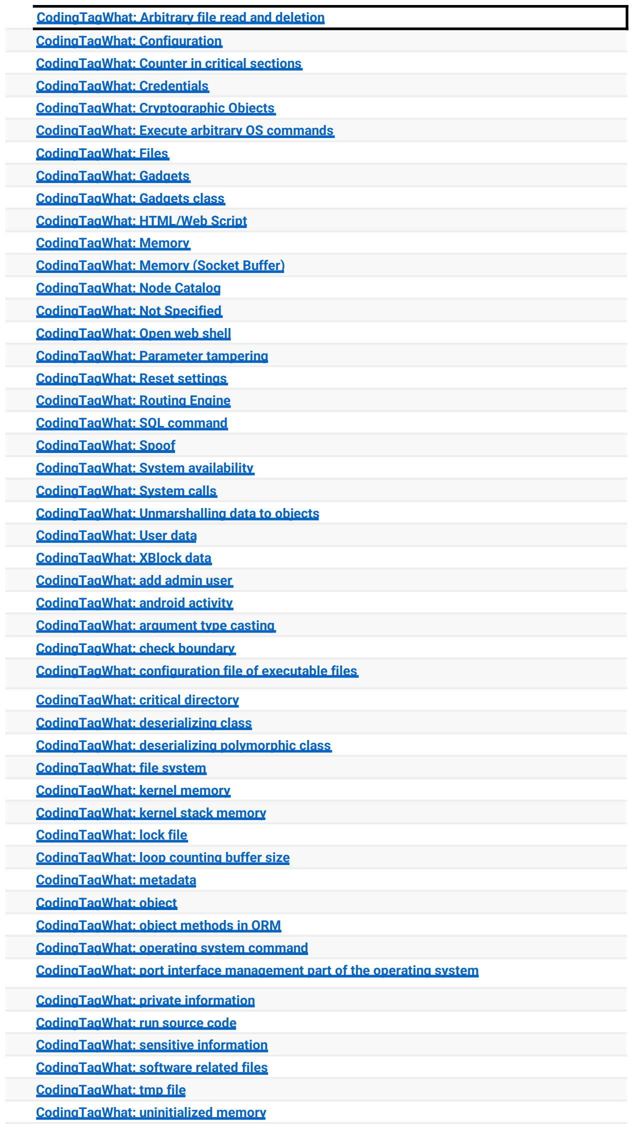
Attack Surface Categorization

Memo#1: What are the targets?



CodingTagWhat: Arbitrary file read and deletion
CodingTagWhat: Configuration
CodingTagWhat: Counter in critical sections
CodingTagWhat: Credentials
CodingTagWhat: Cryptographic Objects
CodingTagWhat: Execute arbitrary OS commands
CodingTagWhat: Files
CodingTagWhat: Gadgets
CodingTagWhat: Gadgets class
CodingTagWhat: HTML/Web Script
CodingTagWhat: Memory
CodingTagWhat: Memory (Socket Buffer)
CodingTagWhat: Node Catalog
CodingTagWhat: Not Specified
CodingTagWhat: Open web shell
CodingTagWhat: Parameter tampering
CodingTagWhat: Reset settings
CodingTagWhat: Routing Engine
CodingTagWhat: SQL command
CodingTagWhat: Spoof
CodingTagWhat: System availability
CodingTagWhat: System calls
CodingTagWhat: Unmarshalling data to objects
CodingTagWhat: User data
CodingTagWhat: XBlock data
CodingTagWhat: add admin user
CodingTagWhat: android activity
CodingTagWhat: argument type casting
CodingTagWhat: check boundary
CodingTagWhat: configuration file of executable files
CodingTagWhat: critical directory
CodingTagWhat: deserializing class
CodingTagWhat: deserializing polymorphic class
CodingTagWhat: file system
CodingTagWhat: kernel memory
CodingTagWhat: kernel stack memory
CodingTagWhat: lock file
CodingTagWhat: loop counting buffer size
CodingTagWhat: metadata
CodingTagWhat: object
CodingTagWhat: object methods in ORM
CodingTagWhat: operating system command
CodingTagWhat: port interface management part of the operating system
CodingTagWhat: private information
CodingTagWhat: run source code
CodingTagWhat: sensitive information
CodingTagWhat: software related files
CodingTagWhat: tmp file
CodingTagWhat: uninitialized memory

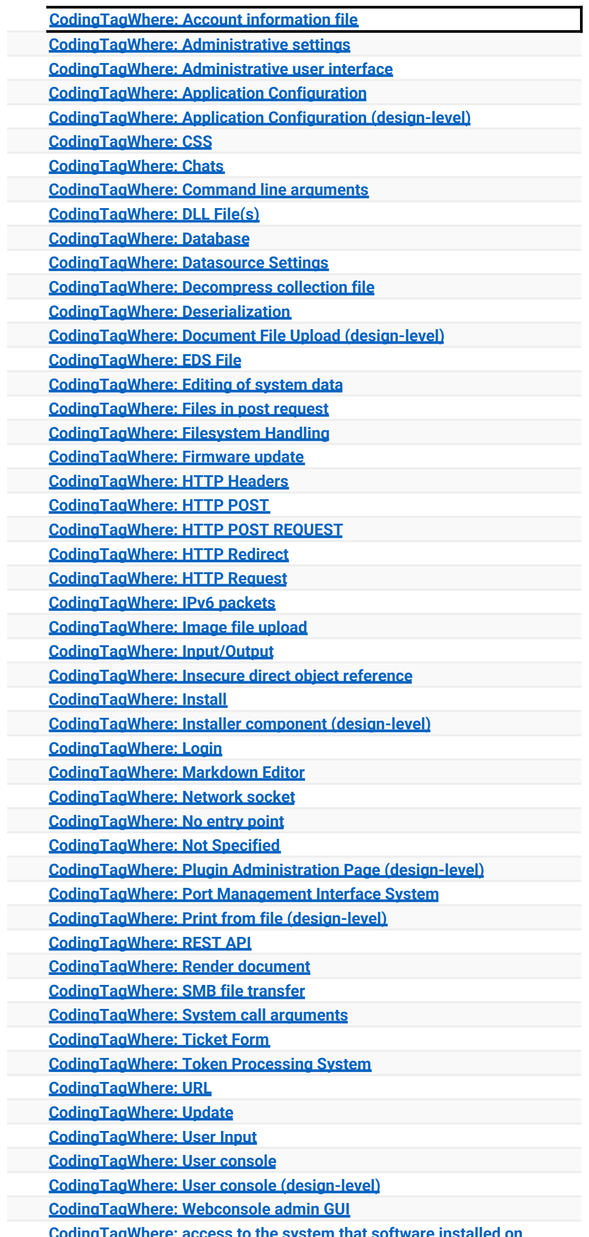
Targets are software components or parts of software applications that attackers try to access.

1. Operating system commands (system calls)
2. Software application files: lock files, tmp files,
3. System files: node catalog in distributed systems
4. HTML/Webscripts
5. Memory allocation/deallocation/tampering/access: socket buffer, kernel memory, kernel stack memory, loops counting buffer size, uninitialized memory, check boundary
6. Marshalling/unmarshalling data objects to/from json: deserializing class, deserializing polymorphic class
7. Android activity
8. Type casting parts
9. Executable code
10. Database
11. Software configuration parts
12. Special Objects or Classes: Cryptographic objects, Gadget Classes
13. Sensitive/private information: Credentials, userdata, metadata

Memo#2: Input data types:

Tiff file	**CVE-2020-6067**	**Out-of-bounds Write**
XML	CVE-2020-6238	CWE-20Improper Input Validation
BLOB A binary large object (blob) is concentrated binary data that's compressed into an individual file inside a database. The large size of the file means they need special storage treatment. Blobs are binary, which means they are usually images, audio or other media.	CVE-2020-7248	Out-of-bounds Write
YAML It's basically a human-readable structured data format. It is less complex and ungainly than XML or JSON, but provides similar capabilities. It essentially allows you to provide powerful configuration settings, without having to learn a more complex code type like CSS, JavaScript, and PHP.	CVE-2020-1947	CWE-502-Deserialization of Untrusted Data
Android Parcel Android Parcel would be that of a message container for lightweight, high-performance Inter-process communication (IPC). .	CVE-2020-0017	CWE-200- Exposure of Sensitive Information to an Unauthorized Actor
JSON	CVE-2019-10749 CVE-2019-10748	CWE-89- Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
	CVE-2018-7489	CWE-184- Incomplete List of Disallowed Inputs
	CVE-2018-18836	CWE-74- Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')
	CVE-2017-18349	CWE20-Improper Input Validation
	CVE-2017-17485	CWE-502- Deserialization of Untrusted Data
	CVE-2014-5017	CWE-89- Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
	CVE-2014-3994	CWE-79- Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
IPv4 packet	CVE-2020-1638	CWE20-Improper Input Validation

Memo#3: Where are the entry points?


CodingTagWhere: Account information file
CodingTagWhere: Administrative settings
CodingTagWhere: Administrative user interface
CodingTagWhere: Application Configuration
CodingTagWhere: Application Configuration (design-level)
CodingTagWhere: CSS
CodingTagWhere: Chats
CodingTagWhere: Command line arguments
CodingTagWhere: DLL File(s)
CodingTagWhere: Database
CodingTagWhere: Datasource Settings
CodingTagWhere: Decompress collection file
CodingTagWhere: Deserialization
CodingTagWhere: Document File Upload (design-level)
CodingTagWhere: EDS File
CodingTagWhere: Editing of system data
CodingTagWhere: Files in post request
CodingTagWhere: Filesystem Handling
CodingTagWhere: Firmware update
CodingTagWhere: HTTP Headers
CodingTagWhere: HTTP POST
CodingTagWhere: HTTP POST REQUEST
CodingTagWhere: HTTP Redirect
CodingTagWhere: HTTP Request
CodingTagWhere: IPv6 packets
CodingTagWhere: Image file upload
CodingTagWhere: Input/Output
CodingTagWhere: Insecure direct object reference
CodingTagWhere: Install
CodingTagWhere: Installer component (design-level)
CodingTagWhere: Login
CodingTagWhere: Markdown Editor
CodingTagWhere: Network socket
CodingTagWhere: No entry point
CodingTagWhere: Not Specified
CodingTagWhere: Plugin Administration Page (design-level)
CodingTagWhere: Port Management Interface System
CodingTagWhere: Print from file (design-level)
CodingTagWhere: REST API
CodingTagWhere: Render document
CodingTagWhere: SMB file transfer
CodingTagWhere: System call arguments
CodingTagWhere: Ticket Form
CodingTagWhere: Token Processing System
CodingTagWhere: URL
CodingTagWhere: Update
CodingTagWhere: User Input
CodingTagWhere: User console
CodingTagWhere: User console (design-level)
CodingTagWhere: Webconsole admin GUI
CodingTagWhere: access to the system that software installed on

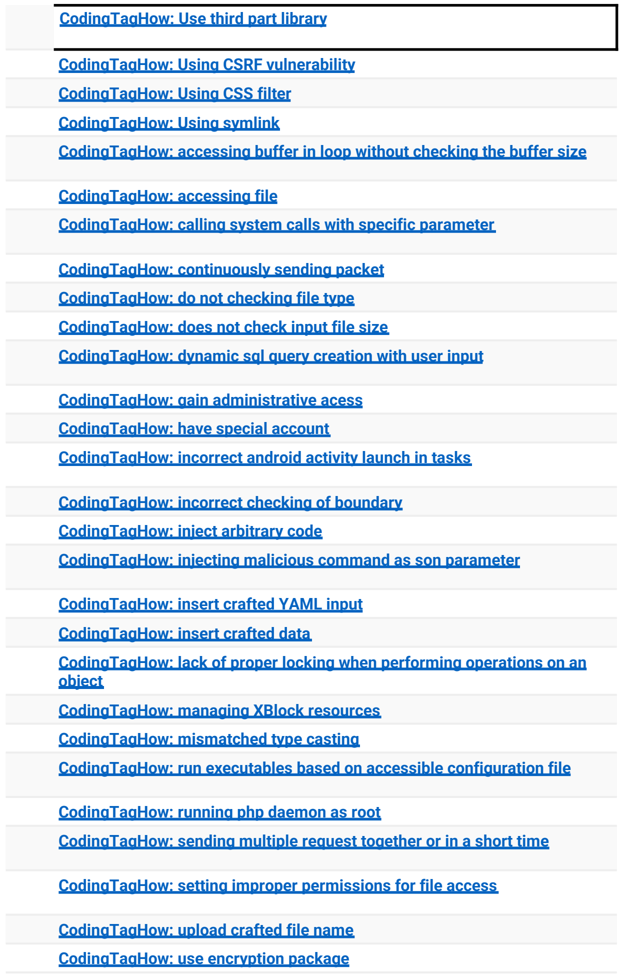
Entry points are parts of software system that attacker can leverage to access targets.

1. System files: account information file (etc/passwd),
2. dll files, eds files
3. Command line arguments
4. Web requests: http post request, http get request, files in post requests
5. Packets: IPv6 packets
6. Network sockets
7. REST APIs
8. Device related arguments
9. Service requests: such as inter procedural communication

Design-level (system sub-modules):

1. Application Configuration
2. File system handling
3. Installer components
4. Update component
5. Editor
6. Chat
7. User Console
8. Web console
9. Plugin administration
10. Port management interface
11. File upload
12. Access to Local system

Memo#4: How does the attack happen?



CodingTagHow: Use third part library
CodingTagHow: Using CSRF vulnerability
CodingTagHow: Using CSS filter
CodingTagHow: Using symlink
CodingTagHow: accessing buffer in loop without checking the buffer size
CodingTagHow: accessing file
CodingTagHow: calling system calls with specific parameter
CodingTagHow: continuously sending packet
CodingTagHow: do not checking file type
CodingTagHow: does not check input file size
CodingTagHow: dynamic sql query creation with user input
CodingTagHow: gain administrative acess
CodingTagHow: have special account
CodingTagHow: incorrect android activity launch in tasks
CodingTagHow: incorrect checking of boundary
CodingTagHow: inject arbitrary code
CodingTagHow: injecting malicious command as son parameter
CodingTagHow: insert crafted YAML input
CodingTagHow: insert crafted data
CodingTagHow: lack of proper locking when performing operations on an object
CodingTagHow: managing XBlock resources
CodingTagHow: mismatched type casting
CodingTagHow: run executables based on accessible configuration file
CodingTagHow: running php daemon as root
CodingTagHow: sending multiple request together or in a short time
CodingTagHow: setting improper permissions for file access
CodingTagHow: upload crafted file name
CodingTagHow: use encryption package

1. Using third party library: encryption package
2. Improper permission: for accessing files, running scripts
3. Run executables based on accessible configuration files
4. Mismatched type casting
5. Dynamic SQL query creation
6. Do not checking input file: type, size
7. Using dangerous technology: CSS filters, symlinks
8. Using incorrect (unsafe) technology: http instead of https
9. Unauthenticated access to account or services
10. Sending multiple requests or packets together in short time