a botnet can use source IP spoofing and DNS to mount a denial of service attack at a victim website. The bots query many different DNS servers requesting the full TXT record of a domain, which often contains many bytes of information. By spoofing the source IP address of their traffic to point to a victim website, the bots can direct the aggregate DNS response, which can be massive, to the victim website, overwhelming its servers.

IPSec provides security services at the IP layer, including

- authentication of source IP addresses
- confidentiality and integrity protection of packet data
- authenticity of packet data; in particular, preventing packet replay

Of course, a network application or protocol can implement its own specific security mechanisms to achieve these goals, but since all network applications must run on top of IP, IPSec ensures secure networking for the many applications that are ignorant about security.

## 17.2 Spoofing Quiz



### Spoofing Quiz

Mark the answer(s) that are correct.

IP Spoofing is useful for...

- Bidirectional communication
- Unidirectional communication

### 17.3 Spoofing Quiz Solution



**Spoofing Quiz**

Mark the answer(s) that are correct.

IP Spoofing is useful for...

Bidirectional communication

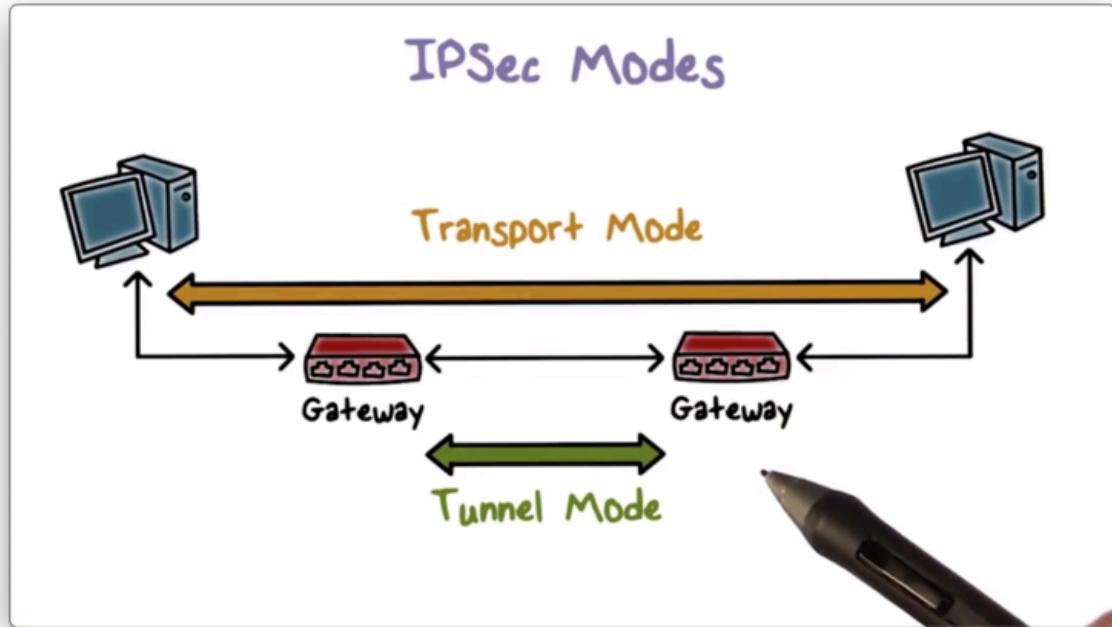
Unidirectional communication



If you spoof your IP address, responses to your packets will not reach you. Therefore, IP spoofing is only useful for unidirectional communication.

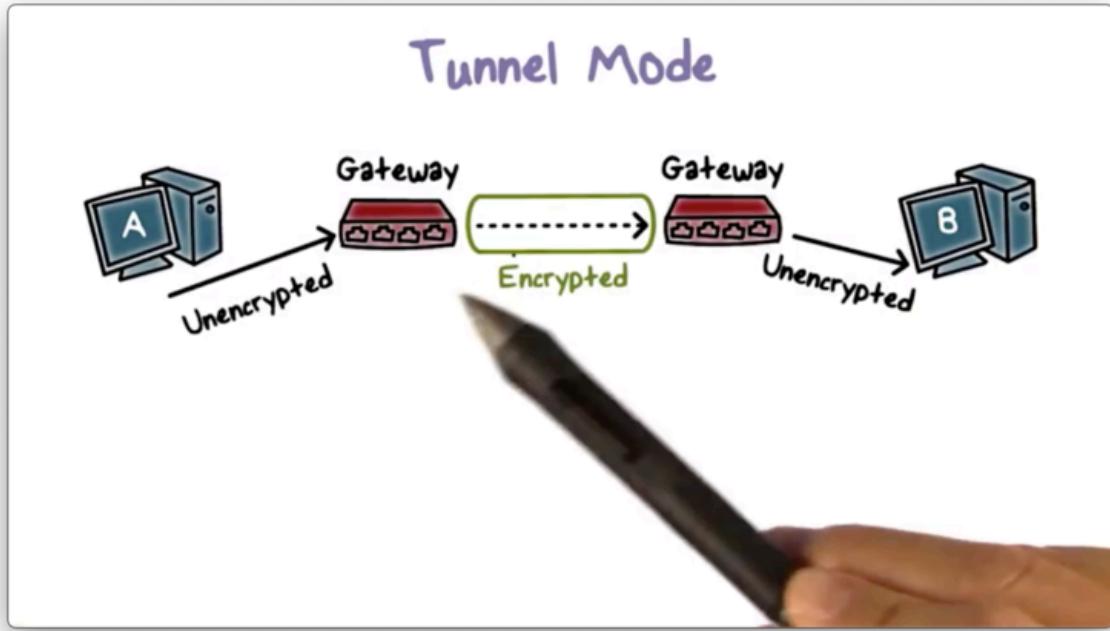
### 17.4 IPSec Modes

IPSec supports two operational modes: transport mode and tunnel mode. In **transport mode**, security protection is provided to traffic end to end, from one host to another. In **tunnel mode**, the protection typically is provided to traffic from the gateway of one network to the gateway of another network. Virtual private networks (VPNs) utilize tunnel mode.



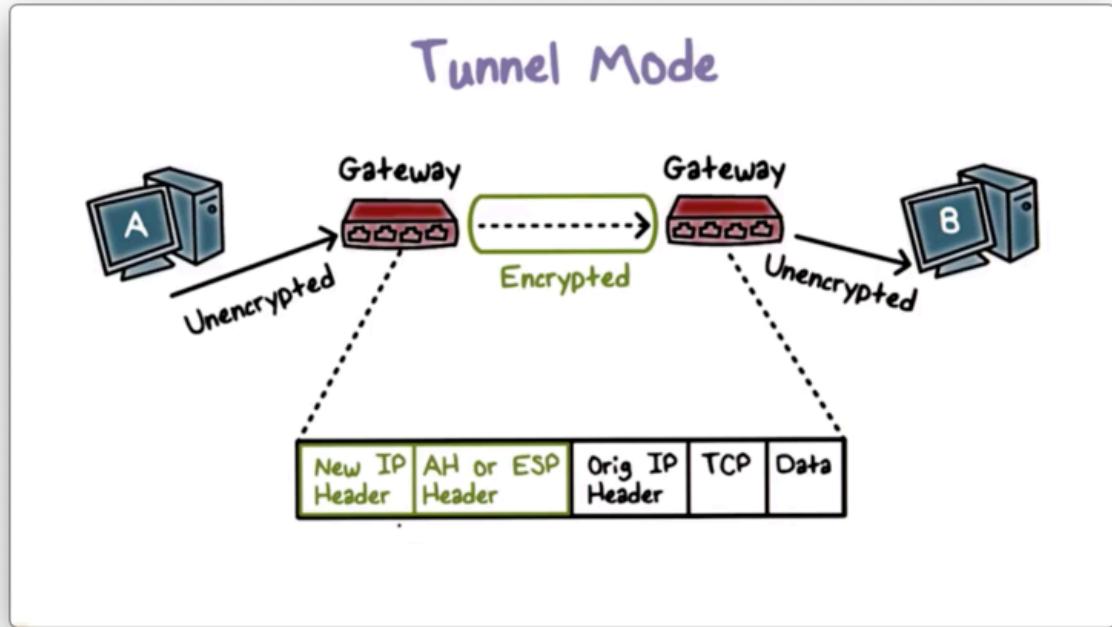
## 17.5 Tunnel Mode

Tunnel mode is the more commonly used operation mode. Suppose we have two end hosts A and B belonging to the same company, but in two different local area networks (LANs) separated by the Internet. If there is an IPSec tunnel between the gateways of the two LANs, then traffic from A to B is automatically protected.



A emits unencrypted packets, and the gateway encrypts them before they leave the LAN. On the receiving side, the gateway to B's LAN decrypts the packets and forwards them to B.

The gateway of A's network encapsulates traffic from A to B by adding a new IP header that specifies its IP as the source IP and the IP of B's gateway as the destination IP to make sure the protected packet is delivered to B's gateway first.



The gateway also includes an IPSec header, which contains information about the protection mechanism used.

The original packet now becomes the data/payload of the new IP packet.

## 17.6 IPSec Quiz



### IPSec Quiz

Fill in the blank with the letter of the correct answer.

IPsec can assure that  .



- A. a router advertisement comes from an authorized router
- B. a routing update is not forged
- C. a redirect message comes from the router to which the initial packet was sent
- D. all of the above

## 17.7 IPSec Quiz Solution

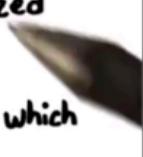


### IPSec Quiz

Fill in the blank with the letter of the correct answer.

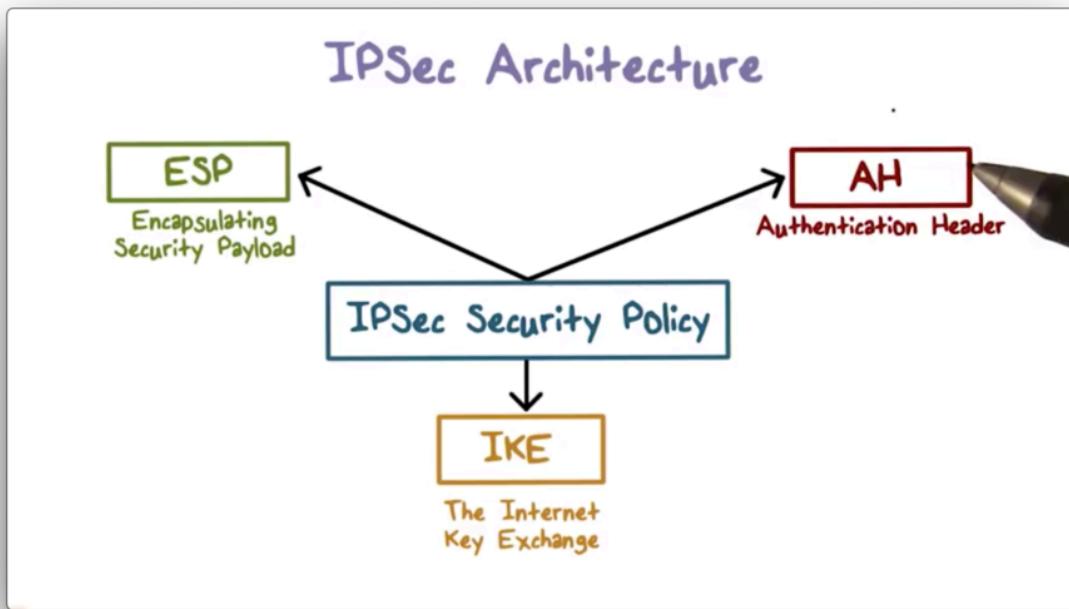
IPsec can assure that  .

A. a router advertisement comes from an authorized router  
B. a routing update is not forged  
C. a redirect message comes from the router to which the initial packet was sent  
D. all of the above



## 17.8 IPSec Architecture

IPSec offers several protocols to perform various functions. These include a key exchange protocol - like the **Internet Key Exchange** (IKE) - used for negotiating protection parameters such as cryptographic algorithms and keys, as well as two types of protection protocols: **Encapsulating Security Payloads** (ESP) and **Authentication Headers** (AH).



### 17.9 Encapsulated Security Payload (ESP)

ESP provides confidentiality protection through IP packet payload encryption. Additionally, ESP provides message authentication to the encrypted payload and IPSec header.

### 17.10 ESP Modes Quiz



#### ESP Modes Quiz

Mark all answer(s) that are correct.

ESP can be securely used in...

- encryption only mode
- authentication only mode
- encryption and authentication mode

### 17.11 ESP Modes Quiz Solution



### ESP Modes Quiz

Mark all answer(s) that are correct.

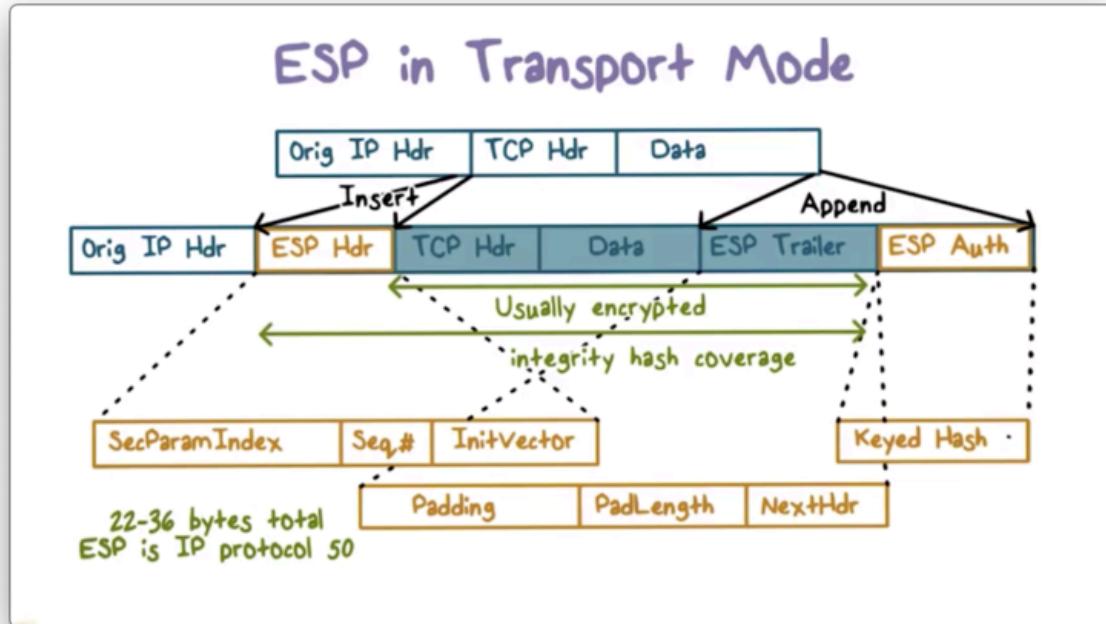
ESP can be securely used in...

- encryption only mode
- authentication only mode
- encryption and authentication mode



### 17.12 ESP in Transport Mode

Here is the new packet layout when IPSec operates in transport mode and uses ESP.



An ESP header is inserted after the original IP header and includes the security parameter index and sequence number, which we will discuss shortly. The ESP header also includes the initialization vector (IV) used for encryption.

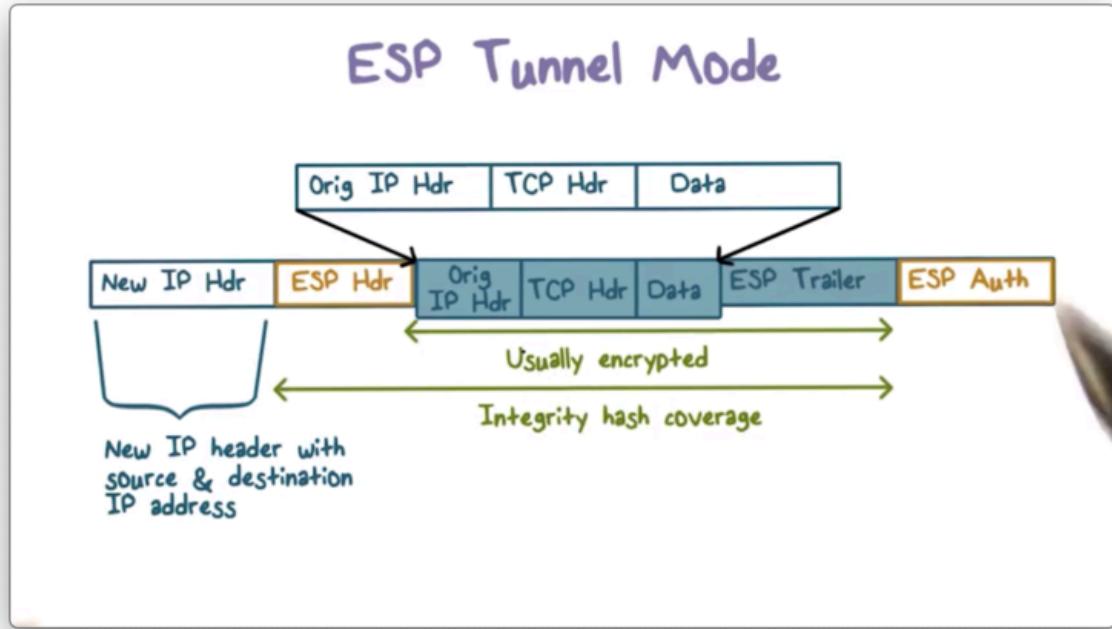
The ESP trailer contains padding information and a next header field, which contains information about the type of data contained in the payload, such as TCP or UDP data.

The packet payload and the ESP trailer are both encrypted, but the ESP header is not because it provides information about how to decrypt the payload, specifically in the security parameter index header. For example, this header can contain information about which algorithm and shared key to use for decryption.

The ESP header and encrypted payload are hashed together with a secret key, and the hash value is appended to the packet. This hash value serves as a message authentication code (MAC) that the receiver can use to verify the authenticity and integrity of the message.

### 17.13 ESP Tunnel Mode

Here is the packet layout when IPSec operates in tunnel mode with ESP.



An ESP header is added after the new IP header, and the packet payload - which contains the entire original packet plus the ESP trailer - is now encrypted. Similarly, the MAC is computed over the entire original packet, plus the ESP header and trailer. Therefore, even the original IP header fields, including the original source and destination IP address, are encrypted and authenticated.

### 17.14 Authentication Header

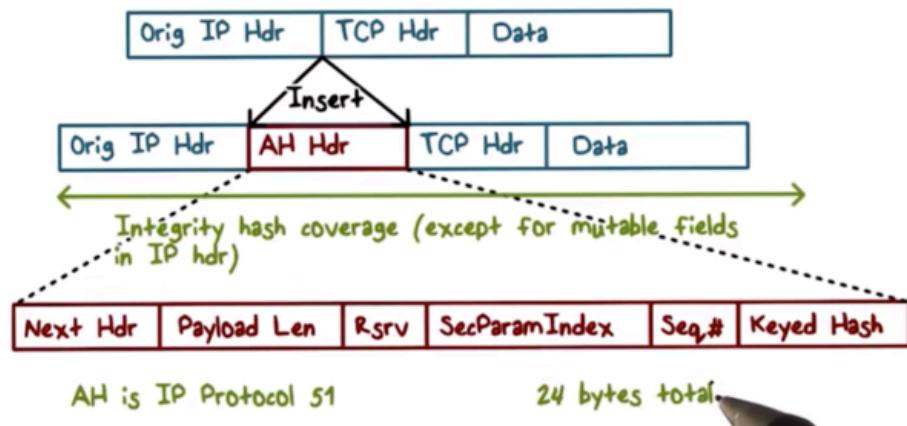
ESP does not authenticate the headers of the transmitted IP packet. If we want to authenticate the entire packet, we can use an authentication header (AH), which contains a MAC for the complete packet.

There are several fields in the IP header - such as the time to live (TTL) field - which may change in transmission. The values of these fields cannot be authenticated, and are often zeroed out when computing the MAC.

While AH does not provide encryption, we can use ESP first to encrypt the payload and then apply AH to authenticate the entire packet.

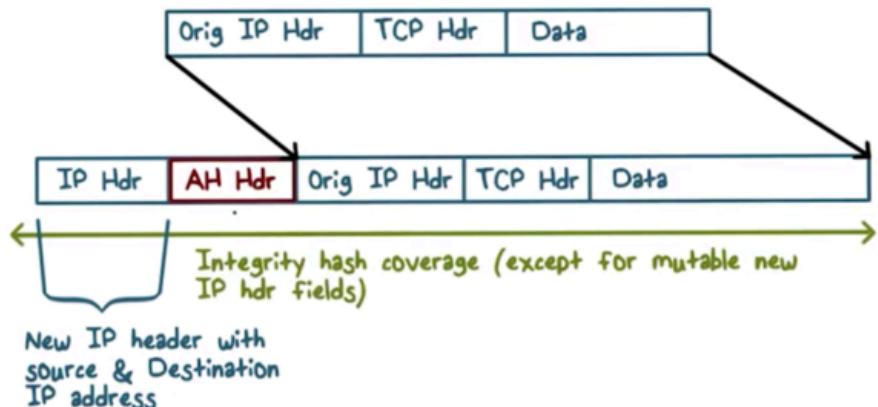
If AH is used in transport mode, the AH header is inserted after the original IP header.

## Authentication Header in Transport Mode



If AH is used in tunnel mode, the AH header is inserted after the new IP header.

## Authentication Header in Tunnel Mode



### 17.15 ESP and AH Quiz



#### ESP and AH Quiz

Label each statement T for True or F  
For False:

- ESP can provide both confidentiality and integrity protection
- If the authentication option of ESP is chosen, message integrity code is computed before encryption
- To protect the confidentiality and integrity of the whole original IP packet, we can use ESP with authentication option in tunnel mode.
- In AH, the integrity hash covers the IP header



### 17.16 ESP and AH Quiz Solution



#### ESP and AH Quiz

Label each statement T for True or F  
For False:

- T    ESP can provide both confidentiality and integrity protection
- F    If the authentication option of ESP is chosen, message integrity code is computed before encryption
- T    To protect the confidentiality and integrity of the whole original IP packet, we can use ESP with authentication option in tunnel mode.
- T    In AH, the integrity hash covers the IP header

### 17.17 Internet Key Exchange

If two parties wish to communicate securely, they typically need to use a security protocol that performs mutual authentication and key exchange.

For two end hosts or gateways to use IPSec for secure communications over the Internet, that protocol is the Internet Key Exchange Protocol (IKE).

IKE allows the two parties to decide the security policies for the traffic between them. Additionally, it allows the parties to agree on a set of security parameters, such as which cryptographic algorithms to use for encryption and hashing. Finally, it allows two parties to establish a shared key for confidential communication.

### 17.18 Security Association

The security parameters for a particular type of traffic - for example, all TCP connections from host A to host B - are described in a **security association** (SA).

Security associations are asymmetric. For the TCP example above, we need one SA to describe traffic flow from A to B and another to describe traffic flow from B to A.

An end host may need many SAs and uses an SA database (SADB) to store them. Each SA has a unique index in the SADB known as the **security parameter index** (SPI).

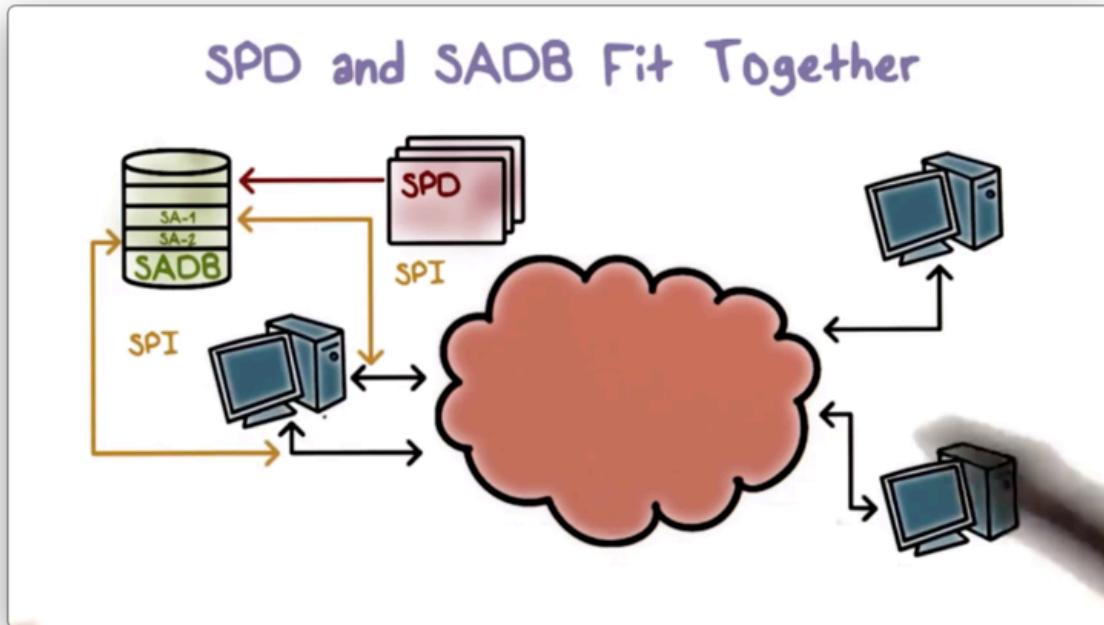
When A and B agree on the security parameters for their communications, each side creates an identical SA entry in their local SADB. Then, B communicates the SPI for its copy to A, which saves it as the SPI for its copy.

Whenever A and B communicate, they include the SPI on any outgoing packet so the receiver can find the corresponding SA and process it according to the agreed-upon security parameters.

The security parameters define the security mechanisms and are determined by the security policies, which are stored in a security policies database (SPDB).

### 17.19 SPD and SADB Fit Together

An SPD entry describes a security policy that decides the security parameters, which are stored in an SA in the SADB. The unique index for the SA of the receiver is the SPI that the sender includes in the IPSec packet header.



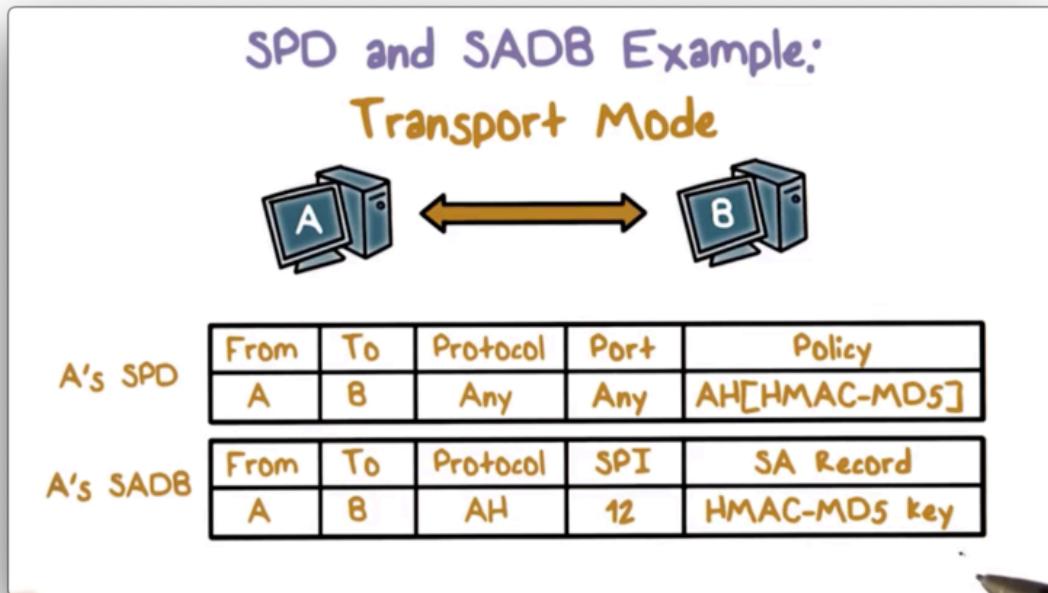
## 17.20 SPD and SADB Example

Recall that transport mode provides end to end traffic protection, while tunnel mode provides traffic protection only between the gateway of the outbound network and the gateway of the inbound network.

### 17.20.1 Transport Mode

Suppose a policy dictates that all traffic from A to B must be authenticated using HMAC with MD5 as the embedded hash function.

A's SPD stores this policy, and both A and B store an SA in their SADB containing the negotiated parameters for the policy. For example, A's SA stores the secret key for HMAC and the SPI to index the SA in B's SADB.

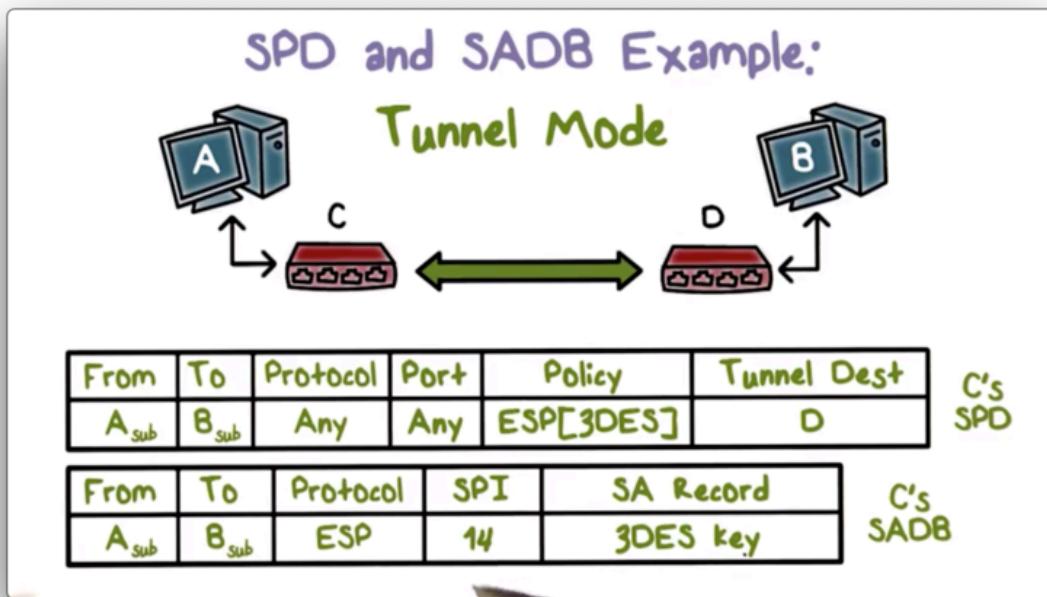


When A sends traffic to B, it includes the SPI in the IPSec header so that B can use it to look up its SA and then process the traffic appropriately.

### 17.20.2 Tunnel Mode

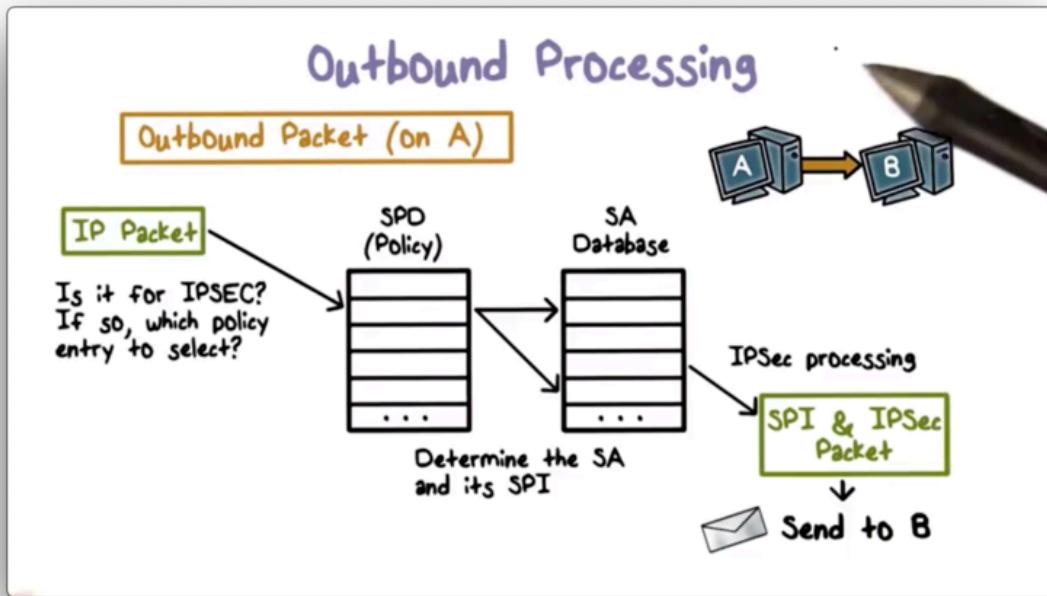
Suppose a policy states that any traffic from A's subnet to B's subnet must be sent to B's gateway D, and must be processed using ESP with 3DES.

Since C is the gateway of the A's subnet, C's SPD stores this policy, and its SADB stores the SA that has the 3DES key and the SPI for looking up the SA in D's SADB.



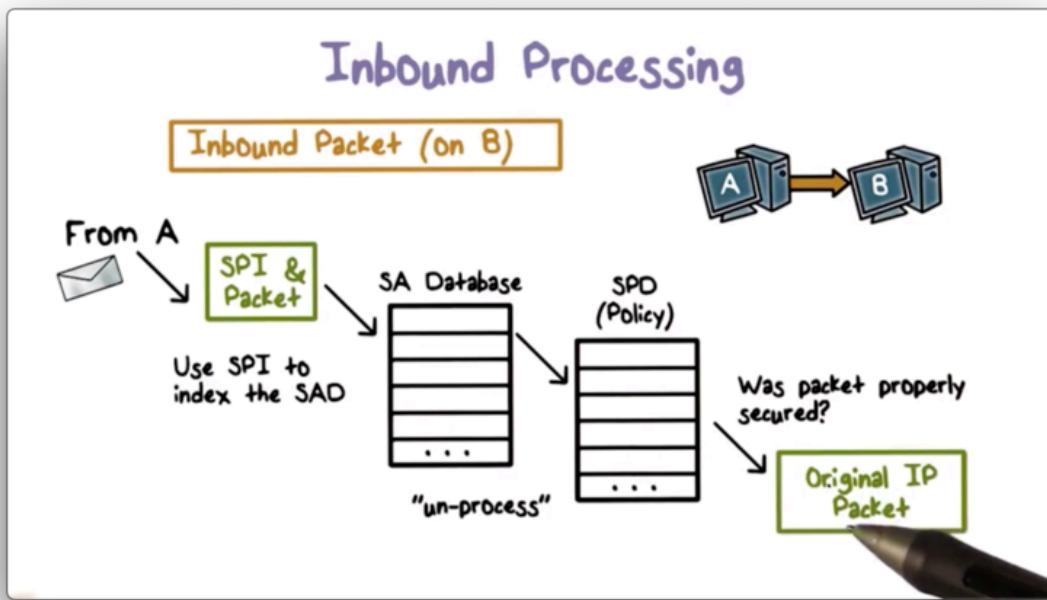
### 17.21 Outbound Processing

Before A sends a packet, it first examines the security policies in the SPD to see if the traffic needs IPsec protection. If an applicable SPD entry exists, then A retrieves the corresponding SA from the SADB and processes the packet accordingly.



## 17.22 Inbound Processing

When **B** receives the packet, it uses the SPI in the IPSec header to look up the SA in the SADB and processes the packet accordingly. Next, **B** looks up the SPD to ensure that **A** secured the packet correctly according to the policy. If so, **B** delivers the packet to the upper layer application.



### 17.23 Anti Replay

The IPSec header contains a **sequence number** field, which is designed to prevent replay attacks. This field is only used if AH is requested, or if the authentication option in ESP is selected.

To utilize sequence numbers, a host  $H$  must maintain a sliding window of size  $n$  - which should be at least 32. Although packets may arrive out of order, their sequence numbers should be within the window of size  $n$ .

Suppose  $H$  maintains a window where  $n = 50$ , which contains the sequence numbers from 100 to 149.

If a packet arrives with a sequence number less than 100,  $H$  rejects the packet. If a packet arrives with a sequence number greater than 149,  $H$  accepts the packet and adjusts the window to cover this packet's sequence number. For example, if  $H$  receives a packet with sequence number 199,  $H$  adjusts its window to cover 150 through 199.

Suppose  $H$  maintains a window where  $n = 50$ , which contains the sequence numbers from 100 to 149.

If a packet arrives with a sequence number between 100 and 149,  $H$  checks the number to see if it has already been seen. If yes,  $H$  rejects the packet; otherwise,  $H$  accepts the packet and records the

sequence number as having been seen.

#### 17.24 IPSec Quiz



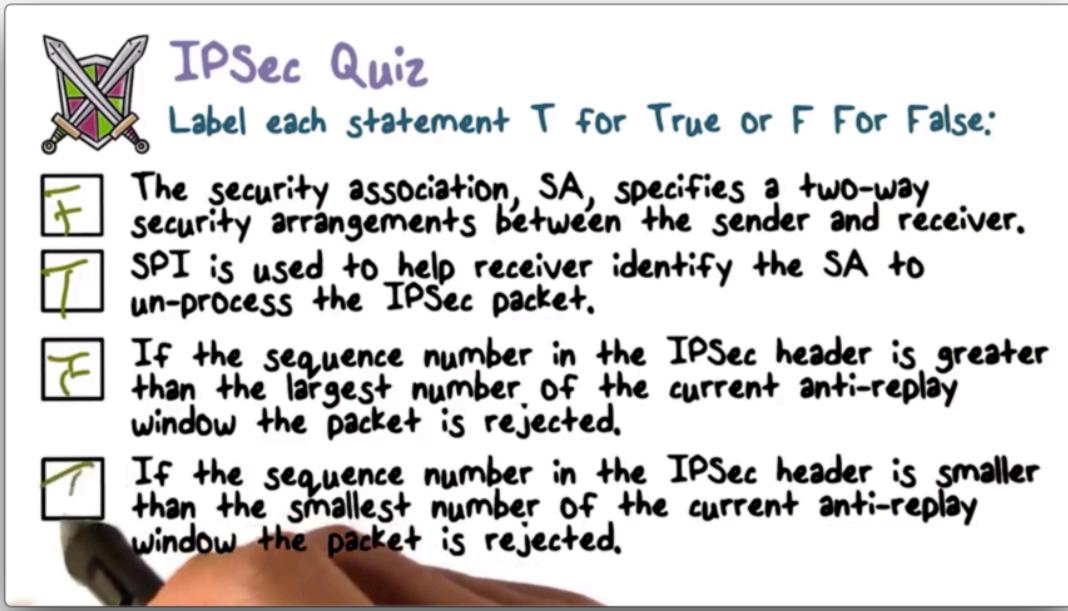
#### IPSec Quiz

Label each statement T for True or F For False:

- The security association, SA, specifies a two-way security arrangements between the sender and receiver.
- SPI is used to help receiver identify the SA to un-process the IPSec packet.
- If the sequence number in the IPSec header is greater than the largest number of the current anti-replay window the packet is rejected.
- If the sequence number in the IPSec header is smaller than the smallest number of the current anti-replay window the packet is rejected.



### 17.25 IPSec Quiz Solution



**IPSec Quiz**  
Label each statement T for True or F For False:

- F** The security association, SA, specifies a two-way security arrangements between the sender and receiver.
- T** SPI is used to help receiver identify the SA to un-process the IPSec packet.
- F** If the sequence number in the IPSec header is greater than the largest number of the current anti-replay window the packet is rejected.
- T** If the sequence number in the IPSec header is smaller than the smallest number of the current anti-replay window the packet is rejected.

### 17.26 Internet Key Exchange

Before A and B can communicate securely, they must agree upon the security parameters for their communication, such as the encryption and authentication algorithms and keys that they plan to use. In other words, they need to establish an SA before any protected communication can begin.

The Internet Key Exchange (IKE) protocol facilitates the establishment of one or more IPSec SAs between hosts. This protocol works in two phases. First, the hosts establish an IKE SA to protect the SA negotiation itself. The IKE SA is bi-directional; that is, it protects the SA negotiation traffic from both sides. Next, the hosts use this SA to protect the negotiations of multiple IPSec SAs.

### 17.27 IKE Phase I

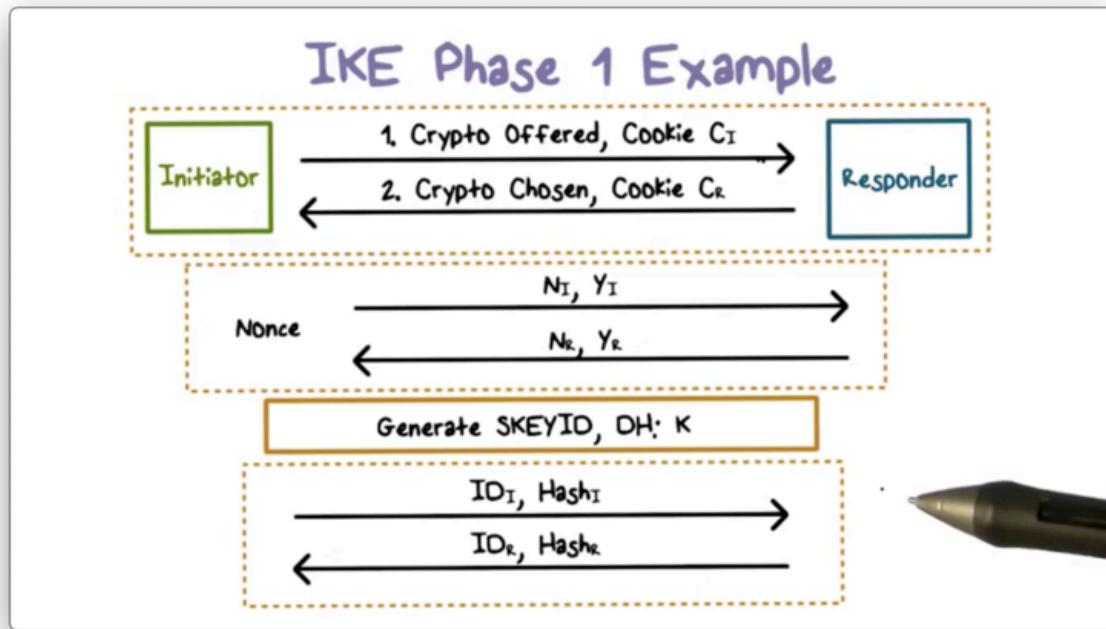
The first phase of the IKE protocol serves to establish a general security association that can be used to establish multiple IPSec security associations in the second phase.

First, both sides negotiate the protection mechanism to use - AH or ESP - and also agree upon which cryptographic algorithms to use, such as AES or HMAC with SHA-1.

Then, they establish a shared secret key using a protocol such as Diffie-Hellman. Both sides can use either a pre-shared key, digital signatures, or public-key encryption to authenticate the key exchange.

### 17.28 IKE Phase I Example

Let's look at an example of phase one of the IKE protocol. For this example, we assume that both sides have a pre-shared secret key.



First, the initiator sends the cryptographic algorithm it proposes, along with a cookie, to the responder. The cookie is a value that the initiator can easily compute, and the responder can easily verify, such as a hash over the initiator's IP address and the current timestamp together.

The cookie proves that the initiator has done some computation and is serious about following through with the protocol. In general, cookies help to mitigate denial of service attacks where an initiator can send many requests to a responder at little to no cost.

After verifying the cookie, the responder sends back its choice of cryptographic algorithm and its own cookie to the initiator.

Second, the two parties exchange  $Y_I$  and  $Y_R$  - the public components of the Diffie-Hellman key exchange - combined with nonce values  $N_I$  and  $N_R$  to prevent replay attacks.

Third, both the initiator and the responder compute the same shared key according to the Diffie-Hellman key exchange as well as other keys necessary for the IKE SA.

Finally, they exchange hash values to authenticate the newly established key using their pre-shared secret key.

### 17.29 Diffie Hellman Quiz



**Diffie-Hellman Quiz .**

The Diffie-Hellman key exchange is restricted to two party communication only.

Is the above statement true or false?



### 17.30 Diffie Hellman Quiz Solution



### Diffie-Hellman Quiz

The Diffie-Hellman key exchange is restricted to two party communication only.

Is the above statement true or false?

False



### 17.31 Diffie-Hellman and Pre-shared Secret

In our example, the initiator and the responder have a pre-shared secret key, and, using this key as well as the information exchanged between them, they can both compute shared keys using a pseudorandom function.

Each party can build a **pseudorandom function** using HMAC and SHA-1 to generate a pseudorandom bitstream. SHA-1 has a property whereby the change in a single bit of input produces a new hash value with no apparent connection to the preceding hash value. This property forms the basis for *pseudorandom number generation*.

The pseudorandom function **PRF** receives two pieces of data as input - a key and a data block - both of which are passed down to HMAC.

First, the initiator and the responder compute a root shared secret, **SKEYID**. Each party computes this value by invoking **PRF**, using the pre-shared key as the key and the previously exchanged nonce values **N<sub>i</sub>** and **N<sub>r</sub>** as the data block.

Next, they compute a shared key to use for IPSec SA, **SKEYID\_d**. Each party computes this value by invoking **PRF**, using **SKEYID** as the key, and the following four values as the data block: **K**, the shared

secret key computed using Diffie-Hellman;  $C_i$  and  $C_r$ , the cookies previously exchanged between the initiator and the responder, and; the number 0.

Both parties compute the keys for IKE message authentication and encryption in a similar fashion.

### Diffie-Hellman and Pre-shared Secret

- PRF is a Pseudo-Random Function
- SKEYID root secret  
 $= \text{PRF}(\text{preshared-key}, N_I | N_R)$
- SKEYID\_d for IPsec SA  
 $= \text{PRF}(SKEYID, K | C_I | C_R | 0)$   
K is the shared secret key computed using Diffie-Hellman
- SKEYID\_a for IKE message data authentication & integrity  
 $= \text{PRF}(SKEYID, SKEYID_d | K | C_I | C_R | 1)$
- SKEYID\_e use to encrypt IKE messages  
 $= \text{PRF}(SKEYID, SKEYID_a | K | C_I | C_R | 2)$



### 17.32 Authentication of the Key Exchange

Now let's take a look at how the initiator and responder authenticate the key exchange. Both parties hash the information they have exchanged, using PRF with SKEYID as the key.

The input data block passed to PRF contains the public components of the Diffie-Hellman exchange, the cookies, the offered cryptographic algorithms, and the identities of the initiator and responder.

## Authentication of the Key Exchange

- To authenticate each other's identity and data that they just exchanged, the initiator and responder each generates a hash digest that only the other could know

$\text{Hash-I} = \text{PRF}(\text{SKEYID}, Y_I | Y_R | C_I | C_R | \text{Crypto Offer} | ID_I)$

$\text{Hash-R} = \text{PRF}(\text{SKEYID}, Y_R | Y_I | C_I | C_R | \text{Crypto Offer} | ID_R)$



Each party can verify the hash value computed by the other because these values are based on information shared by both parties. By verifying these values, each party can authenticate the exchange.

### 17.33 IKE Phase II Keys

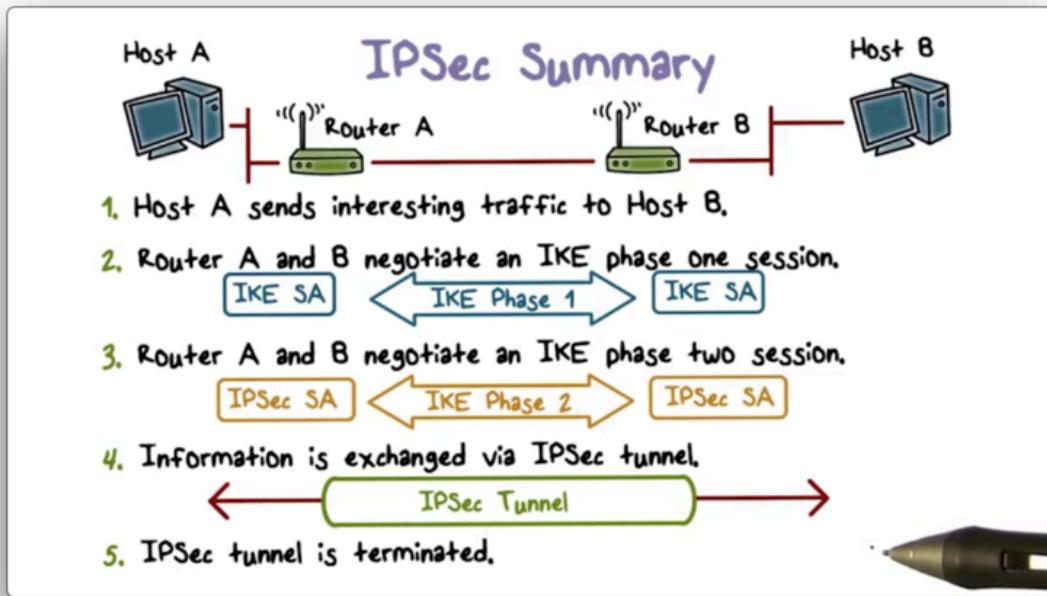
The second phase of IKE deals with establishing IPSec SAs. Multiple SAs can be negotiated using the protection of the same IKE SA established in the first phase of IKE. The two phases work similarly, differing primarily in how the IPSec keys are derived.

If the hosts do not require [perfect forward secrecy](#) (PFS), then they can derive all SA keys using one of the shared keys, [SKEYID\\_d](#), computed in IKE phase one. The weakness of this approach is that if [SKEYID\\_d](#) is somehow leaked, then all of the IPSec SA keys are compromised.

Stronger security requires PFS. With PFS, both sides exchange new nonce values and perform new key exchanges before each IPSec SA negotiation. Therefore, unless the pre-shared master secret key is compromised, the keys for the current IPSec SA are secure, even if other keys previously computed have been compromised.

### 17.34 IPSec Summary

To summarize, if host A and host B want to communicate, the typical IPSec workflow is as follows.



Suppose this is the first time that A sends data to B that, according to policy, requires protection. The gateway of A's network and the gateway of B's network first use the IKE protocol to negotiate the IKE SA and then use that IKE SA to negotiate the IPSec SAs.

Then, the routers can use the SAs to create an IPSec tunnel between them, which protects the traffic from A to B. For example, the packet data can be encrypted and, optionally, the header information and packet data can be authenticated, depending on the SAs used.

When A terminates the connection to B, the IPSec tunnel between the two routers also terminates.

### 17.35 IKE Quiz



#### IKE Quiz

Label each statement T for True or F  
For False:

- An IKE SA needs to be established before IPSec SAs can be negotiated
- The identity of the responder and receiver and the messages they have exchanged need to be authenticated
- With perfect forward secrecy, the IPSec SA keys are based on the IKE shared secret established in Phase I.

### 17.36 IKE Quiz Solution



#### IKE Quiz

Label each statement T for True or F  
For False:

- T An IKE SA needs to be established before IPSec SAs can be negotiated
- T The identity of the responder and receiver and the messages they have exchanged need to be authenticated
- F With perfect forward secrecy, the IPSec SA keys are based on the IKE shared secret established in Phase I.

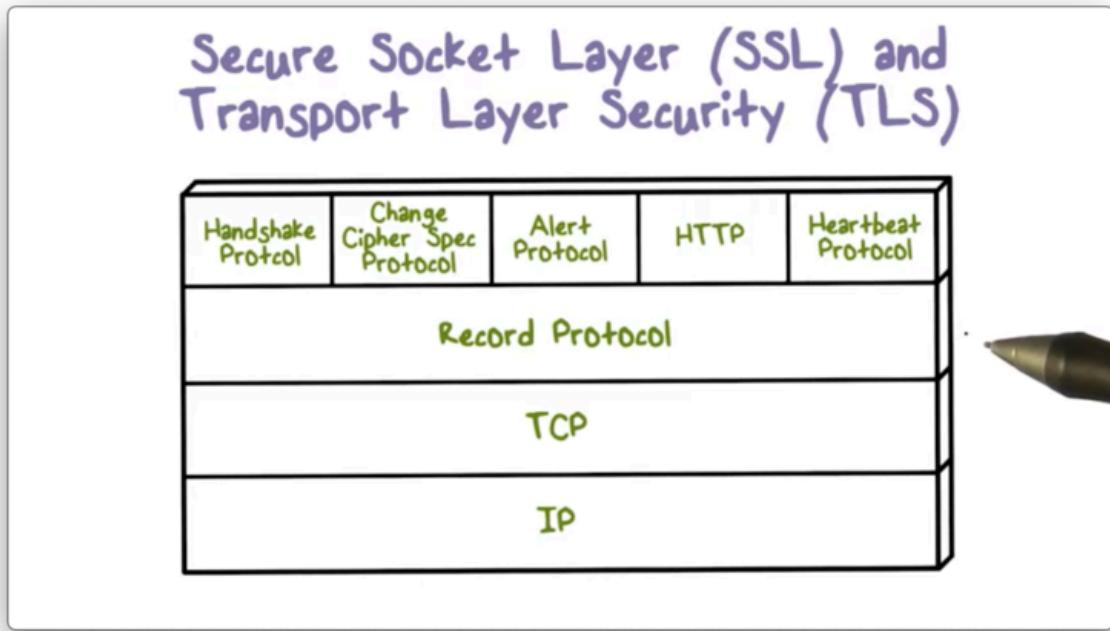
### 17.37 SSL and TLS

One of the most widely used security services is **Secure Sockets Layer** (SSL) and the follow-on standard, **Transport Layer Security** (TLS). TLS is designed to make use of TCP to provide a reliable end-to-end secure service.

TLS can be provided as part of the underlying transport protocol suite, which allows all applications above the transport layer to benefit from the provided security services.

Alternatively, TLS can be embedded in specific application packages. For example, most web browsers come equipped with SSL, and most web servers have implemented the protocol.

TLS is not a single protocol, but rather two layers of protocols, illustrated by the following figure.



The *record protocol* provides basic security services to various higher-layer protocols, such as HTTP or SMTP.

Additionally, TLS defines three application-level protocols - the *handshake protocol*, the *change cipher spec protocol*, and the *alert protocol* - which are used in the management of TLS exchanges.

### 17.38 TLS Concepts

A **TLS session** is an association between a client and a server created by the handshake protocol. A session defines the set of cryptographic parameters to be used by each connection within the session. By defining the parameters at the session level, we avoid having to perform the expensive security negotiation process for each new connection.

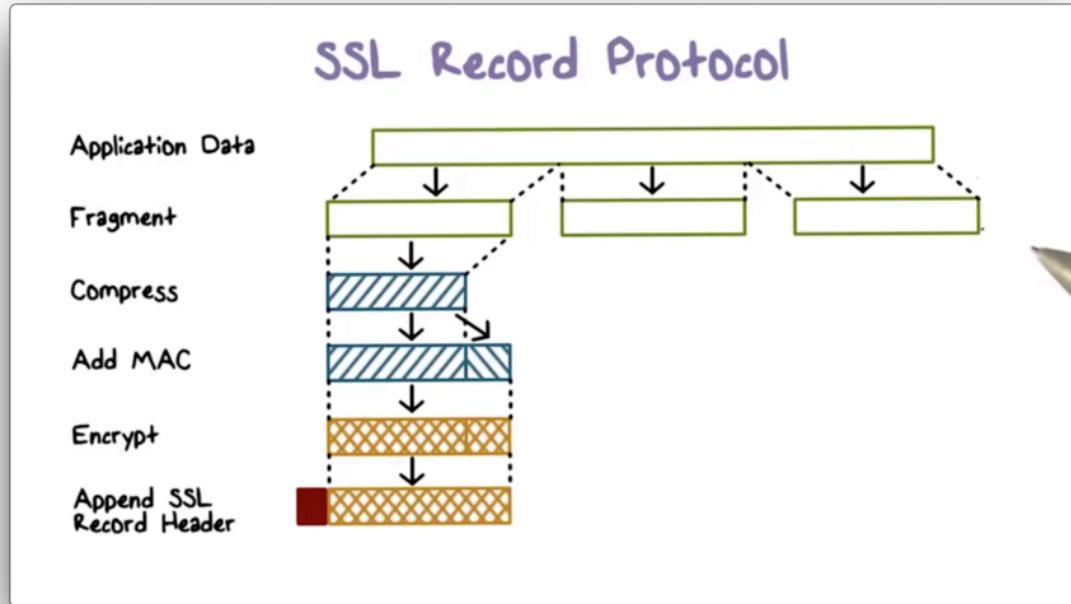
A **TLS connection** is a transport layer relationship between a client and a server. Whereas a TLS session defines a long-term relationship, a TLS connection is transient; for example, if the client closes the connection, the connection terminates, even though the session may remain intact.

### 17.39 SSL Record Protocol

The SSL record protocol provides two services for SSL connections - confidentiality and message integrity - both of which are made possible by the handshake protocol. That is, the handshake protocol

establishes secret keys for use in encryption and message authentication code (MAC) generation, and the record protocol uses these keys to provide those services.

This figure shows the overall transformation of application data using the SSL record protocol.



First, the host  $H$  fragments the application data into blocks that can fit in a TCP segment. Next,  $H$  compresses each block and computes a MAC over the compressed data. Third,  $H$  encrypts the compressed message and MAC using symmetric encryption. Finally,  $H$  prepends a header to the encrypted, authenticated message that includes fields specifying message length and protocol version.

$H$  transmits the data in a TCP segment. The receiving end applies the transformation steps in reverse - decryption, verification, decompression, and reassembly - before delivering the data to the application.

#### 17.40 The Handshake Protocol

The handshake protocol allows a client and server to negotiate security parameters, ultimately resulting in the creation of a TLS session between them.

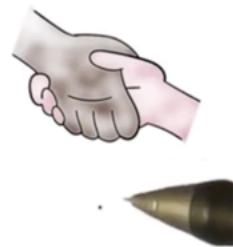
### 17.40.1 Phase One

In phase one, the client and server share their respective security capabilities. The client initiates this phase by sending a `client_hello` message to the server, which contains several parameters: TLS version number, session ID, crypto suite, compression method, and initial random numbers.

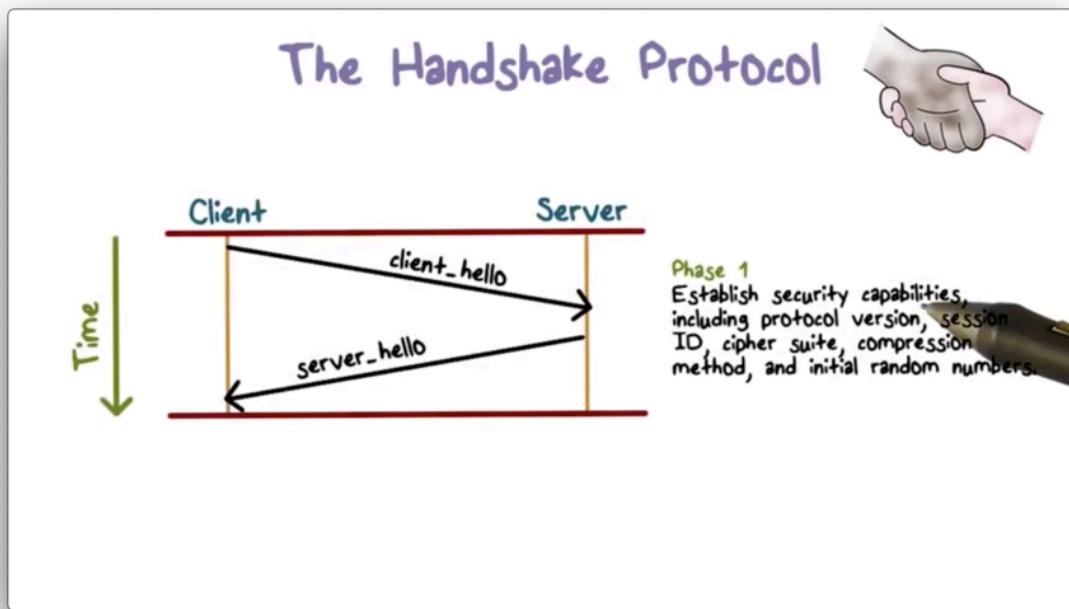
## The Handshake Protocol

### The Parameters:

- **Version:** the highest TLS version understood by the client
- **Random:** a 32-bit timestamp and 28 bytes generated by a secure random number generator
- **Session ID:** a variable-length session identifier
- **CipherSuite:** a list containing the combinations of cryptographic algorithms supported by the client
- **Compression Method:** a list of compression methods supported by the client

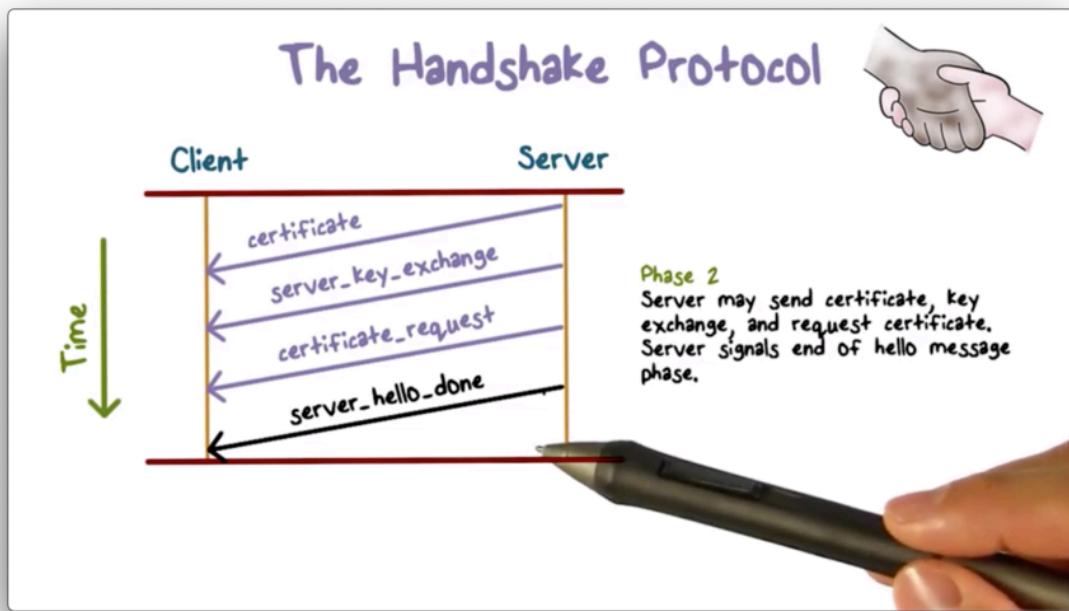


After sending the `client_hello` message, the client waits for the `server_hello` message, which contains the same parameters. Therefore, at the end of phase one, both client and server know each other's security capabilities.



### 17.40.2 Phase Two

The details of phase two depend on the underlying public-key encryption scheme in use. In some cases, the server passes a certificate to the client, possibly with some additional key information, and may request a certificate from the client. Regardless, the final message must be `server_hello_done`, which indicates the end of this phase.

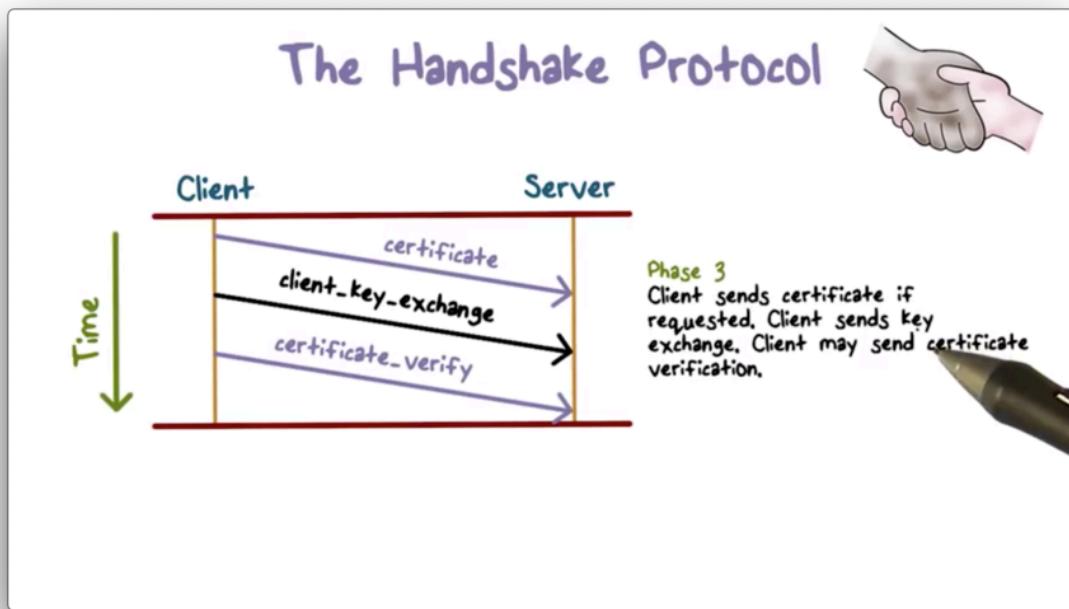


#### 17.40.3 Phase Three

In phase three, the client should first verify the server's certificate. For example, the client should be able to validate that a reputable certificate authority signed the certificate.

After verification, the client can send key exchange information to the server. For example, the client can generate a secret key, encrypt it using the server's public key, and send it to the server.

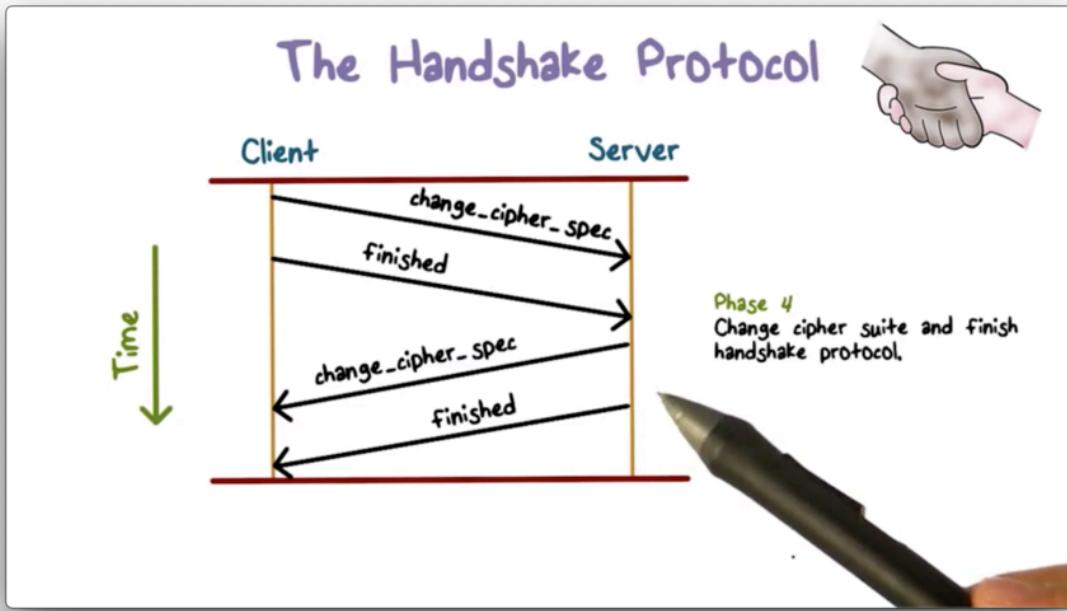
Depending on the application requirements, the client may send a certificate to the server to authenticate itself. If a website is public-facing, then the authentication is typically one-way; that is, the client needs to authenticate the server, but the server need not authenticate the client. On the other hand, internal or private web servers may require mutual authentication.



#### 17.40.4 Phase Four

In phase four, the client sends a `change_cipher_spec` message to the server and copies the pending security parameters to the current cipher spec. It then signals the completion of the handshake protocol.

In response, the server sends its own `change_cipher_spec` back to the client to signal that it also agrees on the exchanged security parameters. Then the server sends its own message to signal the end of the handshake.



At this point, the handshake is complete, and the client and server can begin to exchange application layer data, which will be protected using the negotiated security parameters.

### 17.41 TLS and SSL Quiz



#### TLS and SSL Quiz

Label each statement T for True or F  
For False:

- Most browsers come equipped with SSL and most Web servers have implemented the protocol
- Since TLS is for the Transport layer, it relies on IPSec, which is for the IP layer
- In most applications of TLS or SSL, public keys are used for authentication and key exchange



### 17.42 TLS and SSL Quiz Solution



### TLS and SSL Quiz

Label each statement T for True or F  
For False:

- Most browsers come equipped with SSL and most Web servers have implemented the protocol
- Since TLS is for the Transport layer, it relies on IPSec, which is for the IP layer
- In most applications of TLS or SSL, public keys are used for authentication and key exchange

While transport layer protocols do rely on the IP layer, TLS does not specifically rely on IPSec.

## 18 Wireless and Mobile Security

### 18.1 Introduction to Wifi

A typical use of WiFi is to allow WiFi-enabled personal computers or devices to access the Internet through a wireless access point (AP). Alternatively, devices can connect directly to one another wirelessly through the AP.

Devices connect to the AP wirelessly, and the AP connects to the Internet through physical wiring, often through a router provided by the Internet service provider.

In wireless networking, data is not transmitted via physical wiring, but rather through air, which is an open medium. In other words, there is no inherent physical protection in wireless communications.

Without hard wiring connecting two devices for direct communication, devices in a wireless environment must use broadcasting; that is, a sender must broadcast a message, and a receiver must be listening for a broadcast.

## 18.2 Wifi Quiz



### WiFi Quiz

Select all that apply.

Which of the following are security threats to WiFi:

- Eavesdropping
- Injecting bogus messages
- Replaying previously recorded messages
- Illegitimate access to the network & its services
- Denial-of-service
- All the above

### 18.3 Wifi Quiz Solution



### WiFi Quiz

Select all that apply.

Which of the following are security threats to WiFi:

- Eavesdropping
- Injecting bogus messages
- Replaying previously recorded messages
- Illegitimate access to the network & its services
- Denial-of-service
- All the above

### 18.4 Overview of Wifi Security

The earlier WiFi security standard, Wired Equivalent Privacy (WEP), is easily breakable even when properly configured. The new, more secure standard is 802.11i, which WiFi Protected Access 2 (WPA2) implements. You should always use WPA2 over WEP.

### 18.5 Overview of 802.11i

The 802.11i standard enforces access control, and the underlying access control protocol is based on another standard, 802.1x. 802.1x is flexible because it is based on the [Extensible Authentication Protocol](#) (EAP).

EAP is designed as a carrier protocol whose purpose is to transport the messages of “real” authentication protocols, such as TLS. In other words, you can implement a host of different authentication methods on top of EAP, and therefore on top of 802.1x.

The more advanced EAP methods, such as TLS, provide mutual authentication, which limits man-in-the-middle attacks by authenticating both the server and client. Furthermore, this EAP method results in key material, which can be used to generate dynamic encryption keys.

Additionally, 802.11i follows strong security practices. For example, it uses different keys for encryption and integrity protection, and also uses more secure encryption schemes - AES in particular.

### 18.6 Wifi Security Standards Quiz



### WiFi Security Standards Quiz

Choose the best answer:

Which security standard should be used for WiFi?

WEP

WPA2



### 18.7 Wifi Security Standards Quiz Solution



### WiFi Security Standards Quiz

Choose the best answer:

Which security standard should be used for WiFi?

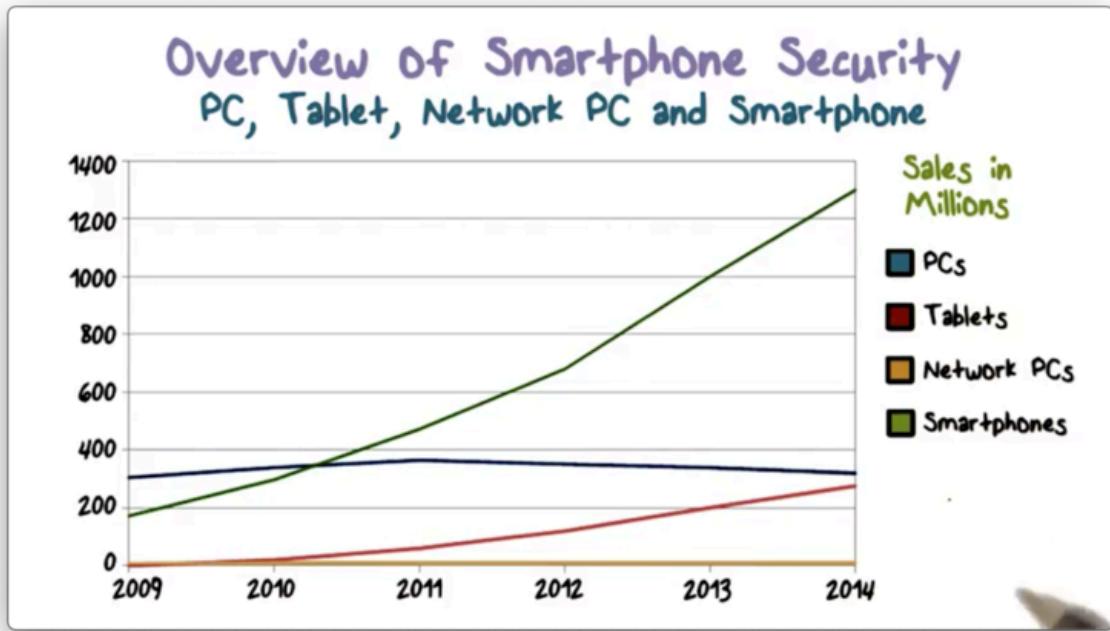
WEP

WPA2



### 18.8 Overview of Smartphone Security

The following plot shows a significant increase in smartphone sales in recent years.



People use smartphones now more than ever, and we are using them for more and more essential tasks. Therefore, we must examine the security of smartphones.

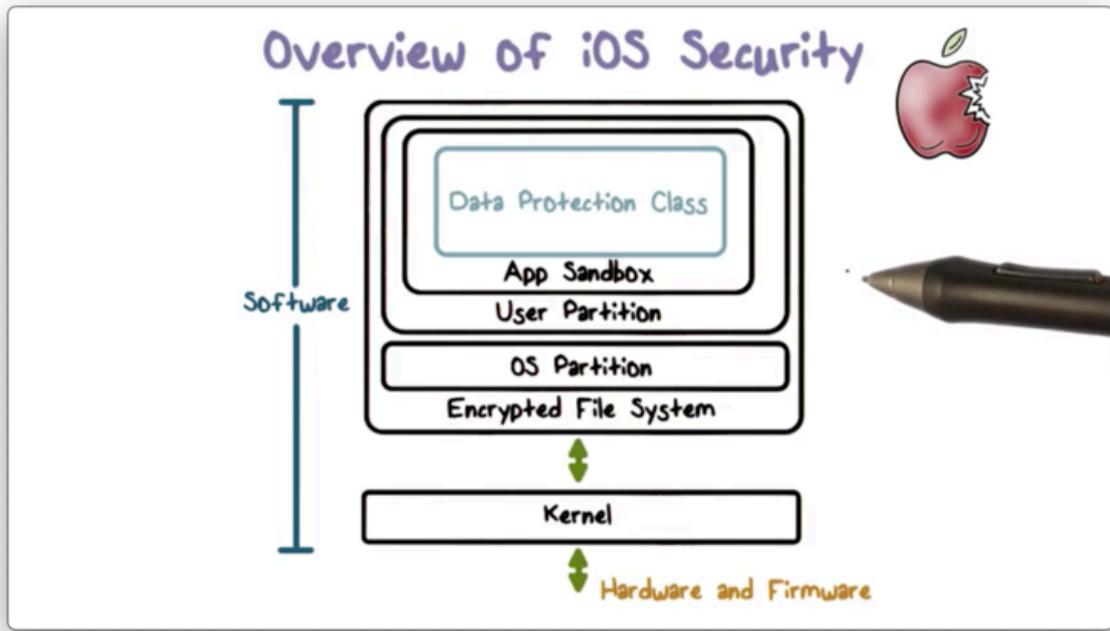
### 18.9 Overview of iOS Security

The iOS security architecture combines both hardware and software features to provide security to iOS devices such as iPhones and iPads.



The architecture contains built-in cryptographic capabilities - for example, the cryptographic engine and keys are embedded into the hardware - for supporting data protection via confidentiality and integrity.

The architecture also provides powerful isolation mechanisms. For example, it uses app sandboxing to protect app security. These sandboxes enable apps to run in isolation, free from interference from other apps. Additionally, sandboxing helps to ensure the integrity of the overall system. In other words, even if an app is compromised, its capability to damage the system is minimal.



### 18.10 Operating System Vulnerabilities Quiz

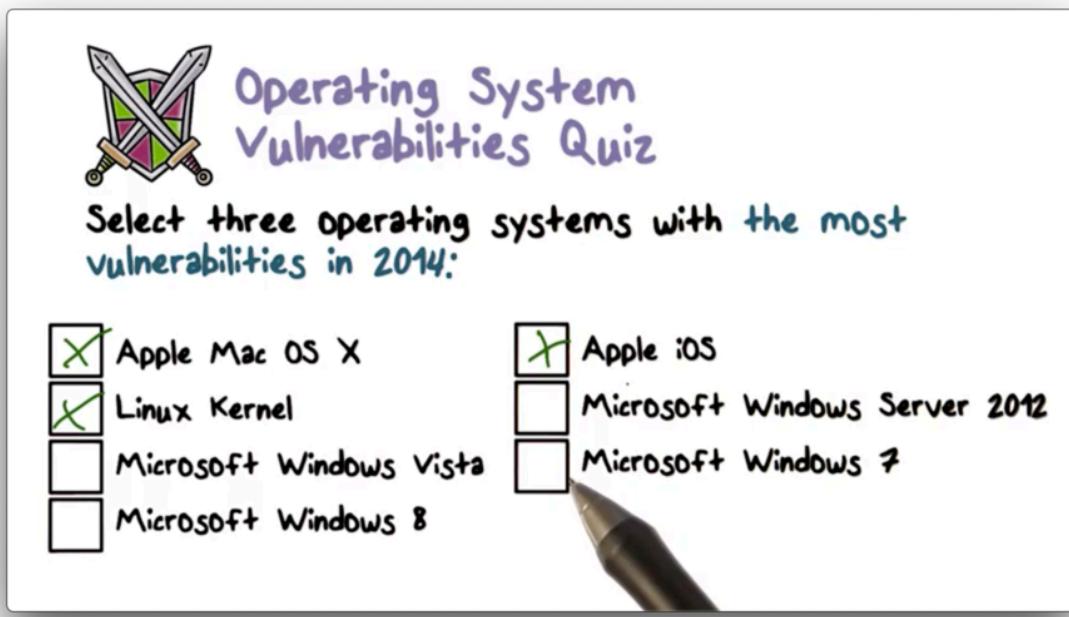
**Operating System Vulnerabilities Quiz**

Select three operating systems with the most vulnerabilities in 2014:

<input type="checkbox"/> Apple Mac OS X	<input type="checkbox"/> Apple iOS
<input type="checkbox"/> Linux Kernel	<input type="checkbox"/> Microsoft Windows Server 2012
<input type="checkbox"/> Microsoft Windows Vista	<input type="checkbox"/> Microsoft Windows 7
<input type="checkbox"/> Microsoft Windows 8	

In the top right corner of the slide, there is a drawing of a pen.

### 18.11 Operating System Vulnerabilities Quiz Solution



A graphic titled "Operating System Vulnerabilities Quiz". It features a logo of two crossed swords on the left. The text "Operating System Vulnerabilities Quiz" is written in purple. Below it, a hand-drawn style text says "Select three operating systems with the most vulnerabilities in 2014:". There is a list of five operating systems with checkboxes:

<input checked="" type="checkbox"/>	Apple Mac OS X
<input checked="" type="checkbox"/>	Linux Kernel
<input type="checkbox"/>	Microsoft Windows Vista
<input type="checkbox"/>	Microsoft Windows 8
<input type="checkbox"/>	Apple iOS
<input type="checkbox"/>	Microsoft Windows Server 2012
<input type="checkbox"/>	Microsoft Windows 7

A hand-drawn style pen is shown pointing towards the bottom right of the list.

Betcha thought it was gonna be all Microsoft, didn't you? Read more [here](#).

### 18.12 Hardware Security Feature

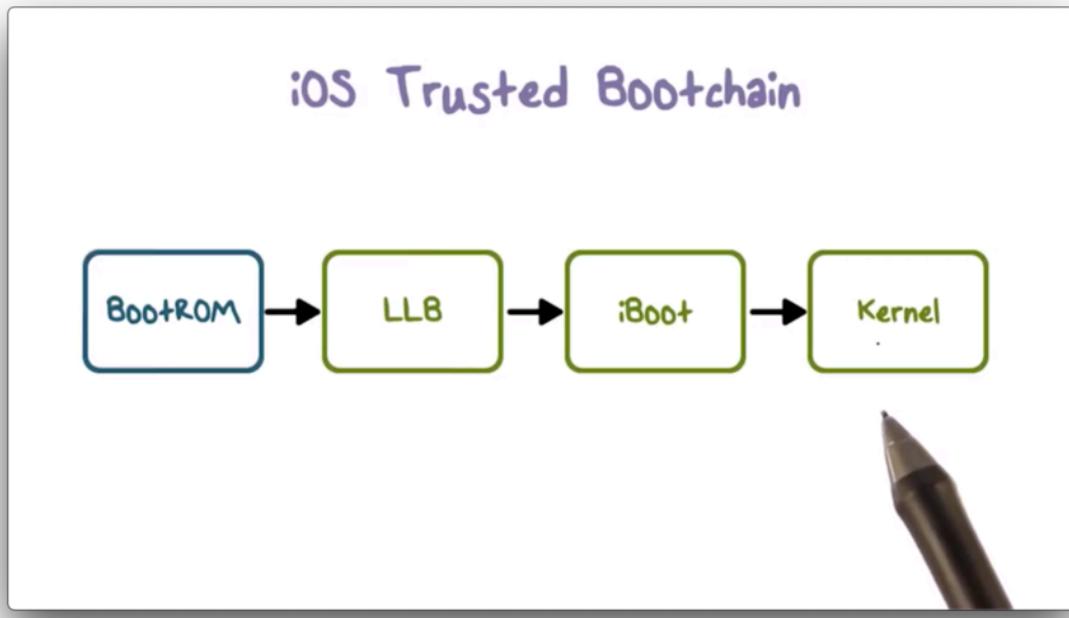
Each iOS device has a dedicated AES-256 cryptographic engine built into the direct memory access path between the flash storage and the main system memory, which makes file encryption/decryption highly efficient.

The device's unique ID (UID) and group ID (GID) are AES 256-bit keys fused into the [secure enclave](#) hardware component during manufacturing. Only the cryptographic engine, itself a hardware component, can read these keys directly. All other firmware and software components can only see the result of an encryption or decryption operation.

A UID is unique to a device and is not recorded by Apple or its suppliers. GIDs are common to all processors in a class of devices, such as those using the Apple A8 processor, and are used for tasks such as delivering system installations and updates.

### 18.13 iOS Trusted Bootchain

iOS uses a **trusted bootchain** to establish the security of an iOS device on boot. Each step in the bootchain (except the first) only executes once the previous step has verified it.



When an iOS device is turned on, each application processor immediately executes code from a section of read-only memory known as the *BootROM*. This immutable, implicitly-trusted code, known as the hardware *root of trust*, is burned into the hardware during chip fabrication.

The BootROM code contains the Apple root CA public key, which is used to verify that Apple has signed the *low-level boot loader* (LLB) before allowing it to load. When the LLB finishes its tasks, it verifies and runs the next stage boot loader, iBoot, which in turn verifies and runs the iOS kernel.

This secure bootchain helps ensure that the lowest levels of software are not tampered with, and enforces that iOS only runs on validated Apple devices.

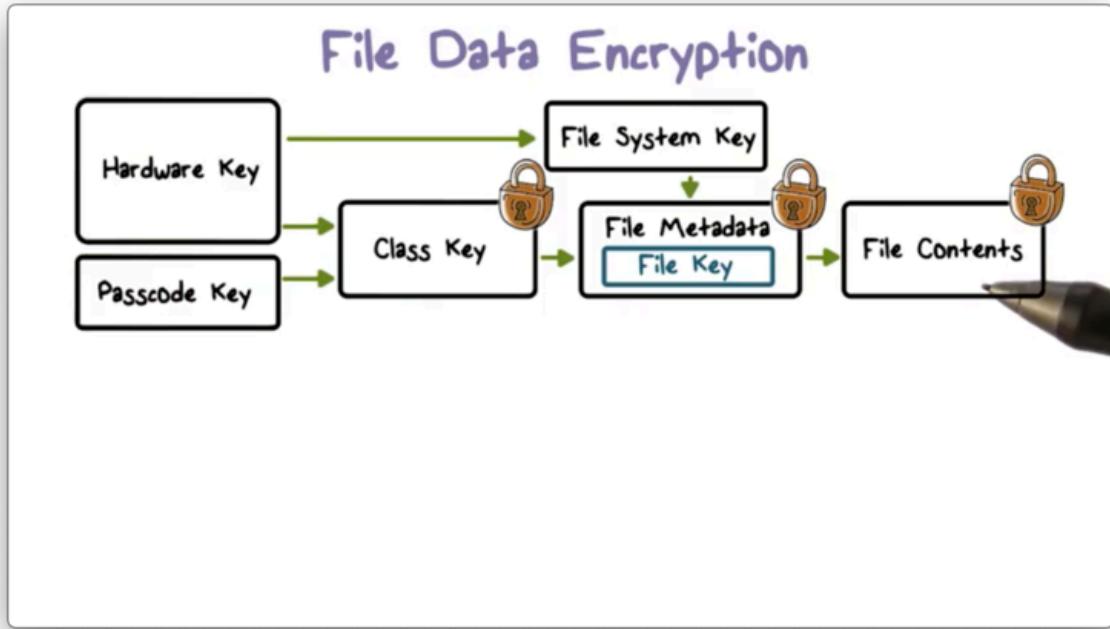
### 18.14 File Data Encryption

In addition to the cryptographic capabilities built into the hardware of each iOS device, Apple uses a technology called **data protection** to further secure data stored in flash memory.

Data protection enables a high level of encryption for user data. Critical system apps such as Messages, Mail, and Calendar use data protection by default, and third-party apps installed on iOS 7 or later

receive this protection automatically.

Data protection constructs and manages a hierarchy of keys - such as class, file, and filesystem keys - that builds on the hardware encryption technologies built into each iOS device.



Each time a file is created, the data protection system generates a new 256-bit *file key*, which it gives to the hardware AES engine. The engine encrypts the file using this key - via the CBC mode of AES - every time the file is written to flash memory.

Every file is a member of one or more file classes, and each class is associated with a *class key*. A class key is protected by the hardware UID and, for some classes, the user's passcode as well. The file key is encrypted with one or more class keys, depending on which classes the file belongs to, and the result is stored in the file's metadata.

The metadata of all files in the filesystem is encrypted using the same random key: the *filesystem key*. The system generates this key when iOS is first installed, or when a user wipes and restarts the device.

When a file is opened, its metadata is decrypted first using the filesystem key, which reveals the encrypted file key. Next, the file key is decrypted using one or more class keys. Finally, the file key is used to decrypt the file as it is read from flash memory.

### 18.15 Security Quiz



#### Security Quiz

Mark all the answers that are true.



- All cryptographic keys are stored in flash memory
- Trusted boot can verify the kernel before it is run
- All files of an app are encrypted using the same key

### 18.16 Security Quiz Solution



**Security Quiz**  
Mark all the answers that are true.

All cryptographic keys are stored in flash memory

Trusted boot can verify the kernel before it is run

All files of an app are encrypted using the same key



### 18.17 Mandatory Code Signing

The iOS kernel controls which user processes and apps are allowed to run. iOS requires all executable code to be signed with an Apple-issued certificate to ensure that all apps come from a known and approved source and have not been modified in unauthorized ways.

Apps provided with the device, such as Mail or Safari, are already signed by Apple. Third-party apps must also be certified and signed using an Apple-issued certificate.

By requiring all apps on the device to be signed, iOS extends the concept of *chain of trust* from the kernel to the apps and prevents third-party apps from uploading unauthorized code or running self-modifying code.

A user-space daemon examines executable memory pages as they are loaded by an app to ensure that the app has not been modified since it was installed or explicitly updated.

### 18.18 Restricted App Distribution Model

A developer must first register with Apple and join the iOS developer program if they want to develop apps for iOS devices. Apple verifies the real-world identity of each developer - whether an individual or business - before issuing a certificate.

Developers use their certificates to sign their apps before submitting them to the App Store for distribution, which means that every app in the App Store can be traced back to an identifiable entity. Associating apps with the real-world identities of their developers serves as a deterrent to submitting malicious code.

Furthermore, Apple reviews all apps in the App Store to ensure that they operate as described and requires iOS devices to download apps exclusively from the official Apple App Store.

The restricted app distribution model, combined with app signing, makes it very difficult to upload malware to the App Store.

### 18.19 App Store Security Quiz



### App Store Security Quiz

Choose the best answer.

In 2013 researchers were able to bypass Apple's App store security. What method did they use?

- Uploaded malware disguised as an app without authorization, bypassing the review and check process.
- Uploaded an app that after it passed the review process morphed into malware.
- Uploaded an app that led users to a site that contained malware.



### 18.20 App Store Security Quiz Solution



## App Store Security Quiz

Choose the best answer.

In 2013 researchers were able to bypass Apple's App store security. What method did they use?

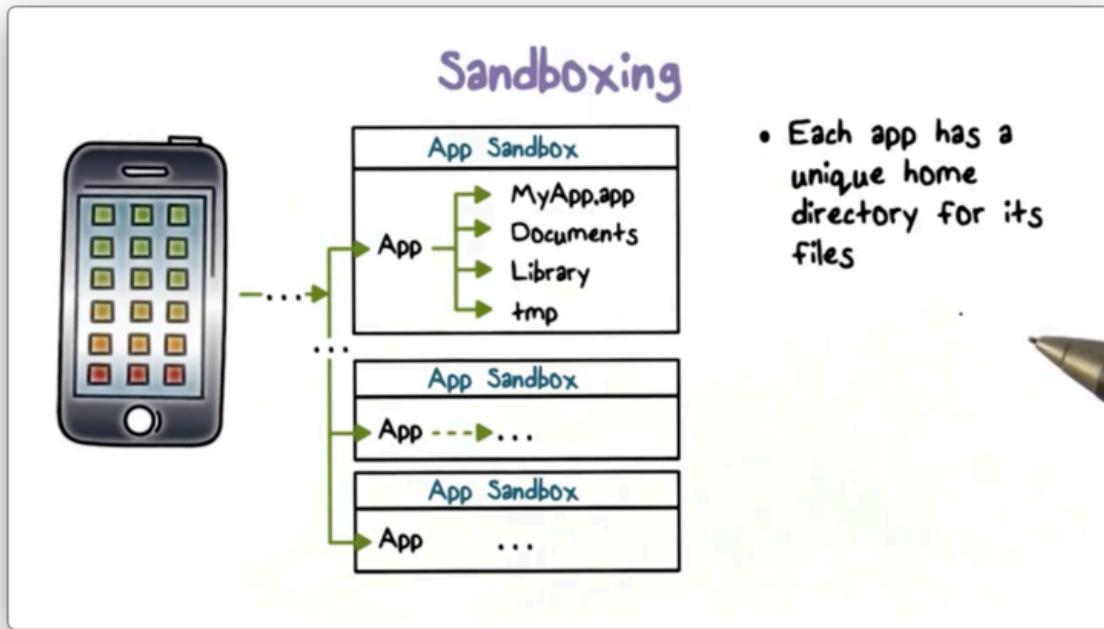
- Uploaded malware disguised as an app without authorization, bypassing the review and check process.
- Uploaded an app that after it passed the review process morphed into malware.
- Uploaded an app that led users to a site that contained malware.

Read more [here](#).

### 18.21 Sandboxing

Once an app resides on a device, iOS enforces additional security measures to prevent it from compromising other apps or the rest of the system.

Each app receives a unique home directory for its files, which is randomly assigned when the app is installed. This directory serves as a sandbox; that is, iOS restricts apps from accessing information outside of the directory.



If a third-party app needs to access external information, it must use services explicitly provided by iOS. This requirement prevents apps from unauthorized access or modification to information it does not own.

Additionally, the majority of iOS processes, including third-party apps, run as a non-privileged user, [mobile](#), which does not have access to crucial system files and resources. The iOS APIs do not allow apps to escalate their own privileges to modify other apps or iOS itself.

Finally, the entire iOS partition is mounted as read-only, and unnecessary tools such as remote login services are not included in the system software.

## 18.22 Address Space Layout Randomization

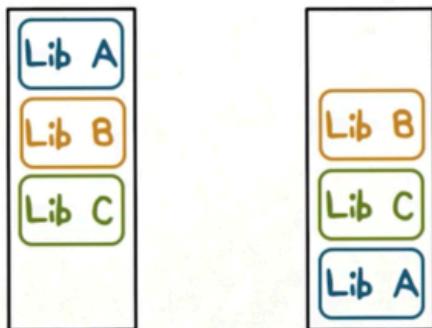
iOS has several other runtime security measures. One such measure is [address space layout randomization](#) (ASLR), which protects against the exploitation of memory corruption bugs.

A common class of attack - which includes the [return-to-libc](#) attack - involves an attacker estimating the addresses of nearby system functions and calling these functions to perform malicious actions or escalate privileges.

As a countermeasure, iOS randomly arranges the different program components in memory upon app initialization. This randomization makes it virtually impossible for an attacker to locate a useful library function to exploit.

## Address Space Layout Randomization

- Stack, heap, main executable, and dynamic libraries.



Memory Layout

### 18.23 iOS Security Quiz



#### iOS Security Quiz

Choose the best answer.

What weaknesses were exploited by researchers in the Apple apps security in 2015?

- The malware was uploadable to the Apple Apps store.
- The malware was able to bypass Sandbox security
- The malware was able to hijack browser extensions and collect passwords.
- All of the above.

### 18.24 iOS Security Quiz Solution



**iOS Security Quiz**  
Choose the best answer.

What weaknesses were exploited by researchers in the Apple apps security in 2015?

- The malware was uploadable to the Apple Apps store.
- The malware was able to bypass Sandbox security
- The malware was able to hijack browser extensions and collect passwords.
- All of the above.



Read more [here](#).

### 18.25 Data Execution Prevention

Another runtime security feature that iOS provides is data execution prevention. **Data execution prevention** is an implementation of the policy that makes writeable and executable pages mutually exclusive.

Specifically, iOS marks pages that are writable in runtime, such as pages that contain the stack, as non-executable, using the ARM processor's `execute never` feature. Reciprocally, iOS marks executable memory pages, such as pages that hold code instructions, as non-writeable.

This mutual exclusivity helps prevent code-injection attacks. To inject code, an attacker must write instructions into a memory page and then subsequently execute those instructions. Since a page cannot be both writeable and executable, an attacker can never execute injected code.

### 18.26 Passcodes and Touch ID

By setting up a device passcode, a user both prevents unauthorized access to their device and automatically enables data protection, which encrypts all of their files.

iOS supports 4-digit numeric and arbitrary-length alphanumeric passcodes. To discourage brute-force attacks, the iOS interface enforces escalating time delays after the entry of an invalid passcode. Users can choose to have the device automatically wiped if the passcode is entered incorrectly after ten consecutive attempts.

A user can opt to use Touch ID instead of a passcode. Touch ID is the fingerprint-sensing system that makes secure access to the device faster and easier.

### 18.27 iOS Quiz



**iOS Quiz**  
Mark all the true answers

- Each app runs in a sandbox and has its own home directory for its files
- All iOS apps must be reviewed and approved by Apple
- iOS apps can be self-signed by app developers



### 18.28 iOS Quiz Solution



**iOS Quiz**  
Mark all the true answers

**T** Each app runs in a sandbox and has its own home directory for its files

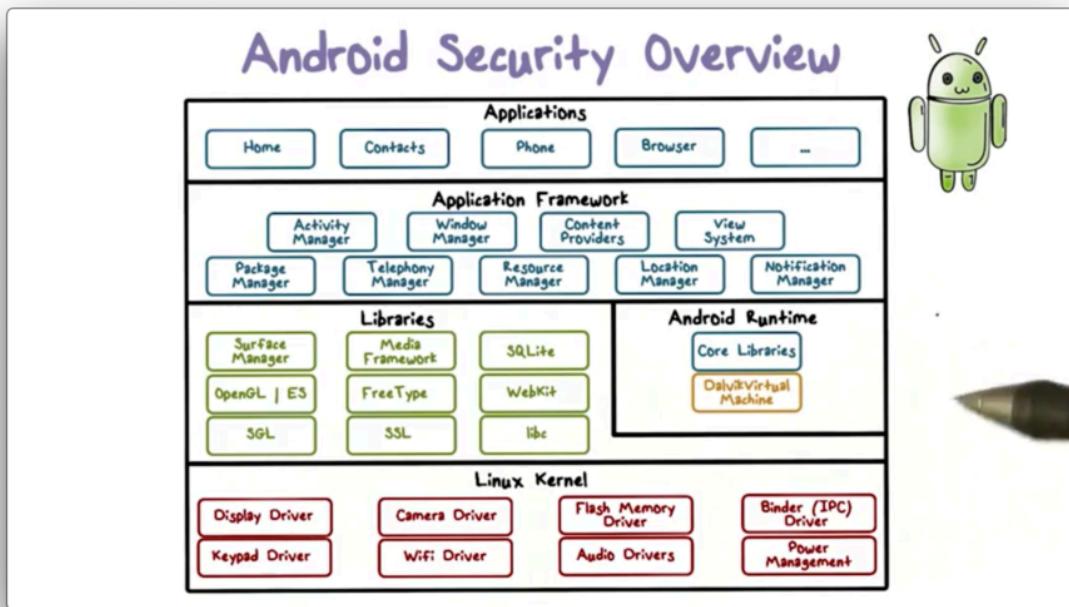
**T** All iOS apps must be reviewed and approved by Apple

**F** iOS apps can be self-signed by app developers



### 18.29 Android Security Overview

Android is implemented as a software-stack architecture, consisting of a Linux kernel, a runtime environment with corresponding libraries, an application framework, and a set of applications.



The Linux kernel sits at the lowest level of the architecture stack and provides a level of abstraction between device hardware and the upper layers of the stack.

Apps are commonly written in Java, which is first compiled to Java Virtual Machine (JVM) bytecode and then translated to bytecode that runs on the **Dalvik virtual machine** DVM, a virtual machine optimized for mobile devices. In particular, the DVM optimizes for memory, battery life, and performance.

The Android core libraries are Java-based libraries that are used for application development. Most of these libraries do not perform any work but instead serve as thin Java wrappers around a set of C and C++ based libraries. These underlying libraries fulfill a wide range of functions, including graphics rendering in 2D and 3D, SSL, and more.

The application framework is a set of services that collectively form the environment in which Android apps run. This framework allows apps to be constructed using reusable, interchangeable, and replaceable components.

Furthermore, an individual app can publish components and data for use by other apps. This capability allows apps to build on top of other apps, in addition to using default components exposed by the framework.

At the top of the software stack are the apps, which include apps that come with the device - such as contacts, phone, and email - as well as any other third-party apps the user has downloaded.

### **18.30 Application Sandbox**

Apps that run in virtual machines are essentially sandboxed in runtime. Sandboxed apps cannot directly interfere with the operating system or other apps, nor can they directly access the device hardware. Each app is granted a set of permissions at install time and can only perform operations permitted by these permissions.

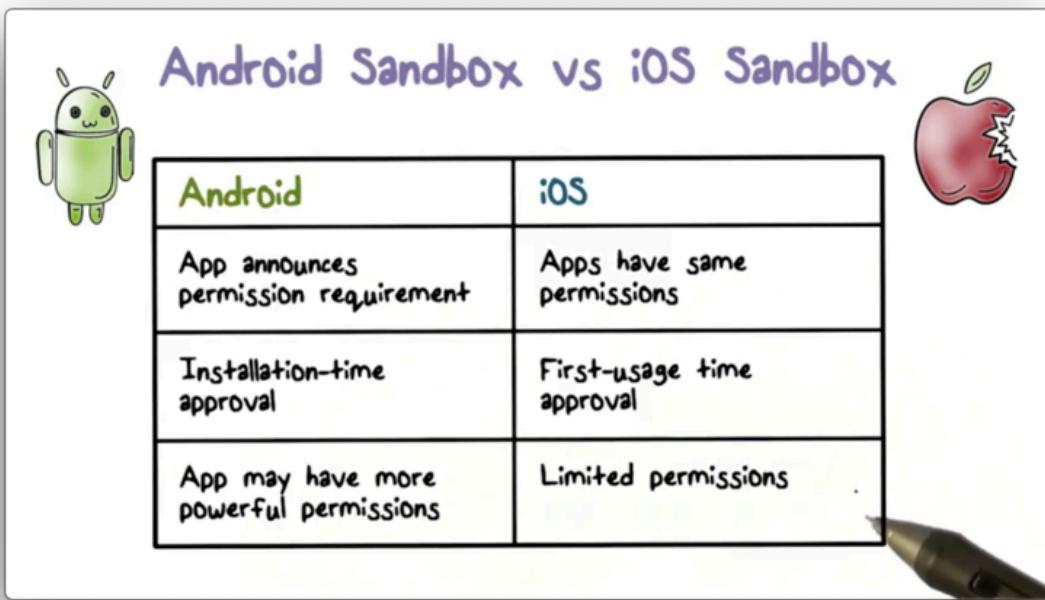
Android assigns a unique user ID (UID) to each app and runs it with that UID in a separate process. The kernel enforces security between apps and the system at the process level through standard Linux facilities, using the user and group IDs associated with an app to determine which system resources and functions an app can access.

By default, apps have limited access to the operating system and other apps. The operating system denies unauthorized requests - such as one app trying to read data owned by another app, or an app attempting to dial the phone without permission - unless the appropriate user privileges are present.

An app can announce the permissions it needs, and a user can grant these permissions during app installation. The permissions are typically implemented by mapping them to Linux groups that have the necessary read/write access to relevant system resources, such as files and sockets.

### **18.31 Android Sandbox vs iOS Sandbox**

From a security perspective, one of the main differences between the Android and iOS sandbox is how they handle permissions.



Android	iOS
App announces permission requirement	Apps have same permissions
Installation-time approval	First-usage time approval
App may have more powerful permissions	Limited permissions

Android apps can announce the permissions that they require, and users can approve these permissions at install time. Notably, Android apps can ask for very powerful permissions.

All iOS apps have the same set of basic permissions. If an app needs to access system resources or data - such as the user's address book - user approval is required at the first access request. In general, iOS apps have limited permissions.

### 18.32 Code Signing

Android also takes a very different approach than iOS in terms of code signing. In particular, all Android apps are self-signed by developers. A developer can create a public key, self-sign it to create a certificate, and then use the key to sign apps.

There is no central authority that signs third-party Android apps, and there is no vetting process for third-party app developers. Anybody can become an Android app developer, self-sign their apps, and upload them to the Google Play Store.

While Apple uses code signing to identify developers and verify app executables, Android uses code signing for different purposes.

Specifically, Android devices use code signing to ensure that updates for an app are coming from the same developer that created the app. Additionally, code signing helps manage the trust relationship

between apps so that they can share code and data.

### 18.33 Android Apps Quiz



### Android Apps Quiz

Mark all the true answers



- Android apps can be self-signed
- Android apps can have more powerful permissions than iOS apps



### 18.34 Android Apps Quiz Solution



The slide features a purple castle icon with crossed swords on the left. To its right, the text "Android Apps Quiz" is written in purple, and below it, "Mark all the true answers" is written in blue. In the center-left, there is a green Android robot icon. To the right of the robot are two statements, each preceded by a green checkbox containing a white 'X'. The first statement is "Android apps can be self-signed" and the second is "Android apps can have more powerful permissions than iOS apps".

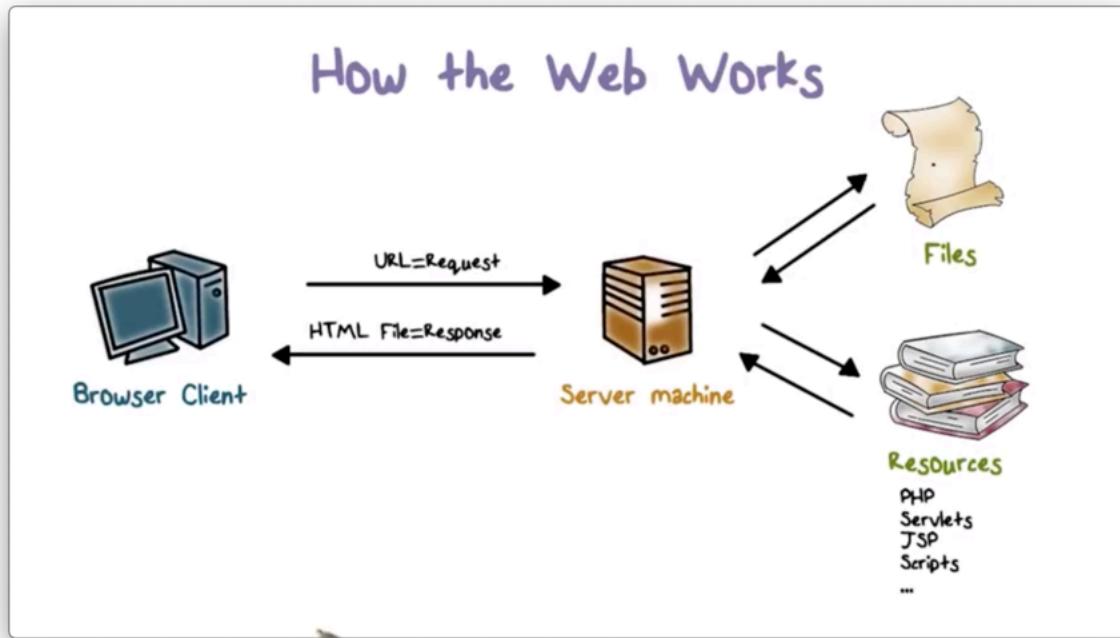
Android apps can be self-signed

Android apps can have more powerful permissions than iOS apps

## 19 Web Security

### 19.1 How the Web Works

A web browser and a web server communicate using the **HyperText Transfer Protocol** (HTTP). The browser requests documents through a URL, and the server responds with documents in **HyperText Markup Language** (HTML).

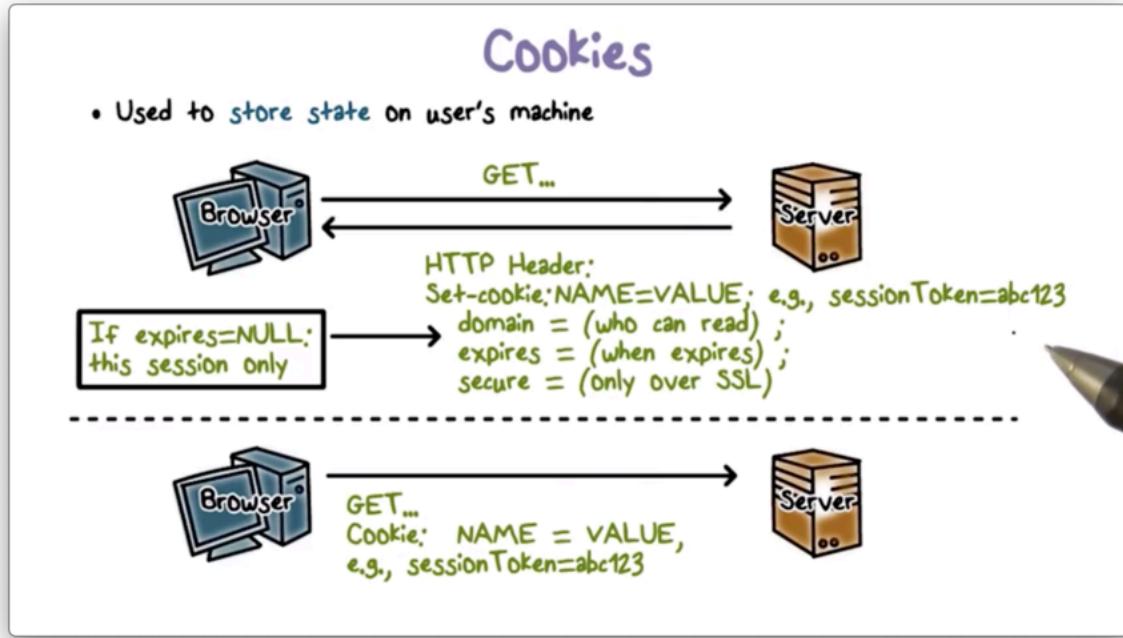


HTML documents can include text, graphics, video, audio, PostScript, JavaScript, and other components. The browser displays text and embedded graphics and executes the JavaScript and other helper scripts.

## 19.2 Cookies

Each HTTP request uses its own HTTP connection; that is, HTTP is a *stateless* protocol. For example, when navigating through a website, each visit to a URL generates a separate HTTP (really TCP) connection.

Browsers and servers use cookies as a way of carrying information, such as user authentication/session state, across multiple HTTP requests.



**Cookies** are small strings of text that a web server can create as part of any HTTP response using the `Set-cookie` HTTP header. Cookies are essentially key/value pairs, such as `userName=user123`.

In addition to the key/value pair itself, a cookie also contains some metadata, including expiration information, domain information, and security requirements for transmission.

A user's browser stores cookies and includes them in subsequent requests as a way to create and preserve state over fundamentally stateless connections.

### 19.3 Cookie Quiz



#### Cookie Quiz

Which of the following are true statements?

- Cookies are created by ads that run on websites
- Cookies are created by websites a user is visiting
- Cookies are compiled pieces of code
- Cookies can be used as a form of virus
- Cookies can be used as a form of spyware
- All of the above



#### 19.4 Cookie Quiz Solution



### Cookie Quiz

Which of the following are true statements?

- Cookies are created by ads that run on websites
- Cookies are created by websites a user is visiting
- Cookies are compiled pieces of code
- Cookies can be used as a form of virus
- Cookies can be used as a form of spyware
- All of the above



Cookies are just strings of text. They are not compiled code, and therefore cannot infect a system the way a virus can.

#### 19.5 The Web and Security

Can a browser trust the content it receives from the server?

Browsers rarely authenticate servers before communicating with them. Even if they do, the content that a server sends may not be trustworthy.

On the browser side, a website may have security vulnerabilities that allow it to display malicious content or execute malicious scripts. Additionally, a website might link to other websites that have security vulnerabilities themselves.

On the server side, a website runs applications that process requests from browsers and often interacts with backend servers to produce content for users.

These web applications, like any software, may have security vulnerabilities. Furthermore, many websites do not authenticate users, which means that attackers are free to send requests designed to exploit security vulnerabilities in these web applications.

## 19.6 Web Security Quiz



### Web Security Quiz

Mark each statement as true or false.

- Web browser can be attacked by any web site that it visits
- Even if a browser is compromised, the rest of the computer is still secure
- Web servers can be compromised because of exploits on web applications

## 19.7 Web Security Quiz Solution



### Web Security Quiz

Mark each statement as true or false.

T Web browser can be attacked by any web site that it visits

F Even if a browser is compromised, the rest of the computer is still secure

T Web servers can be compromised because of exploits on web applications



## 19.8 Cross-Site Scripting (XSS)

Many websites, including social networking sites, blogs, forums, and wikis, display user-supplied data. For example, a user Joe might visit a website and fill out a form indicating that his name is Joe. The website might greet him with a page saying, “Hello Joe.”

Suppose that instead of entering his name, Joe submits the following.

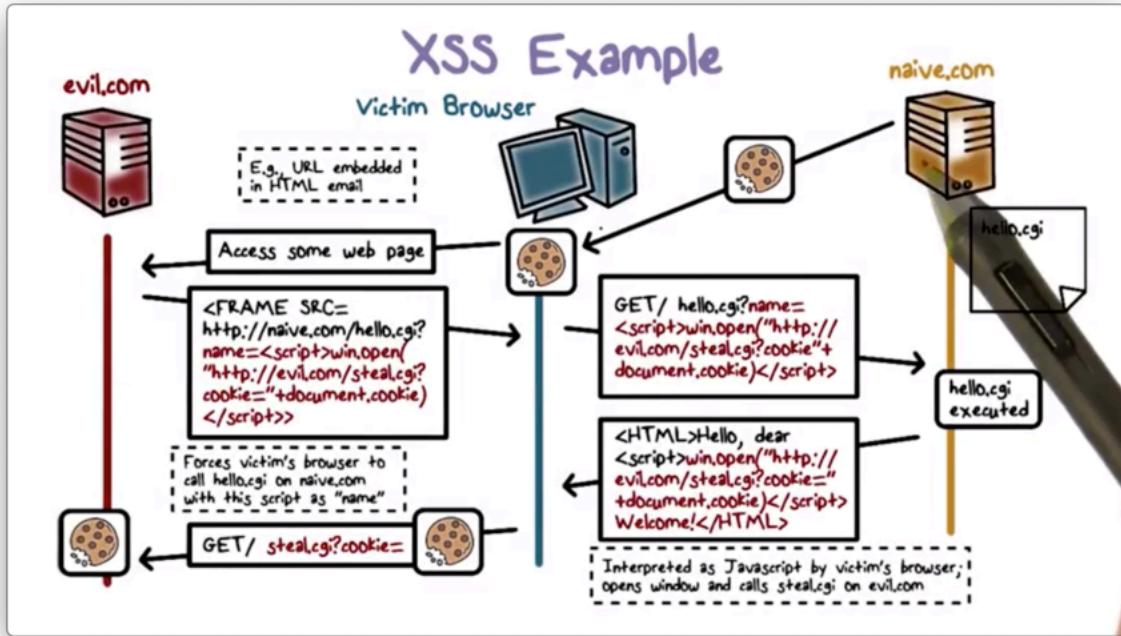
```
1 <script type="text/javascript">alert('Hello World')</script>
```

The website takes this string as the user’s name and includes it in the HTML page sent to the browser. Therefore, when the browser displays this webpage, the script runs, and the webpage displays “Hello World.”

## 19.9 XSS Example

In a **cross-site scripting** (XSS) attack, an attacker tricks the browser into executing malicious scripts without the user’s knowledge.

The following diagram presents how this attack might work.



First, the user logs in to a vulnerable site, `naive.com`, and the browser stores a cookie to `naive.com`. Next, the attacker directs the user to `evil.com`, which returns a page containing a hidden iframe.

The iframe forces the browser to visit `naive.com` and invoke the `hello.cgi` web application, sending the malicious script as the name of the user. `hello.cgi` at `naive.com` then echos the malicious script in the HTML page sent back to the user's browser.

The browser displays the HTML page and executes the malicious script, which steals the user's cookie to `naive.com` and sends it to the attacker. Since the cookie can include session authentication information for `naive.com`, the attacker can now impersonate this user on `naive.com`.

### 19.10 XSS Quiz



#### XSS Quiz

Mark each statement as true or false.

- When a user's browser visits a compromised or malicious site, a malicious script is returned
- To prevent XSS, any user input must be checked and preprocessed before it is used inside html



### 19.11 XSS Quiz Solution

**XSS Quiz**  
Mark each statement as true or false.

- When a user's browser visits a compromised or malicious site, a malicious script is returned
- To prevent XSS, any user input must be checked and preprocessed before it is used inside html

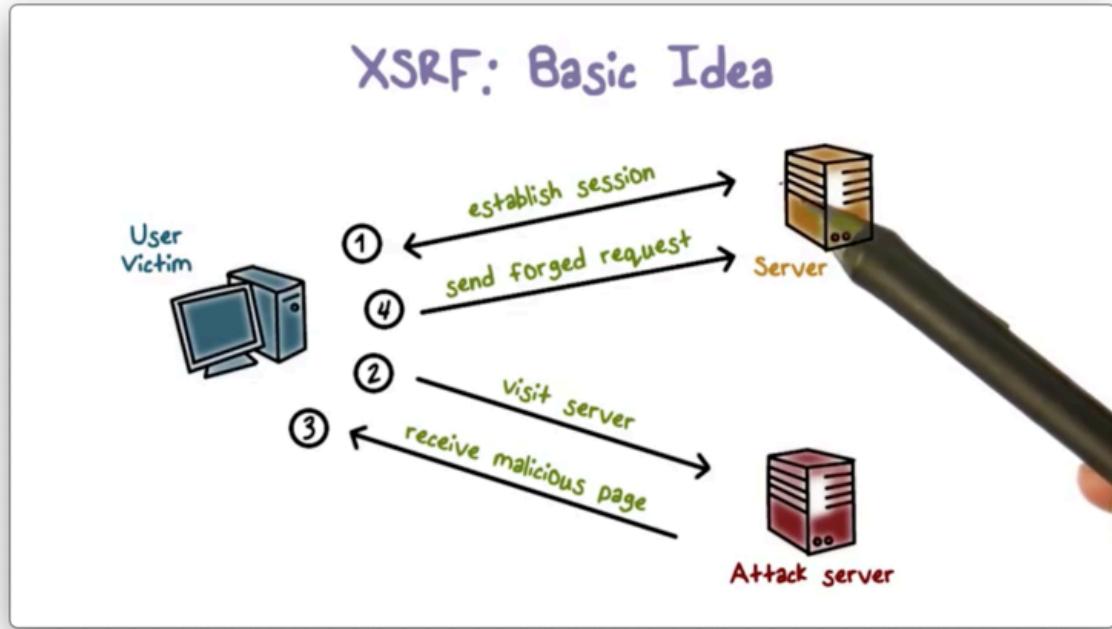
### 19.12 XSRF: Cross-Site Request Forgery

When a user logs in to a site, the server usually writes a cookie to the user's browser that contains session authentication information for that user on that site.

The cookie lives in the user's browser as long as they keep the session alive. Once they log out of the website, the server resets the cookie.

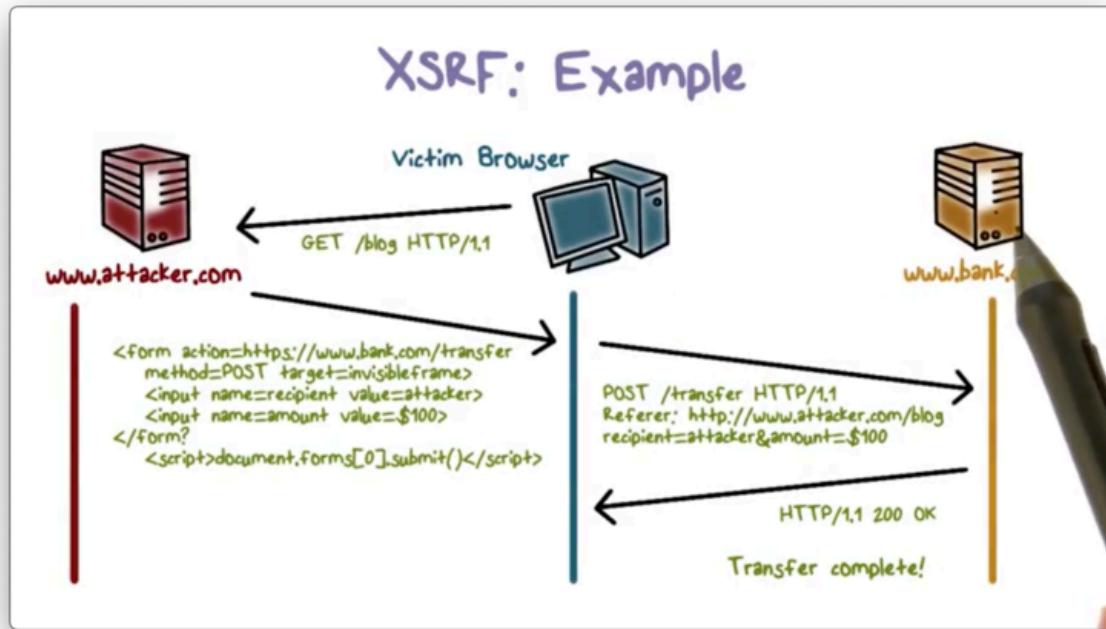
If a user browses to a malicious site in the middle of their session with a trusted site, a script on the malicious site can potentially read the user's cookie to the trusted site and use it to forge requests to that site. This type of attack is called **cross-site request forgery**.

The user never sees this malicious request, since the malicious site often triggers it from a hidden iframe. Additionally, the trusted server doesn't find the request suspicious since it contains the user's cookie, which the server itself granted.



### 19.13 XSRF Example

Here is an illustration of an XSRF example involving a trusted site, bank.com, and a malicious site, attacker.com.



The user logs in to bank.com and keeps the session alive, which means that the browser has a cookie to bank.com. Meanwhile, the attacker phishing the user and directs them to the malicious site attacker.com.

When the user visits attacker.com, their browser downloads and executes the malicious page. The scripts on this page direct the browser to make a request to bank.com on the user's behalf.

Since the user is still logged in to bank.com, their browser also sends the bank.com cookie along with the forged request. As a result, bank.com believes that the request originated from the user and therefore executes the request.

### 19.14 XSRF vs XSS

In cross-site scripting, an attacker injects a script into a badly-implemented website that does not validate user input. As a result, when a user visits this website, their browser downloads and executes the malicious script.

In cross-site request forgery, an attacker forges user requests to a website. As a result, the website executes the attacker's malicious actions as if they were initiated and authorized by the user.

Both XSS and XSRF are the results of security weaknesses in websites, in particular, the lack of authenticating and validating user input.

### 19.15 XSRF Quiz



#### XSRF Quiz

Which of the following methods can be used to prevent XSRF?

- Checking the http Referer header to see if the request comes from an authorized page.
- Use synchronizer token pattern where a token for each request is embedded by the web application in all html forms and verified on the server side.
- Logoff immediately after using a web application.
- Do not allow browser to save username/password and do not allow web sites to "remember" user login
- Do not use the same browser to access sensitive web sites and to surf the web freely
- All the above



### 19.16 XSRF Quiz Solution



### XSRF Quiz

Which of the following methods can be used to prevent XSRF?

- Checking the http Referer header to see if the request comes from an authorized page.
- Use synchronizer token pattern where a token for each request is embedded by the web application in all html forms and verified on the server side.
- Logoff immediately after using a web application.
- Do not allow browser to save username/password and do not allow web sites to "remember" user login
- Do not use the same browser to access sensitive web sites and to surf the web freely
- All the above



### 19.17 Structured Query Language (SQL)

SQL is the most widely-used database query language. We use SQL to retrieve database information, such as tables or records, and modify database information; for example, adding records to a table or modifying the specific values of a record.

### 19.18 Sample PHP Code

Many websites contain forms that users fill out with information that they want the database to persist. When a user submits form data, the web server typically runs a program to transform the data into an SQL query and then sends the query to the database server for execution.

For example, the following PHP snippet builds an SQL query based partially on user input.

## Sample PHP Code

- Sample PHP

```
$selecteduser = $_GET['user'];
$sql = "SELECT Username, Key FROM Key".
      "WHERE Username='\$selecteduser'";
$rs = $db->executeQuery($sql);
```



The security threat here is that specially crafted input can generate malicious SQL queries that can lead to compromise of data confidentiality and integrity.

### 19.19 Example Login

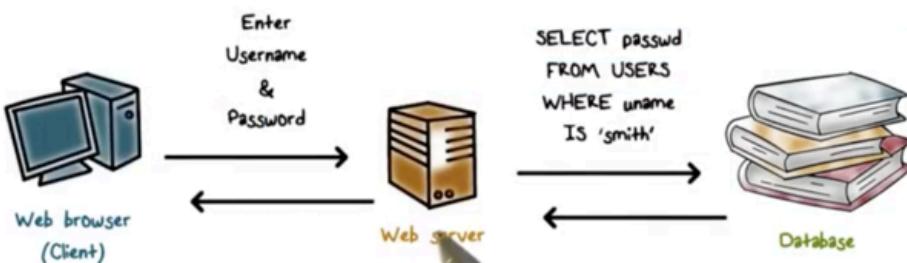
Here is an example of a web form consisting of username and password fields that a user might use to log in to a web site.

## Example Login Prompt



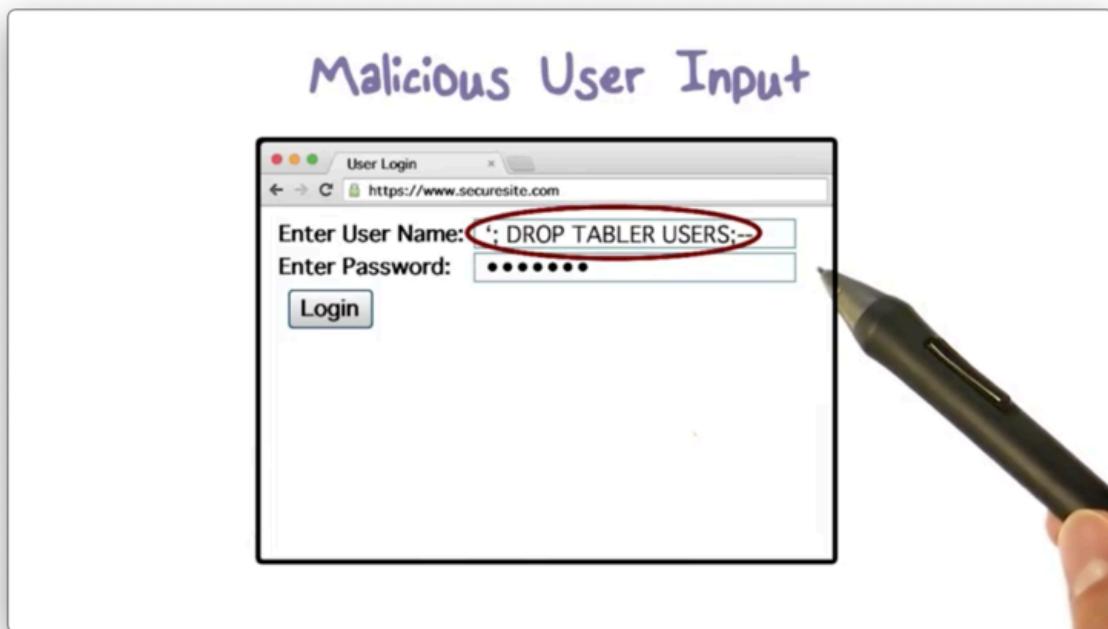
To authenticate the user, the web server first needs to compare the received password hash with the stored password hash for the user. The web server issues an SQL query to the database to retrieve the stored user information.

## Normal Login

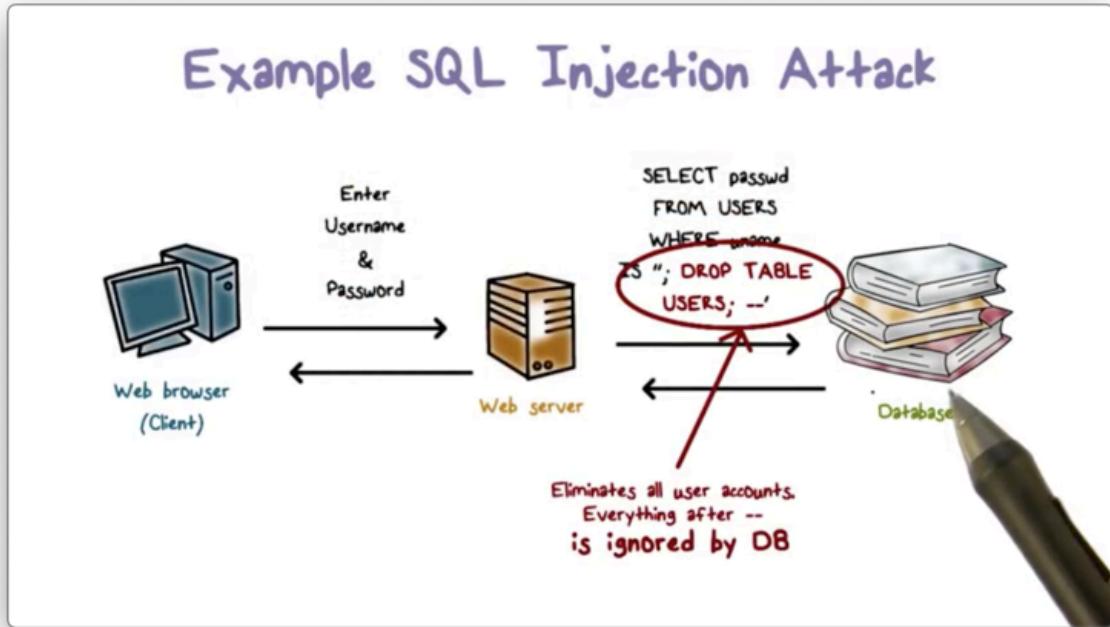


## 19.20 Malicious User Input

Suppose an attacker enters this malicious string as the user name.



What is going to happen?



The SQL query sent from the web server to the backend database server triggers the deletion of all user records.

### 19.21 SQL Injection Quiz



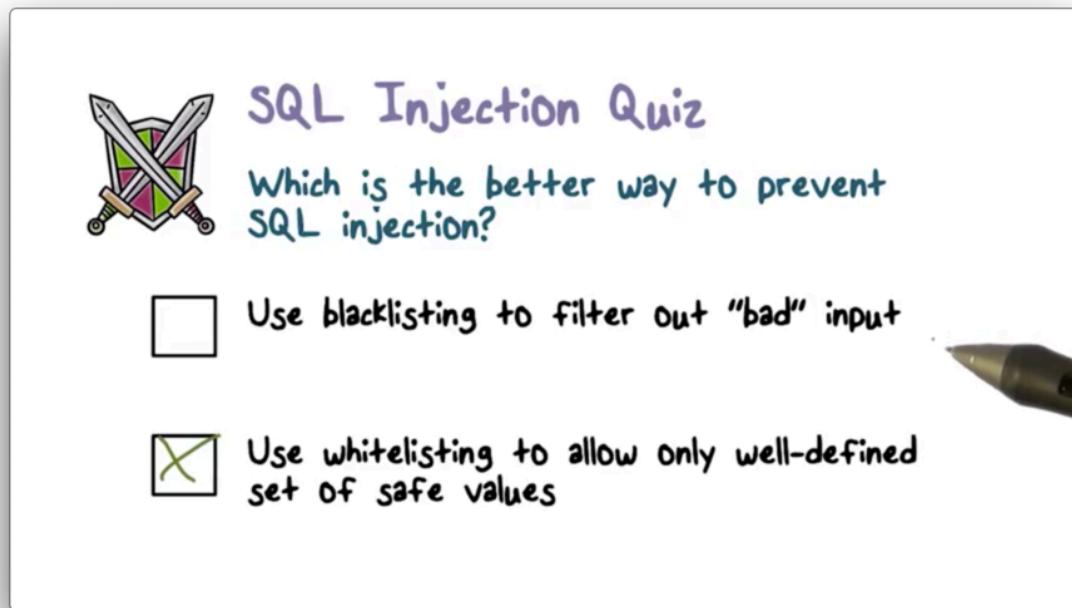
#### SQL Injection Quiz

Which is the better way to prevent SQL injection?

- Use blacklisting to filter out "bad" input
- Use whitelisting to allow only well-defined set of safe values



### 19.22 SQL Injection Quiz Solution



A slide titled "SQL Injection Quiz" featuring a shield with crossed swords icon. The question asks, "Which is the better way to prevent SQL injection?" Two options are listed: "Use blacklisting to filter out 'bad' input" (unchecked box) and "Use whitelisting to allow only well-defined set of safe values" (checked box with a green X). A pen icon is visible on the right.

**SQL Injection Quiz**

Which is the better way to prevent SQL injection?

Use blacklisting to filter out "bad" input

Use whitelisting to allow only well-defined set of safe values

## 20 Cyber Security

### 20.1 Managing Security

The digital components of a business - be they routers, operating systems, or databases - are crucial to successful business operation. A business must implement appropriate security measures to protect these assets against malicious actors attempting to gain unauthorized access.

In addition to mitigating explicit threats, companies often secure our digital assets for legal and compliance reasons. For example, HIPAA mandates a process for sharing digital health data. Therefore, an entity that deals with health data may need to implement specific security measures to meet compliance requirements.

We've spent time talking about the various technical controls - such as firewalls, intrusion detection systems, and anti-virus programs - that organizations can use to secure their computer systems.

Of course, a business must first understand the risks that they face and the threat landscape surrounding their organization and digital assets before implementing any security measure.

Additionally, technical implementations carry associated costs, and businesses are wise to ensure that the cost of securing a system does not exceed the value of the system.

## 20.2 Key Challenges

A business faces different challenges during the process of building and implementing a cybersecurity plan.

For example, a business must identify their assets and the relative value of these assets. Furthermore, a business must understand which assets are at risk, how severe the risk is, and how threat sources are likely to present themselves.

After a business surveys the security landscape of their organization, they then have to implement the security measures. This implementation step requires awareness of the various security solutions and controls present in the market.

Additionally, organizations must perform a cost-benefit analysis before deploying a particular solution. While a business wants to reduce risk, they likely don't want to spend more than the value of the asset they are protecting.

Finally, a business must understand how the employees operate within the organization and the various workflows that a new security implementation can impact. Employees are likely to skirt security measures that drastically hinder their productivity.

### 20.3 Network Use Policy Quiz



#### Network Use Policy Quiz

Cyber security planning and management in an enterprise must define allowed computer and network use by employees. Georgia Tech's computer and network use policy strives to do this for students, faculty and staff.

Check all that you think are required by this policy:

- Georgia Tech account passwords should be changed periodically.
- A compromise of a computer should be reported to someone responsible for cyber security at Georgia Tech.
- Georgia Tech computers cannot be used to download illegal content (e.g., child pornography).



## 20.4 Network Use Policy Quiz Solution



### Network Use Policy Quiz

Cyber security planning and management in an enterprise must define allowed computer and network use by employees. Georgia Tech's computer and network use policy strives to do this for students, faculty and staff. Check all that you think are required by this policy:

- Georgia Tech account passwords should be changed periodically.
- A compromise of a computer should be reported to someone responsible for cyber security at Georgia Tech.
- Georgia Tech computers cannot be used to download illegal content (e.g., child pornography).



## 20.5 Botnet Quiz



### Botnet Quiz

A botnet operator compromises a number of computers in a company. The malware executed by the bots only sends large amounts of spam email but does not exfiltrate sensitive data or interfere with legitimate activities. Choose the appropriate action by the company in this situation:

- The company should detect and prevent abuse of its resources by unauthorized parties.
- Since it poses no risk to company's sensitive data or normal operations, it can be ignored.

## 20.6 Botnet Quiz Solution



### Botnet Quiz

A botnet operator compromises a number of computers in a company. The malware executed by the bots only sends large amounts of spam email but does not exfiltrate sensitive data or interfere with legitimate activities. Choose the appropriate action by the company in this situation:

- The company should detect and prevent abuse of its resources by unauthorized parties.
- Since it poses no risk to company's sensitive data or normal operations, it can be ignored.

## 20.7 Security Planning

So far, we've identified the assets that need protection, determined the parties responsible for understanding the security needs of the organization, examined the technical controls at our disposal, and understood the impact of different implementations on user workflow.

During the process of security planning, a business has to understand that no matter how well they deploy their security, they can never reduce the risk to zero. Therefore, the planning process needs to cover response and recovery as well as prevention and detection.

One component of incident response is accountability. When an incident occurs, the business needs to understand which responsibilities were not fulfilled and take steps to hold the appropriate employees accountable.

## 20.8 Assets and Threats

A business must take inventory of their assets to understand which assets are valuable. These assets include servers, routers, switches, laptops, and mobile devices.

In addition to the physical assets themselves, a business needs to consider the software utilized by these assets; for example, the operating systems running on the servers, laptops and mobile devices; the databases that store large amounts of data, and; the services and applications running on these devices.

The organization must concern themselves with the data stored in the system, whether structured data that lives in databases or unstructured data in files in the filesystem. Some of this data, such as intellectual property, financial records, or customer information, can be sensitive and, therefore, may require additional security measures.

Assets only require security if they are at risk, so a business needs to understand who they are securing their assets from before implementing security controls. These threat actors could be external cybercriminals, passionate hacktivists, or even disgruntled employees.

## 20.9 Security Audit Quiz



### Security Audit Quiz

A news story in 2014 reported that an inspector general's report gave Veteran Affairs (VA) a failing grade for 16th straight year. The CIO of VA discussed a number of challenges that could explain this grade. Mark the ones that you think could be possible reasons: (See the instructor notes for a link to the article)

- The need to manage cyber security for over a million devices each running many services
- Lack of sense of urgency in fixing cyber vulnerabilities.
- Choosing to support key functions even when this could introduce vulnerabilities.



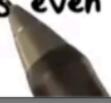
## 20.10 Security Audit Quiz Solution



### Security Audit Quiz

A news story in 2014 reported that an inspector general's report gave Veteran Affairs (VA) a failing grade for 16th straight year. The CIO of VA discussed a number of challenges that could explain this grade. Mark the ones that you think could be possible reasons: (See the instructor notes for a link to the article)

- The need to manage cyber security for over a million devices each running many services
- Lack of sense of urgency in fixing cyber vulnerabilities.
- Choosing to support key functions even when this could introduce vulnerabilities.



NOTE: answers 1 and 3 are correct.

### 20.11 CISO Quiz



#### CISO Quiz

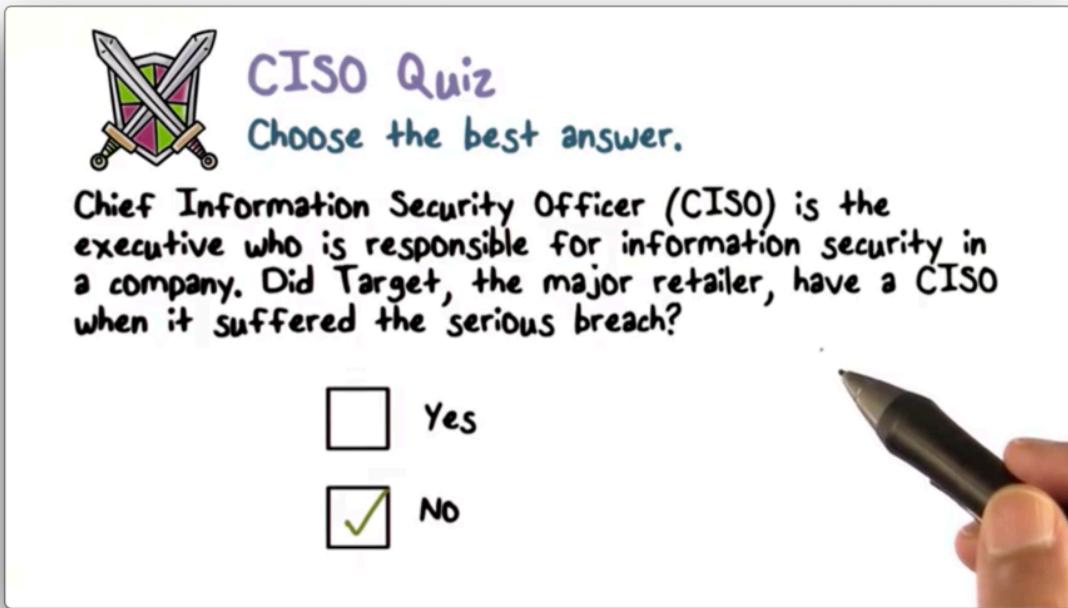
Choose the best answer.

Chief Information Security Officer (CISO) is the executive who is responsible for information security in a company. Did Target, the major retailer, have a CISO when it suffered the serious breach?

Yes

No

## 20.12 CISO Quiz Solution



**CISO Quiz**  
Choose the best answer.

Chief Information Security Officer (CISO) is the executive who is responsible for information security in a company. Did Target, the major retailer, have a CISO when it suffered the serious breach?

Yes

No

## 20.13 Security Planning: Controls

Security planning involves understanding which security controls make the most sense for a given organization's assets and threat posture. Let's take a look at a few of these controls.

**Identity and Access Management** (IAM) is a popular security control. The "identity" part of IAM serves to identify a user who is requesting any type of resource present in the system. The "access management" component of IAM serves to perform access control on the resource, ensuring that a user has sufficient permission to access the resource they have requested.

Credentialing is another security measure that concerns resource access. When a business provides a user with a credential, the credential serves as a proxy for the user's identity, allowing them to access resources without explicitly identifying themselves each time.

A business may have certain password policies, which dictate how long a password must be and how frequently a password needs to change. Additionally, some organizations may require multi-factor authentication; for example, a user may need to use a password and a smart card to access specific resources.

A business can deploy network and host defenses to protect their resources. For example, firewalls

control the traffic that enters and exits a network, and intrusion detection systems and intrusion prevention systems monitor the traffic at the network level and raise alarms when they detect suspicious activity. Anti-virus systems running on local machines help control which external software a user can download and run.

If employees need to access critical systems from external networks, a business may require a VPN solution to tunnel business traffic through these networks securely. Additionally, organizations that permit employees to bring their own devices need to have a plan for securing those devices.

New vulnerabilities are found in software all the time. A business must have a strategy for discovering vulnerabilities and patching them in a timely fashion.

Besides deploying technological controls, a business can also look at employee education. For example, many companies periodically run fake phishing campaigns to show employees what a real phishing attack might resemble. Educational exercises like this can supplement purely technical security solutions.

## 20.14 Security Planning: Security Policy

Security planning requires that a business develops a security policy. A **security policy** articulates the security objectives and goals of an organization, and specifies the rules the organization needs to implement to achieve those objectives.

A security policy often incorporates legal, compliance, and business rationale for the rules it chooses to implement. Security policies may concern prevention and detection, as well as recovery and response.

An organization might have many security goals, such as

- preventing unauthorized data disclosure
- ensuring system availability
- guarding against data corruption
- abiding by compliance requirements

To achieve these goals, an organization must implement certain rules. For example, many companies have web and email policies that restrict the use of these facilities from within the corporate network. A web policy may dictate which sites an employee can visit, while an email policy may restrict a user from corresponding with certain addresses or opening certain attachments.

## 20.15 Georgia Tech Computer and Network Use Policy

Like many organizations, Georgia Tech (GT) has a security policy, the [Computer Network Usage and Security Policy](#). One guiding principle of this policy is to protect valuable IT resources that GT owns, including the data stored on and services used by these resources. Another guiding principle is to prevent any illegal activity from taking place on GT networks and devices.

You can read the entire policy by clicking the link above. We cover a few interesting highlights here.

The policy talks about intellectual property and copyright. As a research institution, GT creates intellectual property that resides on their computers and must decide how to store and share that information. Additionally, GT has to explicitly address copyrighted material since many colleges got in trouble in the past when students would use university networks to download illegal music.

GT is home to individuals from all over the world, and many people work with counterparts in other countries. Since working across national boundaries can raise issues related to customs and border control, the security policy must address these issues.

An organization must define responsibility and accountability when creating a security policy. The security planning process must involve identifying who is responsible for what resources and how they should be held accountable for those resources.

The GT policy defines a distributed model of responsibility. The centralized [Office of Information Technology](#) is responsible for the overall university network. Individual devices, such as workstations, are the responsibility of particular units - the College of Computing, for example - or the individual in possession of the devices.

### 20.16 Computer Use Policy Quiz



#### Computer Use Policy Quiz

Choose the best answer.

Does Georgia Tech's computer and network use policy prohibit personal use of university resources?

Yes

No



### 20.17 Computer Use Policy Quiz Solution



#### Computer Use Policy Quiz

Choose the best answer.

Does Georgia Tech's computer and network use policy prohibit personal use of university resources?

Yes

No



### 20.18 Student Privacy Quiz



#### Student Privacy Quiz

Choose the best answer.

Georgia Tech systems store student data such as grades. The Institute must protect such data due to...

Regulatory reasons

Because the data is sensitive it can only be disclosed to student and his/her family

### 20.19 Student Privacy Quiz Solution



### Student Privacy Quiz

Choose the best answer.

Georgia Tech systems store student data such as grades. The Institute must protect such data due to...

Regulatory reasons *FERPA*

Because the data is sensitive it can only be disclosed to student and his/her family

## 20.20 Anthem Breach Quiz



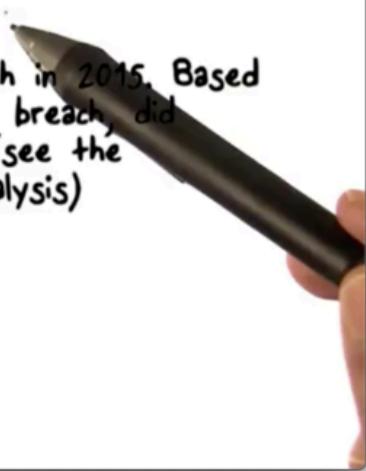
### Anthem Breach Quiz

Choose the best answer.

Anthem suffered from a major breach in 2015. Based on an analysis of its response to the breach, did Anthem respond well to the breach? (see the [instructor notes](#) for a link to the analysis)

Yes

No



## 20.21 Anthem Breach Quiz Solution



### Anthem Breach Quiz

Choose the best answer.

Anthem suffered from a major breach in 2015. Based on an analysis of its response to the breach, did Anthem respond well to the breach? (see the **instructor notes** for a link to the analysis)

Yes

No



Read more [here](#) and [here](#).

## 20.22 Cyber Risk Assessment

Unless a business disconnects from the rest of the world, they cannot eliminate their risk. Regardless of how well an organization plans or how many controls they implement, the possibility of compromise is greater than zero. Systems have unknown vulnerabilities, and people get phished: something can always go wrong.

A business must perform cost-benefit analyses when planning to implement security controls. An investment in cybersecurity requires money and effort, and the investment decisions must be based on risk and its reduction. A business may want to understand how much risk is reduced, from a financial perspective, as a result of purchasing some security control.

These decisions require businesses to quantify their risk. While various frameworks can help a business reduce relative risk, putting a dollar amount to the overall risk faced by an organization is not an easy task.

## 20.23 Quantifying Cyber Risk

When a business wants to quantify risk, they need to consider two things. First, they need to look at the probability of an adverse security event happening. For a given set of assets, threat landscape, and set of security controls, there is some nonzero probability of system compromise. Second, the business must consider the financial impact of the adverse event. For example, a data breach may cost a company \$10 million.

For a probability  $P$  and impact  $I$ , we can define **risk exposure**  $R = PI$ . For example, given a 50% chance of an adverse security event that is going to cost \$10 million, the risk exposure is \$5 million.

An organization wants to reduce their risk exposure, and they can achieve this reduction by implementing security controls. Since every control comes at a cost, an organization is smart to choose those controls that bring the most significant risk reduction relative to the implementation cost.

Given a pre-implementation risk  $pre$  and a post-implementation risk  $post$  and an implementation cost  $cost$  for a particular security measure, we can define **risk leverage**  $L = (pre - post) / cost$ .

Naturally, a business wants to choose controls that maximize their risk leverage. Any control under consideration should have a risk leverage higher than one; otherwise, the cost of the control is higher than the amount of risk it reduces.

## 20.24 Managing Cyber Risk

Risk assessment is a challenging exercise. A business may not fully be aware of the vulnerabilities present on their systems, and they may be ignorant of the threats seeking to compromise these systems. Without this knowledge, they can only guess at the likelihood of a successful attack.

Instead of trying to understand the probability of an attack, an organization might choose to examine the impact of an attack, which can be vast and expensive. For example, the business might suffer a loss of reputation or a decrease in sales. They may have to spend money to bring in cybersecurity firms for incident management and PR. They may incur legal costs, such as being forced to buy identity theft protection for their customers after a data breach.

After a business performs a risk assessment, they have three choices concerning managing their risk: accept it, transfer it, or reduce it.

Occasionally, a company may decide that both the probability and impact of an attack are so low that they are comfortable with the risk as is. Alternatively, a business may choose to transfer their risk by taking out an insurance policy; that way, if something bad happens, the cost is borne by someone else. Finally, an organization can reduce its risk by deploying new technology solutions, educating their employees, and providing security awareness training.

## 20.25 Security Breach Quiz



### Security Breach Quiz

Mark all applicable choices.

A company stores sensitive customer data. The impact of a breach of such data must include...

- Cost of purchasing identity theft protection for customers
- Loss of business due to reduced customer confidence
- Compensation for new cyber security personnel the company hires to better manage cyber security in the future



## 20.26 Security Breach Quiz Solution



### Security Breach Quiz

Mark all applicable choices.

A company stores sensitive customer data. The impact of a breach of such data must include...



Cost of purchasing identity theft protection for customers



Loss of business due to reduced customer confidence



Compensation for new cyber security personnel the company hires to better manage cyber security in the future



### 20.27 Reducing Exposure Quiz



#### Reducing Exposure Quiz

Choose the best answer.

A company is considering two possible IDS solutions to reduce its exposure to attacks on its network. The first one costs \$100K and reduces risk exposure by \$150K. The second one costs \$250K but reduces risk exposure by \$500K. Which solution would you recommend?

Cost	Risk Reduct
100	150
250	500

Cheaper solution that costs \$100K

More expensive solution that costs \$250K

### 20.28 Reducing Exposure Quiz Solution



## Reducing Exposure Quiz

Choose the best answer.

A company is considering two possible IDS solutions to reduce its exposure to attacks on its network. The first one costs \$100K and reduces risk exposure by \$150K. The second one costs \$250K but reduces risk exposure by \$500K. Which solution would you recommend?

Cheaper solution that costs \$100K

More expensive solution that costs \$250K

Cost	Risk Reduct
100	150
250	500

$\frac{150}{100} = 1.5$

### 20.29 Cyber Insurance Quiz



#### Cyber Insurance Quiz

Choose the best answer.

Cyber insurance is still not very popular. Based on a 2014 survey, what percentage of customers of major insurance brokers were interested in buying cyber insurance? (see the instructor notes for a link to the survey)

- Less than 25% (small)
- Over 50% (significant)

### 20.30 Cyber Insurance Quiz Solution



## Cyber Insurance Quiz

Choose the best answer.

Cyber insurance is still not very popular. Based on a 2014 survey, what percentage of customers of major insurance brokers were interested in buying cyber insurance? (see the instructor notes for a link to the survey)

Less than 25% (small)  
 Over 50% (significant)

### 20.31 Enterprise Cyber Security Posture Part 1

The whole point of security planning and security management is to become better prepared when it comes to cybersecurity. **Cybersecurity posture** captures this level of preparation; roughly, it refers to how “good” an organization is at cybersecurity. A cybersecurity posture can either be reactive or proactive.

#### 20.31.1 Reactive

Unfortunately, most postures are reactive. In other words, an organization worries about cybersecurity because of some external entity or event.

In some cases, organizations have legal or regulatory requirements they must meet. If a new regulation stipulates that a particular change must be made, an organization must abide by that regulation.

Other times, an organization is merely responding to customer demands. If customers or other companies refuse to conduct business with an organization that has lax security, that organization would be wise to adjust their cybersecurity posture.

Organizations also react to adverse security events that they have suffered. A data breach, for example, can serve as an impetus to review and revise security practices.

Organizations may also react to adverse security events that others have experienced. For instance, when a bank suffers a breach, other banks may react by ensuring that the exploited vulnerabilities don't exist in their systems.

### **20.31.2 Proactive**

A proactive approach to cybersecurity is best. An organization should conduct an inventory of their assets, assess their risk, plan and analyze, and implement controls that achieve risk reduction, instead of waiting for some external force.

Someone within the company should be worrying about cybersecurity full-time. This individual needs to focus on protecting assets, educating and empowering employees, drafting and maintaining security policies, and holding stakeholders accountable.

Additionally, this individual needs to have influence within the organization, as well as exposure to the highest levels of decision-making. Board-level conversations should address cybersecurity risk and investment, whether in additional personnel or technical controls. Without board-level stakeholders, this individual will struggle to accomplish their goals.

In short, this person, often a Chief Information Security Officer (CISO), must be the cybersecurity champion within the organization and use their expertise and influence to position the organization to proactively mitigate cybersecurity risk.

## **20.32 Enterprise Cyber Security Posture Part 2**

A cybersecurity champion, such as a CISO, needs to make their case to upper management to justify the security policies they want to implement. This task can be difficult for two reasons: first, risk is difficult to quantify, and, second, cybersecurity does not make money.

The primary economic argument for cybersecurity is based on return on investment (ROI), not revenue generation. For example, an investment of \$5 million in cybersecurity that shields an organization from a \$10 million security incident is a worthwhile spend.

Of course, demonstrating this sort of cost-benefit analysis to upper management is difficult. While a CISO would like to quantify risk and risk reduction precisely, exact numbers are notoriously hard to derive. They can present their analyses based on their experience and perception but should be prepared for other leaders to question what they perceive as unsubstantiated opinions.

### 20.33 Security Planning and Management

## Security Planning and Management

- Values at risk
  - Assets, reputation etc.
- Threats and attack vectors
- Plan, implement and manage
  - Deploy appropriate controls
  - Empower people and hold them responsible
  - Plan for response and remediation (do not be surprised)
  - User awareness
- Understand and proactively address risk



### 20.34 Cyber Security Budgets Quiz



#### Cyber Security Budgets Quiz

Choose the best answer.

Are cyber security budgets increasing as the number of reported incidents increases (see the instructor notes for a link to the PwC report)?

Yes

No



### 20.35 Cyber Security Budgets Quiz Solution



## Cyber Security Budgets Quiz

Choose the best answer.

Are cyber security budgets increasing as the number of reported incidents increases (see the instructor notes for a link to the PwC report)?

Yes      2014

No



Read more [here](#).

### 20.36 Proactive Security Quiz



#### Proactive Security Quiz

Choose the best answer.

An example of proactive security measure is...

- Making sure the company complies with all regulatory requirements
- Chief risk officer (CRO) of the company addressing cyber risk regularly at highest level (e.g., board) when other risks are discussed

### 20.37 Proactive Security Quiz Solution



**Proactive Security Quiz**  
Choose the best answer.

An example of proactive security measure is...

Making sure the company complies with all regulatory requirements

Chief risk officer (CRO) of the company addressing cyber risk regularly at highest level (e.g., board) when other risks are discussed



## 21 Law, Ethics, and Privacy

### 21.1 US Laws Related to Online Abuse

People can do many different things online. Some of these activities can be very productive while others can be quite harmful. We can examine the various laws in place that offer protection against different types of online abuse.

For example, consider cybercrime. We know that things like theft and extortion are illegal in the physical world. We've discussed similar activities that take place in the digital realm, such as data theft, identity theft, and ransomware. Thankfully, there are laws that offer us protection against cybercrime just as there are laws that protect us from "real" crime.

Additionally, consider creators of digital objects, such as software or digital music. In the non-digital world, we have ways to protect intellectual property with copyrights, patents, and trade secrets. There is a set of laws that control the copying and distribution of digital objects.

It's important to think about how these laws should be applicable in the context of digital objects. For example, a digital object may be copied and transmitted much more easily than a physical object.

A third area we can think about is our privacy as Internet users. We spend a lot of time online, visiting different websites, using different services, and searching for different things. It's very easy for entities that we interact with digitally to collect a lot information about us. A lot of people worry about the concept of Big Brother watching everything they do, and would like some protection.

## 21.2 Legal Deterrents Quiz



### Legal Deterrents Quiz

Technology and other safeguards for cyber security are largely defensive in nature. The only way they can impact a threat source is by increasing the work factor for an attacker. Can laws be used to reduce the magnitude of threats? Choose the best answer:

- Yes, laws can provide criminal sanctions against those who commit cyber crime
- No, cyber crime has increased even as new laws have been put in place.



### 21.3 Legal Deterrents Quiz Solution



#### Legal Deterrents Quiz

Technology and other safeguards for cyber security are largely defensive in nature. The only way they can impact a threat source is by increasing the work factor for an attacker. Can laws be used to reduce the magnitude of threats? Choose the best answer:

- Yes, laws can provide criminal sanctions against those who commit cyber crime
- NO, cyber crime has increased even as new laws have been put in place.

#### 21.4 Cost of Cybercrime Quiz



#### Cost of Cybercrime Quiz

Choose the best answer.

Cyber crime is a big problem. According to a recent report, what is an estimate of the cost of cybercrime for the United States?



- Ten billion dollars
- Over hundred billion dollars

## 21.5 Cost of Cybercrime Quiz Solution



### Cost of Cybercrime Quiz

Choose the best answer.

Cyber crime is a big problem. According to a recent report, what is an estimate of the cost of cybercrime for the United States?

Ten billion dollars

Over hundred billion dollars

.6% of GDP  
~ 17 Trillion dollars

## 21.6 US Computer Fraud and Abuse Act (CFAA)

The first U.S. law related to cybercrime and illegal online activities is the **U.S Computer Fraud and Abuse Act** (CFAA). The goal of this act was to define what types of online actions constitute criminal activity and then outline sanctions against those actions.

The CFAA primarily focuses on unauthorized access to computers and the data contained therein. Attackers who break into systems can alter the confidentiality, integrity, or availability of the data on these systems. Additionally, they can tamper with the integrity and availability of the service itself, rendering it unable to carry out its functions on behalf of its user. Since computer systems and computer data are valuable, we would like to protect them from these kinds of activities.

Initially, the CFAA only covered access to computers containing information related to national security and defense. Later, the CFAA extended its scope to include computer systems containing financial information, as many people care about the safety and security of their money. More recently still, the CFAA broadened its scope even further to cover any **protected computer**: any computer connected to the Internet.

Additionally, the CFAA makes it illegal to distribute malware or other software that allows a user to

gain unauthorized access to a computer system.

Finally, the CFAA addresses passwords specifically. Trafficking in computer passwords - selling stolen passwords on the black market - is illegal as well.

## 21.7 Digital Millennium Copyright Act Part 1

The **Digital Millennium Copyright Act** (DMCA) concerns intellectual property and digital content. The problem it helps to address is **piracy** - the illegal copying and theft of digital objects.

DMCA says that content creators can copyright their digital objects. As a result, these objects can now receive the same protections as copyrighted objects in the non-digital world.

Additionally, DMCA makes it a crime to circumvent or disable any measures concerning copyright protection and anti-piracy. Not only that, but DMCA also makes it a crime to manufacture and distribute any software or device that can disable anti-piracy protections.

## 21.8 Digital Millennium Copyright Act Part 2

The DMCA does have space for some exceptions, primarily for research and educational purposes. For example, if you are researching the strength and security of an anti-piracy measure, you may need to tamper with that measure deliberately. Additionally, libraries can generally make some limited number of copies of digital objects without being considered in violation of the law.

One of the criticisms that these laws face is that they are too vague and broad, which means that different groups can have different interpretations. On one side, parties like the Recording Industry Association of America have filed suits against entities they believe violated the law, while organizations like the Electronic Frontier Foundation worry about the broad scope of the laws and the potential for abuse brought on by the laws themselves.

## 21.9 Computer Abuse Laws Enforcement

While the mere presence of laws like CFAA and DMCA is helpful, to a degree, to deter attackers from committing digital crimes, the enforcement of these laws remains challenging for several reasons.

First, connecting malicious computer activity with an attacker can be difficult. For example, an attack may originate from a user's laptop without that user's knowledge at all. Clearly, the user is not to blame; instead, an attacker likely infected their machine with malware or bot code. Because of maneuvers like this, the process of pinning an attack to its exact source can be quite challenging.

To make matters worse, the Internet has no national boundaries. Attackers that digitally cross borders - hacking into a computer in the United States using a computer in Russia, for example - can only be brought to justice through transnational coordination. Many countries are unable or simply unwilling to cooperate with other countries in this context.

Additionally, lawmaking is a slow process, and the stakes are high for focused attackers. As a result, cybercriminals evolve quickly and continue to work around the laws that we put in place. Technology and creativity proceed more quickly than legislation.

### 21.10 Melissa Virus Quiz



### Melissa Virus Quiz

The Computer Fraud and Abuse Act was used to prosecute the creator of the Melissa virus and he was sentenced in federal prison and fined by using its provisions. What abuse was perpetrated by the Melissa virus? Choose the best answer.

Data stored on computers was destroyed.

Denial-of-service attacks that made computers unusable

### 21.11 Melissa Virus Quiz Solution



#### Melissa Virus Quiz

The Computer Fraud and Abuse Act was used to prosecute the creator of the Melissa virus and he was sentenced in federal prison and fined by using its provisions. What abuse was perpetrated by the Melissa virus? Choose the best answer.

Data stored on computers was destroyed.

Denial-of-service attacks that made computers unusable

## 21.12 Unauthorized Access Quiz



### Unauthorized Access Quiz

Several people have argued about the overly general and vague language of the CFAA. For example, how exactly is unauthorized access defined? In one case, a company sued its competitor because the competitor's employees created a trial subscription and downloaded data that was available to its subscribers. Do you think this is a violation of unauthorized access? Choose the best answer.

- No, the data was publicly available
- Yes, because it potentially can cause financial loss to the company that sued its competition.

### 21.13 Unauthorized Access Quiz Solution



#### Unauthorized Access Quiz

Several people have argued about the overly general and vague language of the CFAA. For example, how exactly is unauthorized access defined? In one case, a company sued its competitor because the competitor's employees created a trial subscription and downloaded data that was available to its subscribers. Do you think this is a violation of unauthorized access? Choose the best answer.



No, the data was publicly available



Yes, because it potentially can cause financial loss to the company that sued its competition.



### 21.14 DMCA Exclusions Quiz



#### DMCA Exclusions Quiz

Choose the best answer.

The DMCA includes exclusions for researchers but companies have threatened to sue researchers who wanted to publish work related to circumvention of anti-piracy technologies. Which of these is an example of such a threat under DMCA?

- Prof. Ed Felten's research on audio watermarking removal by RIAA
- A research project done by MIT students that found vulnerabilities in the Boston Massachusetts Bay Transit Authority (MBTA).



## 21.15 DMCA Exclusions Quiz Solution



### DMCA Exclusions Quiz

Choose the best answer.

The DMCA includes exclusions for researchers but companies have threatened to sue researchers who wanted to publish work related to circumvention of anti-piracy technologies. Which of these is an example of such a threat under DMCA?

Prof. Ed Felten's research on audio watermarking removal by RIAA

A research project done by MIT students that found vulnerabilities in the Boston Massachusetts Bay Transit Authority (MBTA). GAA



## 21.16 Ethical Issues

Laws are manifestations of what we, as a society, determine to be right and wrong. Laws are not subjective on an individual level, and external entities, such as police officers and judges, enforce and arbitrate laws on behalf of society at large.

Ethics are much more personal and help guide our actions based on an internal sense of right and wrong. Something that feels ethical to you might not feel ethical to me. Organizations that we join - ACM, IEEE, or Georgia Tech, for example - may have additional ethical standards or codes of conduct by which they ask us to abide. We stick to our ethics not for fear of going to jail, but rather because they outline what we perceive to be right and appropriate in the world.

Consider finding a vulnerability in a widely-used software product. What is the ethical disclosure of this vulnerability? On the one hand, public disclosure alerts attackers to the vulnerability. On the other hand, deciding not to disclose the vulnerability removes any motivation for a vendor to fix it. That being said, attackers may have discovered the vulnerability already, so a public disclosure may inform users that they should stop using the software, at least until a patch is out.

Situations like this are present throughout the digital world, and handling them with finesse requires

thoughtfulness and a desire to do the right thing.

### 21.17 Computer Ethics Quiz



#### Computer Ethics Quiz

Choose the best answer.

By mistake, a friend sends sensitive health data in an email to you (wrong attachment). You should not read the information in the attached document because...

- Professional code of ethics requires you to respect privacy of others.
- You can be liable under CFAA.

### 21.18 Computer Ethics Quiz Solution



## Computer Ethics Quiz

Choose the best answer.

By mistake, a friend sends sensitive health data in an email to you (wrong attachment). You should not read the information in the attached document because...

Professional code of ethics requires you to respect privacy of others.

You can be liable under CFAA.



### 21.19 Responsible Disclosure Quiz



#### Responsible Disclosure Quiz

Choose the best answer.

US\_CERT follows a responsible disclosure process for vulnerabilities reported to it. Such a process must...

- Make the vulnerability information available to everyone who may be affected by it immediately,
- Provide a certain period of time for the vendor of the vulnerable system to develop a patch.

## 21.20 Responsible Disclosure Quiz Solution



**Responsible Disclosure Quiz**  
Choose the best answer.

US\_CERT follows a responsible disclosure process for vulnerabilities reported to it. Such a process must...

- Make the vulnerability information available to everyone who may be affected by it immediately,
- Provide a certain period of time for the vendor of the vulnerable system to develop a patch.



## 21.21 Privacy Definition

Justice Louis Brandeis made his famous statement about the “right to be let alone” almost 100 years ago; clearly, concerns about privacy are not new. That being said, privacy in the digital era looks different than privacy in the early twentieth century. When we talk about privacy online, we are primarily referring to a user’s ability to control how data about him or her is collected, used, and shared by someone else.

## 21.22 Privacy is Not a New Problem

People have always worried about what others - friends, family, adversaries, the government - know about them, whether it be their whereabouts, activities, or preferences. In the modern, digital context, websites and other software capture, use and distribute our information at an unprecedented scale.

## 21.23 What is Private

**Privacy**

**What is private?**

- Financial statements, credit card statements, banking records etc.
- Health/medical conditions
- Legal matters
- Biometrics (e.g., fingerprints)
- Political beliefs

• School and employer records

• Web browsing habits?  
What do we search,  
what do we browse?  
Websites we visit?

• Communication (emails and calls)

• Past history (right to be forgotten)



## 21.24 What is Not Private

Privacy

What is not private?

- Where I live? My **citizenship**?
- I am **registered to vote**? (US)
- My **salary** (state employee because Georgia Tech is a public university)



## 21.25 Institutional Privacy

Privacy refers to the ability of an individual to control the data about them that is collected and shared. Of course, privacy doesn't have to only apply to individuals. Indeed, there may be reasons that universities, hospitals, charities, and even political action committees may want to keep information private.

## 21.26 Privacy Quiz



### Privacy Quiz

Choose the best answer.

A 2015 Pew surveyed American adults' attitudes about privacy. What percentage feel that it is important that they be able to control who gets information about them? (see instructor notes for link)



- About 50%     About 25%     Over 90%

### 21.27 Privacy Quiz Solution



## Privacy Quiz

Choose the best answer.

A 2015 Pew surveyed American adults' attitudes about privacy. What percentage feel that it is important that they be able to control who gets information about them? (see instructor notes for link)

About 50%     About 25%     Over 90%



Read more [here](#).

## 21.28 Right to Be Forgotten Quiz



### Right to Be Forgotten Quiz

In 2014, the European Court of Justice ruled that EU citizens have the „right to be forgotten“ on the Internet. For example, Google must not return links to information that can be shown to be „inaccurate, inadequate, irrelevant or excessive“. Which one of the following is an example of information that Google decided not to return as a search result to meet the ECJ ruling? Choose the best answer.

- Story about criminal conviction that was quashed on an appeal
- A doctor requesting removal of links to newspaper stories about botched procedures performed by him

## 21.29 Right to Be Forgotten Quiz Solution



### Right to Be Forgotten Quiz

In 2014, the European Court of Justice ruled that EU citizens have the „right to be forgotten“ on the Internet. For example, Google must not return links to information that can be shown to be „inaccurate, inadequate, irrelevant or excessive“. Which one of the following is an example of information that Google decided not to return as a search result to meet the ECJ ruling? Choose the best answer.

Story about criminal conviction that was quashed on an appeal  
 A doctor requesting removal of links to newspaper stories about botched procedures performed by him



## 21.30 Threats to Privacy Part 1

While we have some idea of what types of information we believe should be private in an online context, we should realize that our desires for privacy are not always respected.

For example, when we talk about network requests from, say, a browser to a server, we have to understand that our communication must pass through the greater Internet. While people generally believe that information regarding which websites they visit should be private, traffic sniffing and analysis are threats to that belief. An attacker who observes the packets sent between you and a website can violate the confidentiality of this information.

Additionally, large scale surveillance is a threat to privacy. Some law enforcement officials claim that allowing encryption without providing a backdoor allows criminals to communicate in complete secret, while others argue that we are currently in the golden age of surveillance. Every piece of information about us is tracked and collected.

Finally, it may be possible for others to infer information about us, even if they don't possess it directly. For example, when we talked about database inference attacks, we discussed how different data setups, combined with public information an attacker already has, make it easy to de-anonymize data

about specific individuals. In the world of big data and data mining analytics today, these inferences are becoming more common.

### **21.31 Threats to Privacy Part 2**

Social media poses a threat to privacy, as well. While we use these social sites as a tool for communicating with our friends and sharing with the people around us, we also share all of the information we create with companies like Facebook and Twitter.

Additionally, web tracking serves to reveal information about our browsing habits. Obviously, when we visit a web site, the owner of that site knows that we visited, but the owner can also choose to pass that information along to third parties for analysis. For example, an e-commerce site may let Google know that you visited so that Google can serve you an advertisement for the site later on.

Finally, we enable this behavior. We frequently pass over our information to vendors - whether our name, phone number, or email address - for small discounts in the form of digital coupons and loyalty program memberships. We violate our own privacy, in a sense, by feeding information into a system that we know is designed to exploit us.

### **21.32 Privacy Threats to Online Tracking Info**

One threat to privacy in the context of online tracking concerns information collection. Many of us are aware that certain websites and services collect information about us because most platforms require us to agree to such collection. If companies did not tell us they were collecting information about us but did it regardless, we might consider them to be invading our privacy.

Once a company has our information, we might be concerned with how they use it. Information is powerful, and a company that does not disclose how they use the information they collect might wield it in a way that invades our privacy.

Additionally, we might be curious as to how long a company might keep such information. People change, and information that was valid at one time in the past may not be valid in the present.

Perhaps a company has all the best intentions with our data. We might not be worried about them specifically, but we might want to examine their business partners and other entities with whom they share data. These partners may not use best practices for keeping our data safe from attackers, and the entities that they partner with might be even less professional.

A company may specify all of these factors - what data they collect, what they do with the data, how long they keep the data, and with whom they share the data - in a document known as a **privacy policy**.

Indeed, the privacy policy itself can serve as a threat to privacy. Imagine reading and agreeing to a policy that later is changed to be much more lax concerning data collection, use, and distribution. A privacy policy that can change at any time, to any degree, without subsequent consent, is as bad as no privacy policy at all.

Finally, we also have to consider the security implications of collecting and storing all of this information. Since attackers have no regard for privacy and consent, we can assume that they will treat our data in the worst way possible. Companies need to address how they safeguard our data even in the face of theft.

### **21.33 Example: Google Privacy Policy**

While most websites have privacy policies, few of us honestly look at them carefully, if at all. Let's look at some components of Google's privacy policy.

Some information that Google collects about us is the information that we give it. For example, when we create a new account in Gmail, we likely provide our name, alternative email address, and phone number.

Other information is collected less visibly. When we visit certain websites or use Google services, it can record the actions we perform and the content we consume.

Google uses cookies - a general website-based tracking mechanism - to keep various pieces of state on our browsers as we move about the internet. As a result, Google can collect information about us even when we are not connected to a particular Google service at the moment.

Additionally, Google captures information about the device we use to communicate with their services. For example, when our browser connects to Gmail, it passes along information such as our operating system, our IP address, and network information.

Google's original killer feature was search, and so Google naturally collects and retains your search queries to understand what kind of information you are interested in consuming.

If you use Google Voice, Google has information about who you call and how long you talk. Additionally, some Google services may collect location information.

### **21.34 Google: How is Info Used**

We saw how Google collects information about us, and what type of information it collects, so now we have to ask how Google uses that information. Google has two primary uses for the data it collects about us.

First, it uses the data to personalize our experience with its services. Two random people are likely interested in quite different types of information, and the more information that Google has about them, the more it can tailor an experience to them.

For example, a doctor searching for “lens” might be referring to the part of the eye, while a photographer might be looking for the part of the camera. Without collecting (or inferring) the occupation of both searchers, Google might serve the same search result to two contextually different queries.

Second, Google uses the data to serve better-targeted advertisements. Google makes its money selling ads and, the more it knows about you, the better it can understand which products and services you are likely to buy. This knowledge allows it to charge a higher premium for ad delivery using its network.

### **21.35 Google: Who is it Shared With**

Now that we understand how Google collects and uses our information, we can begin to look at how and with whom Google shares our information.

Google allows us to opt-in to sharing our information with other businesses and services. For example, if we use a “Log in with Google” button on a third party website, Google makes us confirm that we indeed consent to give this third party our information.

Additionally, Google may share information with individuals who provide user support to an organization to which we belong. These individuals include, for example, domain administrators and resellers. Google also shares information with affiliates, which are other trusted businesses or people.

Finally, Google can share the information with the government for legal reasons, and it has improved its transparency in recent years concerning what kind of information it shares with the government or law enforcement.

### **21.36 Google: Information Security**

Our conversation would not be complete without an examination of the security regarding our information and privacy.

Many Google services use HTTPS for encryption of data transmission, as well as encryption for data persistent on their servers. Additionally, Google requires two-factor authentication and makes you enter your password and a token before giving you access to your account on a new device.

Google also provides transparency around changes to its privacy policy. Basically, its stance on privacy policy changes is that you have to explicitly consent to any changes that may reduce your rights

concerning your privacy. In other words, any changes that make the privacy more lax require your agreement before those changes go into effect with your data.

### 21.37 EFF Quiz



#### EFF Quiz

Check all that apply:

The Electronic Frontier Foundation (EFF) ranks websites with privacy scores based on how they deal with issues related to privacy. It gave AT&T one of the lowest scores (just one out of five stars). What explains this low score?

- Does not disclose data retention policies
- Does not use industry best-practices
- Does not tell users about government data demands

### 21.38 EFF Quiz Solution



## EFF Quiz

Check all that apply:

The Electronic Frontier Foundation (EFF) ranks websites with privacy scores based on how they deal with issues related to privacy. It gave AT&T one of the lowest scores (just one out of five stars). What explains this low score?

- Does not disclose data retention policies
- Does not use industry best-practices
- Does not tell users about government data demands



Read more [here](#)

### 21.39 Google Privacy Policy Quiz



## Google Privacy Policy Quiz

Choose the best answer.

Does Google privacy policy disclose data retention policy?



Yes  
 No



### 21.40 Google Privacy Policy Quiz Solution



## Google Privacy Policy Quiz

Choose the best answer.

Does Google privacy policy disclose data retention policy?



Yes

No



### 21.41 Legal Deterrents Quiz



#### Legal Deterrents Quiz

Mark all applicable answers.

Poor privacy is good for bad guys because they can use information about you to craft...

Targeted phishing attacks

Gain access to your online accounts



## 21.42 Legal Deterrents Quiz Solution



The slide features a shield with crossed swords icon on the left. To its right, the title "Legal Deterrents Quiz" is written in purple, and below it, the instruction "Mark all applicable answers." is written in blue. A statement in black text reads: "Poor privacy is good for bad guys because they can use information about you to craft...". Below this statement are two checkboxes. The first checkbox, which has a green checkmark, is followed by the text "Targeted phishing attacks". The second checkbox, which also has a green checkmark, is followed by the text "Gain access to your online accounts". A hand holding a pen is visible on the right side of the slide.

## 21.43 Facebook Privacy Policies

Now that we have discussed what a particular privacy policy might address, we can talk about how well a company adheres to its policy. For this example, we are going to look at Facebook.

Facebook has not done a good job of sticking to its privacy policy. In fact, the U.S. Federal Trade Commission (FTC) had to step in to make sure Facebook stopped doing things with user's information without their consent.

Facebook did numerous things that violated the privacy of its users. For example, it made information that users had designated as private, particularly regarding friend lists, public without their consent.

Additionally, Facebook made personal information available to applications of friends, which means that a friend could access information through an application that a user didn't agree to share.

Furthermore, Facebook shared information with advertisers that they had promised not to share. Finally, Facebook said that certain applications were verified when they were not verified, thus misleading users.

As a result, the FTC has taken action against Facebook and mandated that it acts in specific ways to rectify the situation.

For example, Facebook now has to undergo a third-party privacy audit every two years for the next twenty years. The FTC has decided that it cannot take Facebook's word concerning privacy, and so a third party must independently verify that Facebook is doing the right thing.

Additionally, Facebook is prohibited from misrepresenting privacy and security settings provided to its users. If they say something is not public, it should not be public.

Finally, the FTC also requires that Facebook gets consent from users before sharing any information in a way that exceeds their privacy settings.

Essentially, consent is the focal point when it comes to privacy. If you give express, affirmative consent to a particular action, a company can perform that action. Since Facebook did not get consent before using users' information, the FTC had to impose sanctions on them.

## **21.44 Privacy Enhancing Technologies**

We've talked about how privacy is a good thing, so now let's examine some of the tools and technology we can use to enhance our privacy online.

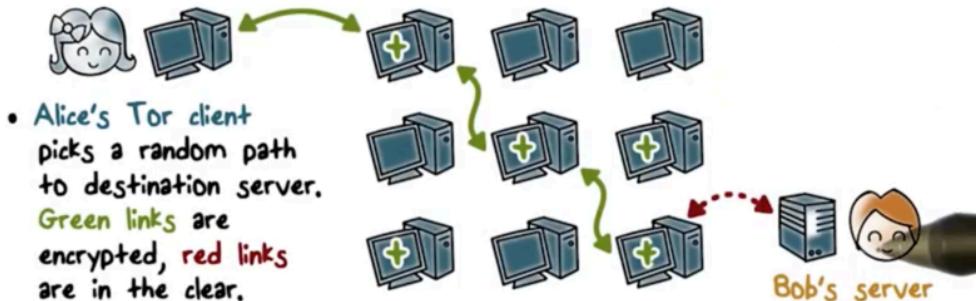
One of the privacy threats that we discussed was traffic analysis. Through traffic analysis, attackers can understand who is talking to whom by looking at the source and destination information embedded in a packet.

If we don't want anyone to see this information, we can use a technology called **TOR**. Using a TOR client, a user can route their traffic through a random path of TOR nodes to their destination. Each node only knows its immediate predecessor and successor and cannot tell where a packet originated or where it is ultimately headed.

## Privacy Enhancing Technologies

**TOR** (network traffic analysis would not allow someone to know where we are coming from)

- Alice does not want web service to know she is accessing it.



Because of this setup, we can use TOR to implement anonymous communication and prevent attackers from learning about the parties with whom we communicate.

### 21.45 TOR

Now that we've taken a brief look at [The Onion Router](#) (TOR) let's dive deeper into how it works and what services it can provide to us.

First, a TOR client uses a directory service to get a set of TOR nodes that can help us route a message from point A to point B. The client picks some random subset of the nodes and orders them. This order forms the path that the traffic traverses as it passes from A to B.

Next, the client encrypts the message as follows. It first encrypts the message with the public key of the final node. Then, it encrypts that result with the public key of the second to last node. It works backward until it finally encrypts the bundle with the public key of the first node.

This strategy results in a message that is encrypted in layers, hence the “onion”. As the traffic passes through the network, each node decrypts its layer, unwrapping the message one decryption at a time. The final node reveals the plaintext packet and forwards it the last hop to its destination.

## 21.46 Controlling Tracking on the Internet

Browsers give us some control with regard to how we are tracked on the Internet. There are a few techniques we can employ to try to reduce the amount of information collected about us.

For instance, we can block third-party cookies. When we visit a site like newyorktimes.com, it can place cookies on our browser. Additionally, it can embed content from other third-party sites, like Facebook, Twitter, and Google, and those sites can also place cookies in our browser. By blocking third-party cookies, we reduce tracking to just the site that we are currently visiting.

Additionally, you can tell your browser that you don't want to be tracked. Many browsers allow you to set flags like this, and they pass that information along to the websites that you visit. Whether the website honors the request is a different story.

You can also delete your cookies periodically, block popups, and use private browsing. Each of these activities helps to reduce the amount of information an entity can capture about you.

## 21.47 Fandango Quiz



### Fandango Quiz

Choose the best answer.

The FTC charged Fandango, the online movie ticket purchasing company, for not protecting user privacy. This action was taken because Fandango...

shared user data without informing users  
 did not secure user data

### 21.48 Fandango Quiz Solution



## Fandango Quiz

Choose the best answer.

The FTC charged Fandango, the online movie ticket purchasing company, for not protecting user privacy. This action was taken because Fandango...

- shared user data without informing users
- did not secure user data

### 21.49 Tracking Quiz



#### Tracking Quiz

Choose the best answer.

If a company tracks your activities based on your machine's IP address. One possible defense against it is...

- Disable cookies
- Use Tor



### 21.50 Tracking Quiz Solution



#### Tracking Quiz

Choose the best answer.

If a company tracks your activities based on your machine's IP address. One possible defense against it is...

Disable cookies

Use Tor