



Classificazione Decimale Dewey:

005.8 (23.) SICUREZZA DEI DATI

GERARDO IOVANE, GERMANO INGENITO

RED TEAM CYBERSECURITY LECTURES

**ANDROID OFFENSIVE OPERATIONS:
THEORY, ENGINEERING AND
RED PREREQUISITES**





©

ISBN
979-12-218-2569-5

PRIMA EDIZIONE
ROMA 9 MARZO 2026

Table of Contents

PART 1	15
1 INTRODUCTION	15
2 RECONNAISSANCE AND INTELLIGENCE GATHERING	36
3 PAYLOAD ENGINEERING AND OBFUSCATION	60
4 DELIVERY MECHANISMS	79
5 5. EXPLOITATION AND EXECUTION	99
6 PERSISTENCE AND ANTI-FORENSICS	122
7 COMMUNICATION, EXFILTRATION AND CONTROL	132
8 CONCLUSION	151
PART 2	155
9 IAM ATTACK ON ANDROID	155
10 CONSOLIDATING LOGIN ON ANDROID (NEARBY SHARE INJECTION)	160
11 PERSISTENCE ON ANDROID DEVICE	164
12 STEALTH AND ANTI-FORENSICS ON ANDROID	169
13 EXFILTRATION OR SABOTAGE ON ANDROID	174
14 REVERSIBLE C4I INTERACTION ON ANDROID	178
15 ADVANCED MULTI-CHANNEL EXFILTRATION ON ANDROID	182
16 SECONDARY PAYLOAD DROPPER ON ANDROID	188
17 ENVIRONMENTAL SURVEILLANCE & KEYLOGGER ON ANDROID	192
18 OTA UPDATE MALICIOUS ON ANDROID	196
19 CAMOUFLAGE COMMUNICATION LAYER ON ANDROID	200
20 ANTI-FORENSICS MODULE SU ANDROID	203
21 OPPORTUNISTIC ROOT ESCALATION ON ANDROID	207
22 ACCESSIBILITY SERVICE EXPLOITATION ON ANDROID	210
23 KERNEL EXPLOIT DETECTOR ON ANDROID	213
24 PERSISTENCE IN BOOTLOADER/RECOVERY ON ANDROID	217

25 C2 COMMUNICATION OVER DNS TUNNEL ON ANDROID.....	221
26 EXPLOIT CHAIN HANDLER ON ANDROID	225
27 WI-FI STACK OPPORTUNISTIC SNIFFING ON ANDROID	228
28 MOBILE NETWORK METADATA COLLECTION ON ANDROID	231
29 REMOTE MICROPHONE ACTIVATION ON ANDROID	235
30 REMOTE CAMERA ACTIVATION ON ANDROID.....	239
31 REMOTE FILE ACCESS & EXFILTRATION ON ANDROID	244
32 STEALTH PERSISTENCE MECHANISMS ON ANDROID	248
33 COMMAND & CONTROL INTERACTION (C2 LIGHTWEIGHT PROTOCOL) ON ANDROID	251
34 SELF-DESTRUCT AND CLEAN-UP MODULE ON ANDROID	255
35 SECURE NETWORK STEALTH TUNNELING ON ANDROID.....	258
36 ANTI-FORENSICS AND ANTI-ANALYSIS ON ANDROID	262
37 DATA OBFUSCATION AND ENCRYPTED STORAGE ON ANDROID.....	266
38 PERSISTENCE REINFORCEMENT TECHNIQUES ON ANDROID	269
39 STEALTH SERVICE CLOAKING ON ANDROID.....	272
40 SELF-UPDATE AND REMOTE RECONFIGURATION ON ANDROID.....	276
41 C2 ADAPTIVE COMMUNICATION ENGINE ON ANDROID	280
42 SELF-DESTRUCTION AND WIPEOUT MECHANISMS ON ANDROID.....	284
43 ANTI-FORENSICS AND TRACE ERASURE ON ANDROID.....	288
44 PERSISTENCE VIA HIDDEN BOOT RECEIVERS ON ANDROID.....	292
45 OBFUSCATION AND EVASION TECHNIQUES ON ANDROID.....	295
46 HIDDEN UPDATES AND REMOTE MODULE INJECTION ON ANDROID	298
47 DYNAMIC NETWORK PROXY AND C2 SWITCHING ON ANDROID.....	302
48 STEALTH FILE DROPPER AND AUTO-EXECUTE ON ANDROID	305
49 DYNAMIC PERMISSION ESCALATION ON ANDROID	309
50 PHASE 42 – PERSISTENCE THROUGH ACCESSIBILITY SERVICES ON ANDROID	312
51 ENCRYPTED DATA EXFILTRATION OVER WEBSOCKETS ON ANDROID.....	315
52 REMOTE PAYLOAD UPDATE VIA WEBSOCKET ON ANDROID	321
53 DYNAMIC COMMAND EXECUTION ENGINE ON ANDROID.....	322
54 COVERT COMMUNICATION VIA PUSH NOTIFICATIONS ON ANDROID	325

55 DEPLOY INITIAL STEALTH ON ANDROID328

56 AUTOLOADING THE STEALTH PAYLOAD ON ANDROID.....331

57 EXTENDED FINGERPRINTING AND ADVANCED SYSTEM DATA COLLECTION ON ANDROID334

58 ADVANCED PERSISTENCE VIA SERVICE ON ANDROID.....338

59 DYNAMIC ANTI-ANALYSIS AND ANTI-DEBUGGING ON ANDROID342

60 STEALTH PAYLOAD AUTO-UPDATE ON ANDROID.....346

61 STEALTH DATA EXFILTRATION FROM ANDROID VIA MASKED HTTP350

62 REAL-TIME MONITORING OF CRITICAL EVENTS ON ANDROID354

63 STEALTH HARVESTING AND EXFILTRATION OF CONTACTS, SMS AND CALL LOGS ON ANDROID.....358

64 SILENT COLLECTION AND EXFILTRATION OF FILES OF INTEREST ON ANDROID363

65 ENCRYPTING COLLECTED FILES BEFORE EXFILTRATION ON ANDROID.....367

66 CREATING A SENSITIVE FILE AUTO-DELETION MODULE ON ANDROID370

67 CREATING A C2 STEALTH COMMUNICATION MODULE VIA PUSH NOTIFICATION ON ANDROID..... 373

68 IMPLEMENTING A HIDDEN PERSISTENCE MODULE VIA BROADCASTRECEIVER ON ANDROID.....377

69 CREATING AN ANTI-FORENSICS MODULE ON ANDROID..... 379

70 CREATING A FILE AND DIRECTORY OBFUSCATION MODULE ON ANDROID383

71 CREATING A TRAFFIC RANDOMIZATION MODULE FOR MASKING ON ANDROID.....386

72 CREATING A STEALTH NETWORK SCANNER MODULE ON ANDROID.....389

73 CREATING A STEALTH KEYLOGGER MODULE ON ANDROID.....392

74 CREATING A STEALTH SCREEN CAPTURE MODULE ON ANDROID.....395

75 CREATING AN AUTO-UPDATE PAYLOAD MODULE ON ANDROID398

76 CREATING A PAYLOAD AUTO-HEALING MODULE ON ANDROID.....402

77 CREATING A STEALTH PAYLOAD AUTO-REMOVAL MODULE ON ANDROID.....405

78 CREATING AN ADVANCED PERSISTENCE MODULE ON ANDROID.....408

79 CREATING A KEYLOGGER MODULE VIA ACCESSIBILITY SERVICES ON ANDROID411

80 CREATING A NETWORK AUTO-CONFIGURATION MODULE ON ANDROID414

81 CREATING A STEALTH DATA EXFILTRATION MODULE ON ANDROID417

82 CREATING A PAYLOAD SELF-UPDATE MODULE ON ANDROID.....	420
83 CREATING A SECURE DATA ERASURE FORM ON ANDROID	424
84 CREATING A STEALTH AUTO-UNINSTALL MODULE ON ANDROID	428
85 CREATING A TIMED SELF-DESTRUCTION MODULE ON ANDROID.....	432
86 CREATING A SELF-DESTRUCTION REMOTE TRIGGER MODULE ON ANDROID.....	436
87 DYNAMIC PAYLOAD GENERATION AND DEPLOYMENT ON ANDROID	440
88 PAYLOAD CONTROL ESCALATION ON ANDROID	444
89 DEEP SABOTAGE SYSTEM OPERATIONS ON ANDROID.....	448
90 ADVANCED ANTI-TAMPERING AND SELF-PROTECTION ON ANDROID.....	451
91 HARDENING COUNTERMEASURES BYPASS ON ANDROID	455
92 MEMORY SCRAMBLING ANTI-FORENSICS ON ANDROID.....	459
93 SECURE ERASE ON-DEMAND ON ANDROID.....	462
94 ANTI-SCAN & CLOAKING SYSTEM INOCULATION ON ANDROID	466
95 DYNAMIC DECEPTION AND DECOY DEPLOYMENT ON ANDROID	469
96 SELF-DESTRUCTION TRIGGER VIA REMOTE COMMAND ON ANDROID.....	472
97 SECURE REMOTE UPDATE MECHANISM ON ANDROID	476
98 ADVANCED RECON MODULE ON ANDROID	480
99 PHYSICAL ACCESS MODULE ON ANDROID	484
100 CLOUD ACCOUNT TOKEN THEFT ON ANDROID.....	488
101 BIOMETRIC SPOOFING FRAMEWORK ON ANDROID	491
102 SMART SELF-HEALING AND EVASION ON ANDROID.....	494
103 FULL MULTI-STAGE ATTACK ORCHESTRATION ON ANDROID.....	498
104 REMOTE COMMAND-AND-CONTROL CHANNEL ON ANDROID	502
105 DYNAMIC PAYLOAD UPDATE MECHANISM ON ANDROID.....	506
106 ANTI-FORENSICS ENGINE ON ANDROID.....	510
107 ENVIRONMENTAL AWARENESS ENGINE ON ANDROID	514
108 MULTI-LAYER ADAPTIVE STEALTH ENGINE ON ANDROID	518
109 AUTONOMOUS CODE MUTATION ENGINE ON ANDROID	522
110 OFFLINE COMMAND QUEUE PROCESSOR ON ANDROID.....	526
111 MULTI-HOP RELAY NETWORKING ON ANDROID.....	530

112 AI-DRIVEN STEALTH STRATEGY OPTIMIZER ON ANDROID534

113 QUANTUM-RESILIENT ENCRYPTION LAYER ON ANDROID538

114 MODULAR EXFILTRATION FRAMEWORK ON ANDROID542

115 IN-MEMORY DATA PERSISTENCE MODULE ON ANDROID546

116 DYNAMIC EVASION TECHNIQUE ENGINE ON ANDROID550

117 ADVANCED STEALTH ROOTKIT LAYER ON ANDROID.....554

118 SELF-HEALING AND AUTO-UPDATE MECHANISM ON ANDROID.....557

119 INTELLIGENT C2 COMMUNICATION PROTOCOL ON ANDROID561

120 ADAPTIVE ANTI-FORENSICS SUITE ON ANDROID565

121 DYNAMIC EVASION TACTICS ENGINE ON ANDROID569

122 ROOTKIT-LEVEL STEALTH MODE ON ANDROID572

123 SELF-HEALING PERSISTENCE LAYER ON ANDROID.....574

124 SMART COMMAND & CONTROL CHANNEL ON ANDROID579

125 ADVANCED PAYLOAD ENCRYPTION SYSTEM ON ANDROID.....583

126 COVERT LOCAL STORAGE CHANNEL ON ANDROID.....588

127 NETWORK ACTIVITY CAMOUFLAGE MODULE ON ANDROID592

128 STEALTH UPDATE MECHANISM ON ANDROID596

129 DYNAMIC C2 SWITCHING MODULE ON ANDROID600

130 ADAPTIVE PERSISTENCE ENGINE ON ANDROID604

131 ADVERSARIAL REPROGRAMMING ENGINE ON ANDROID608

132 MULTI-SANDBOX EVASION MODULE ON ANDROID612

133 COVERT DATA SMUGGLING MODULE ON ANDROID.....615

134 DISTRIBUTED REDUNDANCY COMMUNICATION SYSTEM ON ANDROID618

135 TEMPORAL SHIFTING ATTACK MODULE ON ANDROID.....621

136 NETWORK WATERMARKING AND OBFUSCATION ON ANDROID624

137 QUANTUM-INSPIRED KEY SCHEDULING DISRUPTION ON ANDROID627

138 ADAPTIVE PAYLOAD POLYMORPHISM ON ANDROID630

139 COMMAND TIMING RANDOMIZER ON ANDROID.....633

140 ENVIRONMENTAL FINGERPRINTING ENHANCER ON ANDROID.....636

141 DYNAMIC SENSOR TRIGGER MODULE ON ANDROID640

142 PROXIMITY-BASED TRIGGER ACTIVATION ON ANDROID	644
143 LIGHT SENSOR BEHAVIORAL MODULE ON ANDROID	646
144 APPLICATION USAGE BEHAVIORAL TRIGGER ON ANDROID.....	651
145 ADVERSARIAL BEHAVIOR DETECTION BYPASS ON ANDROID	654
146 DYNAMIC COMMAND AND CONTROL (C2) SWITCHING ON ANDROID	658
147 SECURE KEY DERIVATION FOR SESSION ENCRYPTION ON ANDROID	662
148 STEALTH FILE EXFILTRATION MODULE ON ANDROID	665
149 BACKGROUND DATA SYNCHRONIZATION SPOOFING ON ANDROID.....	669
150 SECURE PAYLOAD DELIVERY MODULE ON ANDROID	673
151 SENSOR DATA INTERCEPTOR ON ANDROID.....	676
152 BROWSER ACTIVITY LOGGER ON ANDROID.....	681
153 CLIPBOARD SNIFFER ON ANDROID	685
154 STEALTH AUDIO CAPTURE ON ANDROID	688
155 ACCESSING CONTACTS ON ANDROID	692
156 SMS/MMS EXTRACTION ON ANDROID	696
157 ACCESSING THE MEDIA GALLERY ON ANDROID	698
158 MONITOR APPS INSTALLED AND UPDATED ON ANDROID	702
159 ADVANCED KEYLOGGING ON ANDROID	705
160 PERIODICAL SCREENSHOT CAPTURE ON ANDROID	708
161 ANTI-WIPE MODULE (ANDROID REMOTE FORMAT PREVENTION).....	713
162 INVISIBLE KEYLOGGER MODULE ON ANDROID.....	716
163 ADVANCED PERSISTENCE MODULE VIA DEVICE ADMIN	719
164 REAL-TIME SMS EXFILTRATION MODULE	722
165 CALL HISTORY CAPTURE MODULE	725
166 BLUETOOTH INFORMATION COLLECTION MODULE	729
167 INSTALLED APP HISTORY EXTRACTION MODULE	733
168 VPN DETECTION MODULE IN USE	737
169 TOR CONNECTION DETECTION MODULE	740
170 AIRPLANE MODE RESILIENCE MODULE.....	743
171 PERSISTENCE RECOVERY MODULE FROM SYSTEM WIPE.....	747

172 PERSISTENT CLOUD BACKUP HOOKUP MODULE.....751

173 PERSISTENT CLOUD BACKUP HOOKUP MODULE.....755

174 ANDROID KERNEL ADVANCED FUZZING MODULE (STAGE 1: SYSCALL SURFACE)..... 758

175 ANDROID KERNEL ADVANCED FUZZING MODULE (STAGE 2: IOCTL INTERFACE).....762

176 ANDROID KERNEL ADVANCED FUZZING MODULE (STAGE 3: BINDER INTERFACE).....765

177 RECENT ANDROID CVE EXPLOITATION MODULE (AUTO-DEPLOY)769

178 MODULE EXPLOIT CVE-2023-20963 (PRIVILEGE ESCALATION773

179 MODULE EXPLOIT CVE-2023-2136 (BUFFER OVERFLOW SKIA LIBRARY).....776

180 MODULO EXPLOIT CVE-2023-21674 (ZERO-CLICK MESSAGING EXPLOIT)779

181 AUTOMATIC MULTI-EXPLOIT CHAINING MODULE (CVE COMBINATION ATTACK)782

182 ANDROID POST-EXPLOITATION PERSISTENCE MODULE (ROOTKIT STEALTH INSTALLER)785

183 ANDROID CLEAN-UP AND ANTI-FORENSICS MODULE (SELF-DESTRUCTION MODE)789

184 C2 STEALTH COMMUNICATION MODULE ON ANDROID (HIDDEN BEACONING SYSTEM)793

185 DYNAMIC PAYLOAD UPDATE MODULE C2 (SELF-UPDATING AGENT)796

186 PAYLOAD DOWNLOAD AND EXECUTE MODULE ON ANDROID (ON-DEMAND EXECUTION ENGINE)..... 800

187 DATA EXFILTRATION STEALTH MODULE ON ANDROID (COVERT CHANNEL EXFILTRATION)804

188 ANTI-DETECTION SYSTEM STATUS SPOOFING MODULE (FAKE ENVIRONMENT SIMULATION).....811

189 ANTI-ANALYSIS ENVIRONMENT DETECTION BYPASS MODULE (SANDBOX EVASION ENGINE).....814

190 DYNAMIC MEMORY INJECTION MODULE.....819

191 FILELESS MALWARE EXECUTION MODULE ON ANDROID (IN-MEMORY EXECUTION ENGINE)..... 819

192 C2 COMMUNICATION OVER LEGITIMATE APPS CHANNELS MODULE (STEALTH C2 OVER SIGNAL/TELEGRAM/WHATSAPP).....823

193 FILE VAULT AND PROTECTOR MODULE ON ANDROID (SECURE HIDDEN CONTAINER ENGINE).....827

194 ADVANCED KEYLOGGER MODULE FOR ANDROID (STEALTH INPUT CAPTURE ENGINE)	832
195 SCREENSHOT STEALTH CAPTURE ANDROID MODULE (SCREENSPY ENGINE)	836
196 CAMERA CAPTURE STEALTH ENGINE ANDROID MODULE (STEALTHCAM ENGINE)	840
197 ANDROID STEALTH AUDIO CAPTURE MODULE (STEALTHMICROPHONE ENGINE)	843
198 DATA EXFILTRATION ENGINE MODULE VIA COVERT CHANNELS	846
199 ROOT DETECTION AND ANTI-FORENSICS ENGINE MODULE	849
200 SELF-UPDATE & SELF-HEALING ENGINE MODULE	852
201 SECURE C2 COMMUNICATION ENGINE MODULE	855
202 ANTI-REVERSE ENGINEERING & OBFUSCATION ENGINE MODULE	858
203 SECURE FILE VAULT MANAGEMENT ENGINE MODULE	861
204 STEALTH PERSISTENCE & RECOVERY ENGINE MODULE	864
205 EVACUATION & CLEANUP ENGINE MODULE	867
206 INTELLIGENCE GATHERING ENGINE MODULE	870
207 EXFILTRATION & CHANNELING ENGINE MODULE	874
208 FINAL SELF-DESTRUCTION & SANITATION MODULE	877
209 EVASION MODULE STATIC RECOGNITION STORE VALIDATION	880
210 PERSISTENCE MODULE ON 5G INFRASTRUCTURES	883
211 EXFILTRATION MODULE VIA LOCAL NAS API	886
212 BLUETOOTH PERMISSIONS ESCALATION MODULE	889
213 AGGRESSIVE PERSISTENCE MODULE VIA PUSH NOTIFICATION ABUSE	892
214 KEYLOGGER MODULE ON CUSTOM VIRTUAL KEYBOARDS	895
215 MONITORING SENSOR DATA STEALTH MODULE	898
216 CLOUD SHADOW COPY PERSISTENCE MODULE	901
217 MFA TOKEN MINING MODULE	904
218 ARTIFICIAL INTELLIGENCE ANTIMALWARE AVOIDANCE MODULE	907
219 STEALTH DNS TUNNELING MODULE	910
220 PERSISTENCE VIA OBFUSCATED SCHEDULED TASKS MODULE (WINDOWS)	913
221 PAYLOAD OBFUSCATION MODULE VIA COMPILATION ON TARGET	916
222 BYPASS WINDOWS DEFENDER MODULE VIA LIVING-OFF-THE-LAND BINARIES (LOLBINS)	919

223 PERSISTENCE MODULE VIA REGISTRY RUN KEYS STEALTH	921
224 DATA EXFILTRATION MODULE VIA HTTP COVERT CHANNELS	924
225 CREDENTIAL DUMPING MODULE FROM LSASS WITH STEALTH TECHNIQUES	927
226 FILELESS EXECUTION MODULE VIA POWERSHELL REFLECTIVE LOADING	931
227 STEALTH PERSISTENCE MODULE VIA WMI EVENT SUBSCRIPTION.....	934
228 DLL INJECTION MODULE VIA REMOTE THREAD	937
229 STEALTH KEYLOGGING MODULE WITH SYSTEM HOOKS	941
230 PERSISTENCE MODULE VIA SCHEDULED TASK STEALTH.....	944
231 PERSISTENT REVERSE SHELL MODULE VIA NAMED PIPE	947
232 CREDENTIAL HARVESTING MODULE VIA BROWSER EXTRACTION	951
233 NETWORK SNIFFER STEALTH MODULE VIA RAW SOCKET.....	955
234 PRIVILEGE ESCALATION MODULE VIA UAC BYPASS FODHELPER.....	958
235 WMI EVENT SUBSCRIPTION PERSISTENCE MODULE	961
236 POWERSHELL PAYLOAD LOADER STEALTH MODULE	964
237 MODULO DNS TUNNELING CLIENT STEALTH.....	967
238 USB AUTO-EXECUTION DROPPER MODULE	970
239 REVERSE SHELL PERSISTENCE MODULE VIA SCHEDULED TASK.....	972
240 CREDENTIAL EXTRACTION MODULE FROM CHROME/EDGE (WINDOWS)	976
241 PERSISTENCE MODULE VIA REGISTRY RUN KEY	979
242 DLL SIDELADING ATTACK FRAMEWORK MODULE	982
243 PYTHON KEYLOGGER STEALTH MODULE VIA CLIPBOARD HOOK.....	985
244 FILE EXFILTRATION MODULE VIA HTTP POST STEALTH	988
245 POWERSHELL FILELESS PERSISTENCE MODULE VIA WMI EVENT SUBSCRIPTION....	991
246 POWERSHELL AMSI BYPASS IN-MEMORY MODULE.....	994
247 TOKEN IMPERSONATION MODULE VIA WINDOWS API.....	997
248 REVERSE TCP SHELL MODULE IN PYTHON (CROSS-PLATFORM).....	1001
249 STEALTH REMOTE SCREENSHOT MODULE IN PYTHON	1004
250 REVERSE SHELL MODULE VIA DNS TUNNELING.....	1007
251 WEB SHELL FORM VIA HTTP POST	1110
252 DNS SPOOFING MODULE FOR DATA EXFILTRATION	1013

253 HTTP HEADER INJECTION DATA EXFILTRATION MODULE	1016
254 FTP DATA EXFILTRATION MODULE	1019
255 KEYLOGGER MODULE VIA USB HID (HUMAN INTERFACE DEVICE)	1022
256 DATA EXFILTRATION MODULE VIA CLOUD STORAGE	1025
257 WEBSOCKET DATA EXFILTRATION MODULE	1028
258 DATA EXFILTRATION MODULE VIA SMB (SERVER MESSAGE BLOCK).....	1031
259 EMAIL DATA EXFILTRATION MODULE (SMTP)	1034
260 HTTP/HTTPS POST DATA EXFILTRATION MODULE	1037
261 DNS (DOMAIN NAME SYSTEM) DATA EXFILTRATION MODULE	1040
262 ICMP (INTERNET CONTROL MESSAGE PROTOCOL) DATA EXFILTRATION MODULE..	1043
263 FTP (FILE TRANSFER PROTOCOL) DATA EXFILTRATION MODULE	1046
264 TELNET DATA EXFILTRATION MODULE	1049
265 WEBSOCKET DATA EXFILTRATION MODULE	1051
266 SMTP (SIMPLE MAIL TRANSFER PROTOCOL) DATA EXFILTRATION MODULE	1054
267 HTTP(S) DATA EXFILTRATION MODULE.....	1057
268 DNS (DOMAIN NAME SYSTEM) DATA EXFILTRATION MODULE	1060
269 ICMP (INTERNET CONTROL MESSAGE PROTOCOL) DATA EXFILTRATION MODULE.	1063
270 DATA EXFILTRATION MODULE VIA HTTP(S) GET	1066
271 WEBSOCKET DATA EXFILTRATION MODULE	1068
272 FTP (FILE TRANSFER PROTOCOL) DATA EXFILTRATION MODULE	1071
273 SMTP DATA EXFILTRATION MODULE (SIMPLE MAIL TRANSFER PROTOCOL).....	1074
274 DNS (DOMAIN NAME SYSTEM) DATA EXFILTRATION MODULE	1077
275 ICMP (INTERNET CONTROL MESSAGE PROTOCOL) DATA EXFILTRATION MODULE.	1080
276 HTTP(S) DATA EXFILTRATION MODULE.....	1083
277 WEBSOCKET DATA EXFILTRATION MODULE	1086
278 BIBLIOGRAPHY	1089

Part 1

1 Introduction

The emergence of mobile devices has revolutionized digital interaction, with Android capturing over 70% of the mobile OS market share in 2023. Consequently, Android has become a popular target for threat actors, facilitated by its open-source nature and vast ecosystem. Offensive attacks against Android systems are diverse and include malware, spyware, zero-click exploitation, and multi-phase attack chains that abuse Android's modularity and permission management. "Android Offensive Operations: Theory, Engineering and Red Team Practice" aims to provide a formal investigation into the methodologies and techniques applied in offensive operations for Android systems. The primary question to be addressed is: How can offensive operations targeting Android systems be systematically modeled, engineered, and simulated within a red team framework, and what methodologies, tools, and payload structures are most effective for achieving persistence, stealth, and data exfiltration in real-world Android environments? The discipline of offensive cybersecurity targeting Android systems has become a focus area in computer science, particularly due to the burgeoning of mobile-based malware and advanced anti-forensic tactics. There have been substantial publications and reports documenting the increasing volume and sophistication of Android-based malware samples, including novel repackaging techniques, obfuscation schemes, and stealth surveillance capabilities. There is a growing necessity to simulate offensive operations to thoroughly understand attack surfaces. Red team testing, payload engineering, and attack simulation can provide a pathway to better understand the limitations of Android and enable more robust security implementations. This paper aims to thoroughly explore the principles and organization of offensive cybersecurity on Android systems, providing a detailed reference that combines a comprehensive treatment of foundational concepts and advanced technical modules. The goal of this document is to (1) introduce the method of red teaming on Android devices, including reconnaissance, payload building, and software architecture, (2) present a framework to execute multi-phase attack chains through the implementation of custom Android payloads, and (3) introduce advanced methods for zero-click payload delivery, rootless persistence, anti-forensics, covert surveillance, and dynamic control. The combination of conceptual discussion, examples, and simulations serves to fulfill the analytical and operational needs of both researchers and practitioners alike. The methodology of this document combines literature review, historical analysis, comparative analysis, and source criticism to explore different attack

methods, techniques, and available tools. Real attacks are used to simulate different Android payloads, allowing for dynamic analysis and concrete, scenario-based examples.

While Android offensive cybersecurity has made progress in simulating adversarial tactics, (i.e., MITRE ATT&CK) and detection/prevention approaches, knowledge gaps still exist within red teaming, offensive simulation, and multi-phase attack chains. Several related works either introduce single techniques or exclusively address detection. No publications discuss offensive simulations and how they can be used to model more realistic adversarial behavior, particularly in a red team setting.

This work provides a combination of technical depth and simulation-based demonstrations to bridge the knowledge gap. The introductory part introduces the motivation behind the project and describes the central research question and goals. The following chapters explain the process of reconnaissance and intelligence gathering, focusing on OSINT and metadata profiling. Then, it delves into advanced topics, such as payload engineering and obfuscation, which is followed by file-based delivery, proximity-based delivery, and C2-initiated delivery. Afterward, exploitation and execution are described, including trigger mechanisms, API abuse, background execution, and hiding the payload. Next, a chapter is focused on Android payload persistence and anti-forensics. Finally, the last chapter delves into the communication, data exfiltration, and control aspects that are important to Android red team operations.

1.1 Android OS Security Architecture

Android's permission-based security model is a basic principle of its architecture. It is meant to enforce fine-grained access control over system resources and user data. Each application must explicitly declare the permissions that it requires in the XML manifest file. Permissions are granted to limit access to resources such as the camera, location, or contacts. This limitation is system-wide enforced. However, user behavior plays a big role in defeating this security model, as users approve permission grants without careful assessment. According to Ahmed and Sallow (2017, p. 3) and Enck et al. (2009, p. 5), this behavioral vulnerability exposes Android devices to information leakage and malicious code exploitation. The permission-based security model is most favorable to red teams and attackers because of this vulnerability.

In Android, each application is assigned to a unique UID to isolate applications, so direct inter-application access is not allowed. Isolation is enforced by the operating system's sandboxing. Application isolation

can be bypassed using Android permissions and/or inter-process communication. Only applications that share the same UID can communicate with each other directly or access another's data. The theoretical application isolation proposed by Enck et al. (2009, p. 1) is compromised through enforcement flaws, incorrect permissions, and/or attack strategy. Application-level privilege escalation can be obtained by repackaging or through IPC and application-level vulnerabilities.

Originally, Android's permission control system was designed with defense-in-depth in mind. All system resources are controlled through permissions granted in the XML manifest file. Permissions control acts as a binding contract that regulates application-OS interaction. Security inconsistencies between the granular application controls have led to possible permission-related attack scenarios. As per Ahmed and Sallow (2017, p. 3) and Enck et al. (2009, p. 5), Android developers can improperly configure permissions. If these loopholes are found by a red team, an attacker may utilize permission flaws to propagate attacks. The Dalvik Virtual Machine (DVM) is a component of the software stack which operates as a just-in-time interpreter that is designed and optimized for mobile devices. According to Ahmed and Sallow (2017, p. 2), applications are isolated in the DVM using sandboxing. As a result, only the compromised application's environment is at risk of malicious activity. However, in practice, applications make many calls to system APIs for the most basic functionality. Enck et al. (2009, p. 1) suggests that system API calls can be abused for covert activities. For example, activities in one Android application can start another, but only if certain IPC criteria are met. Intents, content providers, and broadcast receivers are the main IPC mechanisms. Broadcast intents have several purposes. However, they have security implications because they can be sent by and delivered to any applications. As a result, malicious applications may intercept sensitive system information.

The system APIs and middleware introduce many risk points to Android because the orchestration of inter-component communication is dependent on them. As reported by Ahmed and Sallow (2017, p. 2), it is possible to modify API implementation for permission control. An adversary may make modifications that allows them to retrieve restricted data or escalate privileges. An API is a dynamic attack surface because attackers may discover novel payload injections at various times, as reported by Ahmed and Sallow (2017, p. 2). Theoretical intrusion prevention tools fall behind against sophisticated attack techniques that an adversary can use during payload delivery. Due to this, it is essential that system designers test defense strategies against potential threats.

Attackers can bypass the operating system-level sandboxing in ways that theoretical intrusion prevention cannot account for. Malicious code can be hidden and added at runtime, allowing it to bypass strict checks.

An attacker may be able to use dynamic code loading, dynamic obfuscation, and native code in APK files to evade code control limitations. According to Ahmed and Sallow (2017, p. 2), an Android device is not as isolated as it can be because isolation levels can be compromised through exploiting shared system resources, improper permissions, and incorrectly configured services. During a red team, if an attacker wants to bypass isolation for reconnaissance and penetration, he/she may use obfuscated code in Android's native library. By doing so, it makes detection, debugging, and inspection difficult for system security measures.

Repackaging Android applications is an all too prevalent attack in this mobile security ecosystem. Attackers can download a legitimate application's APK, decompile it, add malicious payload code, and then repackage it and re-upload it to either the app store or to third party repositories for distribution. According to Ahmed and Sallow (2017, p. 3) and Verma (2024, p. 1), the open source nature of Android applications makes repackaging quite easy and widespread. Vidas et al. (2011, p. 6) report that in 2 hours of observation, 3.5 repackaged versions are available for distribution, each of which has approximately 300 downloads.

The openness of the Android ecosystem is not only a source of rapid application growth but also a primary reason for the prevalence of Android-based malware. According to Verma (2024, p. 1), adversaries can analyze and understand Android source code much faster than a software developer or defender. The source code and internal algorithms allow attackers to design sophisticated payload techniques to either retrieve sensitive data or evade security protocols. Android's rapid ecosystem expansion has resulted in tremendous growth of fragmentation. As a result, red teams need to test across many devices and custom ROMs that can have varying security protocols installed. Many custom ROMs can also include undocumented application or API versions that can weaken security defenses, as noted by Caturano (2021, p. 106). A device that installs a custom ROM can install code or applications that can bypass normal system protections. Custom ROM developers can also install firmware that weakens many traditional countermeasures. Adversaries may exploit these variants in custom ROMs to deliver effective attack models.

Over time, tools have been developed to identify possible malware contained within repackaged applications. Both static and dynamic approaches are utilized in malicious code analysis. Static tools such as MalDozer and RevealDroid use API-based feature vectors, artificial intelligence and machine learning to classify applications as malicious or legitimate with an accuracy rate of up to 97.2%, as reported by Bacci et al. (2018, p. 2). Static analysis can achieve good results when not up against strong obfuscation

strategies. Dynamic analyzers such as Alterdroid simulate Android application execution. Alterdroid's analysis achieves close to a 99% detection rate for Android malware, Bacci et al. (2018, p. 2), but does so with a specific corpus of popular and prevalent malware samples. Many newer and more advanced dynamic application analysis technologies are being developed, and one commonality they all have is that they are no match for more modern and complicated obfuscation strategies.

A broader design problem for the Android ecosystem is balancing usability against security for Android device users. Users can be easily manipulated to bypass security barriers if an application request appears innocent. It is important to have end users in charge of access policies, but as Ahmed and Sallow (2017, p. 3) suggests, they are sometimes irresponsible when asked for sensitive permissions. They may blindly approve resource permission requests to install the most harmless of applications. Due to this behavioral loophole, red teams always take advantage of user error in attack model execution. By combining the weaknesses of architectural and implementation flaws with user manipulation, a realistic and effective attack model can be created for red teaming, confirming the findings reported by Caturano (2021, p. 5).

1.2 Threat Landscape and Actors

The threat landscape for Android devices is significantly affected by its immense market adoption and various types of attackers. Boasting over 70% of the smartphone market, Android devices have become a dominant target for attackers ranging from organized cybercriminals to nation states conducting advanced persistent threats. As Vidas et al. (2011, p. 1) and Ahmed and Sallow (2017, p. 1) argue, Android's dominance presents an ideal opportunity for attackers to exploit a largely homogenous landscape, allowing them to create sophisticated, highly scalable attacks with minimal investment. With over four billion downloads, Android's central marketplace creates a platform for both efficiency and scale, thus increasing its attackability (Vidas et al. 2011, p. 1). The ecosystem's scalability is therefore an advantage for attackers who wish to execute attacks across a vast landscape of similar device models.

The ease with which a malicious app can gain legitimacy contributes to opportunistic attacks against Android devices. In the app ecosystem, attackers publish seemingly benign applications on the marketplaces to attain a high rate of downloads from users. For instance, an example app of Vidas et al. (2011, p. 6) accumulated over 200 downloads in just 24 hours after publishing. The effort that social engineering plays as a means for distribution is paramount in opportunistic attacks, as well as the

exploitation of the implicit trust users hold within the application ecosystem. With a more open app ecosystem and lower development overhead than other platforms, both novice and veteran attackers are able to freely exchange attack tactics and techniques, making opportunistic attacks more profitable and widespread on Android, as pointed out by Ahmed and Sallow (2017, p. 1).

The permission system on Android safeguards resources from misuse by apps installed by the user. The permission system relies on the user to appropriately accept these requests for apps to gain the ability to access resources within the Android phone. Ahmed and Sallow (2017, p. 3) argue that the lack of OS verification for the user-consented permissions allows attackers to easily engineer the consent from users. Social engineering is paramount in the Android attack surface. Similar to the traditional web environment, users fall prey to cleverly crafted advertisements and links, which manipulate their willingness to share permissions. Exploiting user confusion and inherent trust, adversaries can effectively distribute malicious software to devices, a feat that can be easily replicated thanks to Android's popularity. Users are prone to granting more permissions than an application needs (overprivileged), and many third-party applications ask for excessive access to resources they do not need to operate (Ahmed and Sallow 2017, p. 3).

In repackaging attacks, attackers decompile a legitimate application, insert malware, and publish it to official marketplaces or other third-party application stores. The open-source nature of the Android operating system allows attackers to modify the APK, embed malware, and repackage legitimate applications (Ahmed and Sallow 2017, p. 3). Further, the openness of many application stores gives way for attackers to push these malicious applications without raising red flags to most users (Vidas et al. 2011, p. 5). This allows the adversary to benefit from users' implicit trust of familiar applications in the official Android app stores. These repackaged applications can have widespread effect as seen in past malware such as the Geinimi and DroidDream campaigns, which infected dozens of legitimate Android applications. Repackaging is popular with malware authors due to the increased stealth. Many malware detection schemes are unable to detect these attacks, as static analysis of applications fails to realize that malicious code was injected. Although static analysis of such attacks may be successful when the applications are running dynamically, Vidas et al. (2011, p. 6) explain that common antivirus techniques utilizing signature-based dynamic analysis are also unable to capture the malware. Red teams, in this respect, are crucial to simulating attackers using this technique and, by extension, assessing defensive strategies as proposed by Ahmed and Sallow (2017, p. 3).

Another type of attack, privilege escalation, gives adversaries root access to devices in order to circumvent security barriers. Bugiel et al. (2011, p. 2) described such an attack called Soundcomber that leveraged