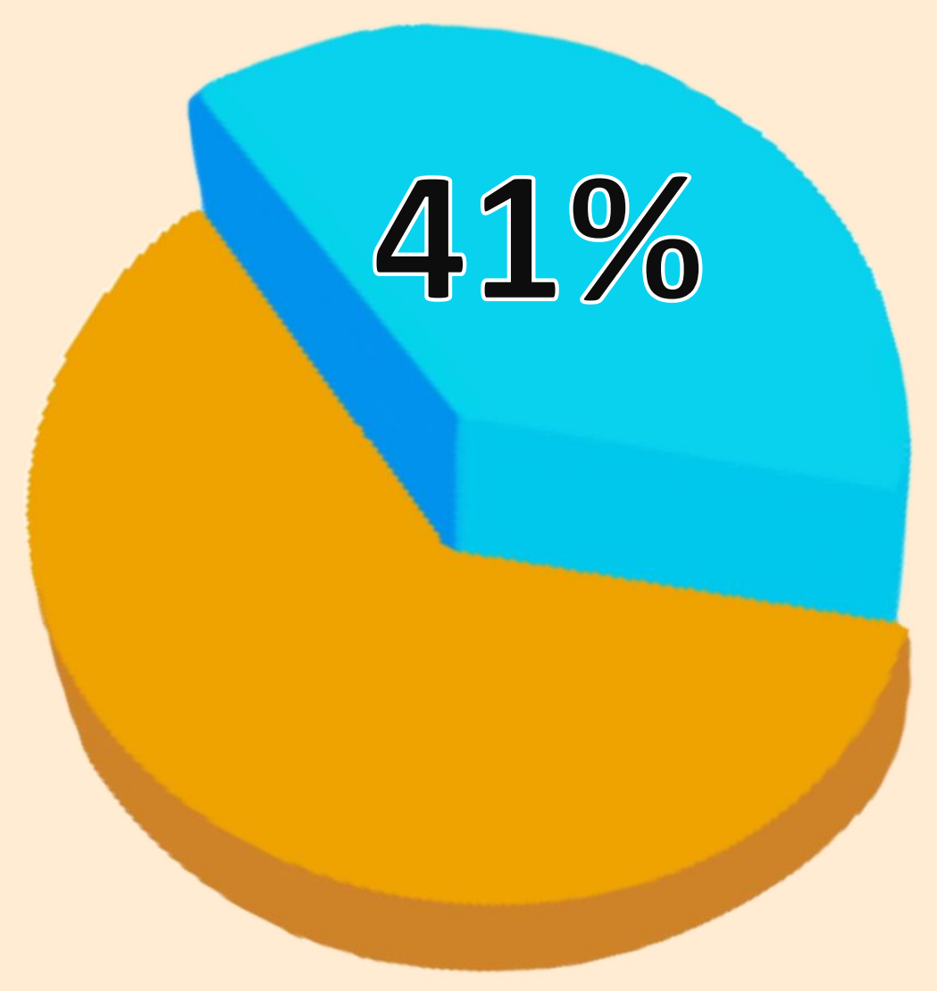
Mobile Game Hacks and Defenses

Introduction to mobile game security

Yang Hu
gymf1010@email.arizona.edu

Shu Lin
shulin@email.arizona.edu

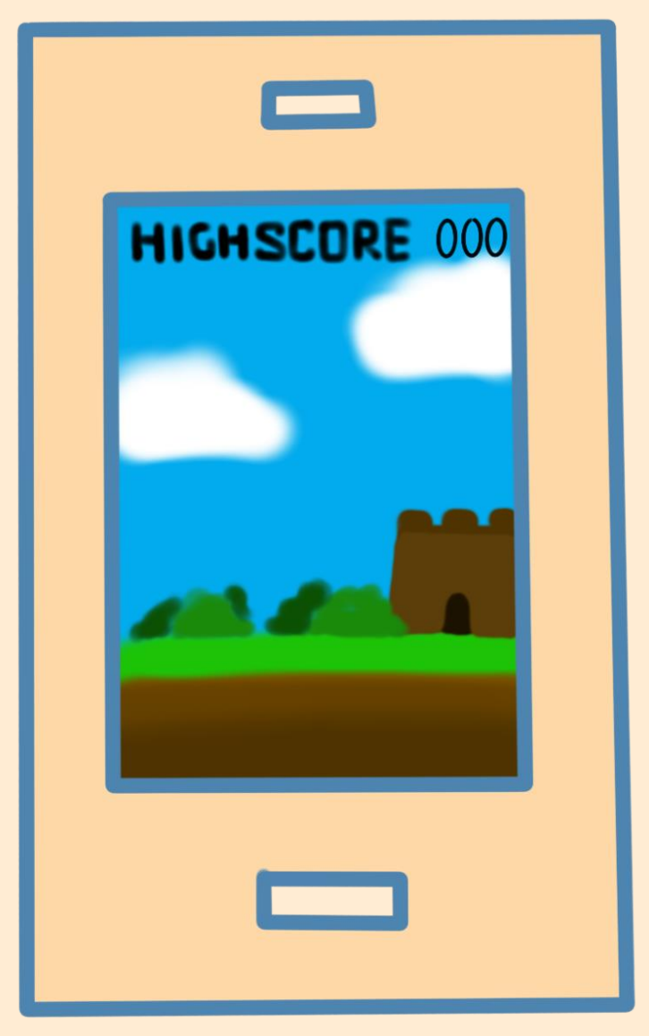
Introduction



Easy Coding

Mobile Game is occupying **41%** of the game market. However, due to its **intermittent Internet communication** and relatively **casual development**, Mobile Game is also significantly more vulnerable to hacks.

Scenario



Game state	Traffic	Server
Memory	Packet (payload)	High score ranking
File		
Code		

Consider a scenario that we want to tamper the **high score** in a mobile game.

In order to find out the game's vulnerability and perform undetectable cheatings, we need to apply different levels of hacking techniques

Hacks & Defenses

Simple

Only require simple tools

Memory Search

High Score: 0			
0	0	72	1
2	1	0	1
0	-1	7	0

High Score: 1			
1	-7	72	2
2	3	1	1
5	7	4	91

High Score: 2			
32	3	-1	0
2	-5	2	1
19	-3	2	11

File Tampering

Memory could be dynamic and thus untraceable. But, saving files are always static.

Secret.txt

```
#Don't tamper!

Highscore: 0x2
```

↓

```
# Nice try

Highscore: 0xff
```

Dynamic Memory

High Score: 2			
32	3	-1	0
2	-5	-9 ➔ 2	
19	-3	2	11

Memory Encryption

XOR values before saving to memory.

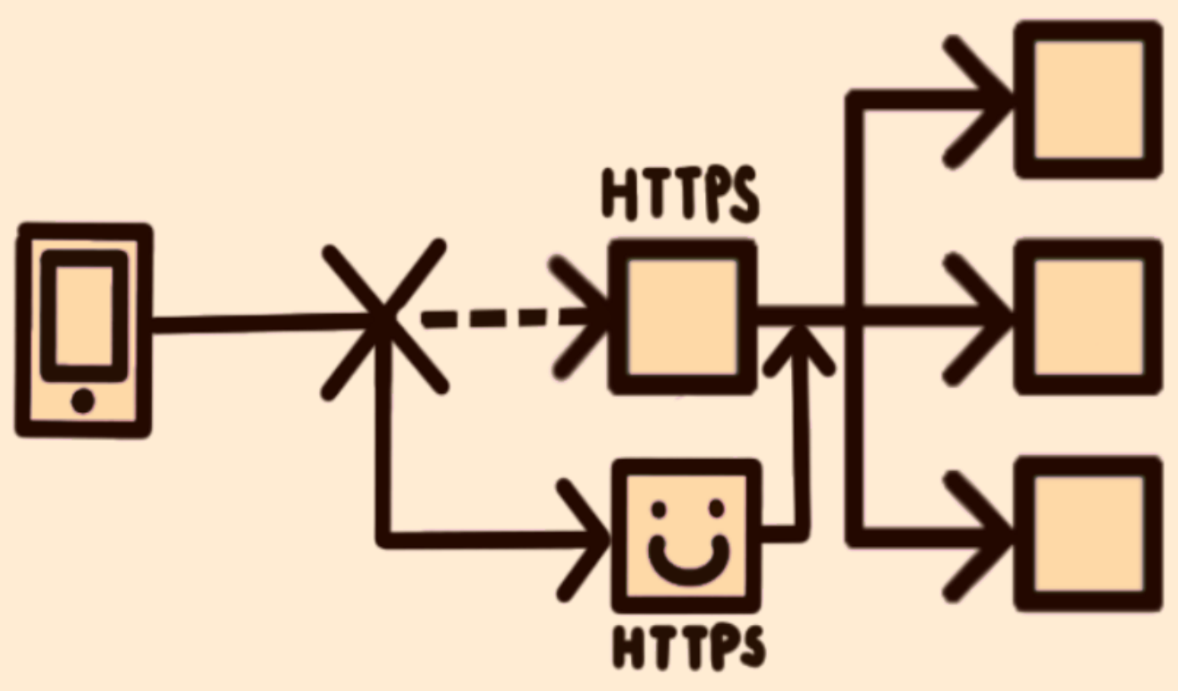
File Encryption

```
0x11 0xA9 0x8
0xA  0x2   0x9
0x8  0x92  0x2
0xE  0x02  0x7
```

Save to server

Sensitive game state would no longer be found locally.

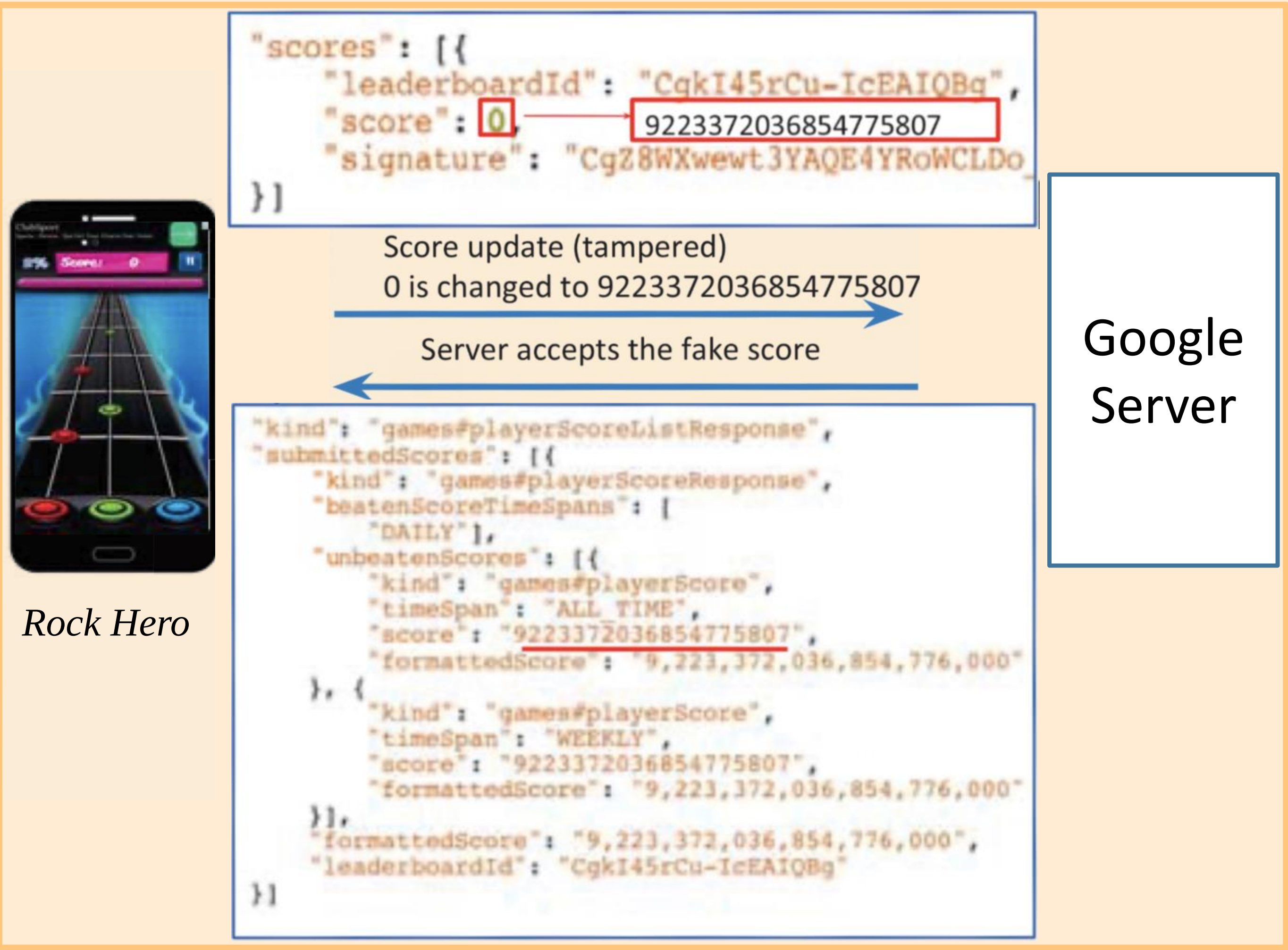
Medium



Require Basic computer science knowledge and network skills

Hijacking Net Packet by Disguised Proxy

With a fake certificate, we can hijack a net packet using a proxy. We could then tamper the payload and hack the game.



```
"scores": [{
    "leaderboardId": "CgkI45rCu-IcEAIQBg",
    "score": 0,
    "signature": "CgZ8WXwewt3YAQE4YRoWCLDo
}]
```

9223372036854775807

Score update (tampered)
0 is changed to 9223372036854775807

Server accepts the fake score

```
"kind": "games#playerScoreListResponse",
"submittedScores": [{
    "kind": "games#playerScoreResponse",
    "beatenScoreTimeSpans": [
        "DAILY"],
    "unbeatenScores": [{
        "kind": "games#playerScore",
        "timeSpan": "ALL_TIME",
        "score": "9223372036854775807",
        "formattedScore": "9,223,372,036,854,776,000"
    }, {
        "kind": "games#playerScore",
        "timeSpan": "WEEKLY",
        "score": "9223372036854775807",
        "formattedScore": "9,223,372,036,854,776,000"
    }],
    "formattedScore": "9,223,372,036,854,776,000",
    "leaderboardId": "CgkI45rCu-IcEAIQBg"
}]
```

Google Server

Rock Hero

Check Proxy	Payload Encryption	Packet Signing
Validate the proxy A common method could be keeping a list of valid certificates.	Encrypt the data in the packet.	Sign the packet by hashing payload content. A common method could be XOR the payload with a local function name.
If (proxy IN valid_certificates) then …	"scores": [{… "score" : 0x8A2E1; … }]	data xor func

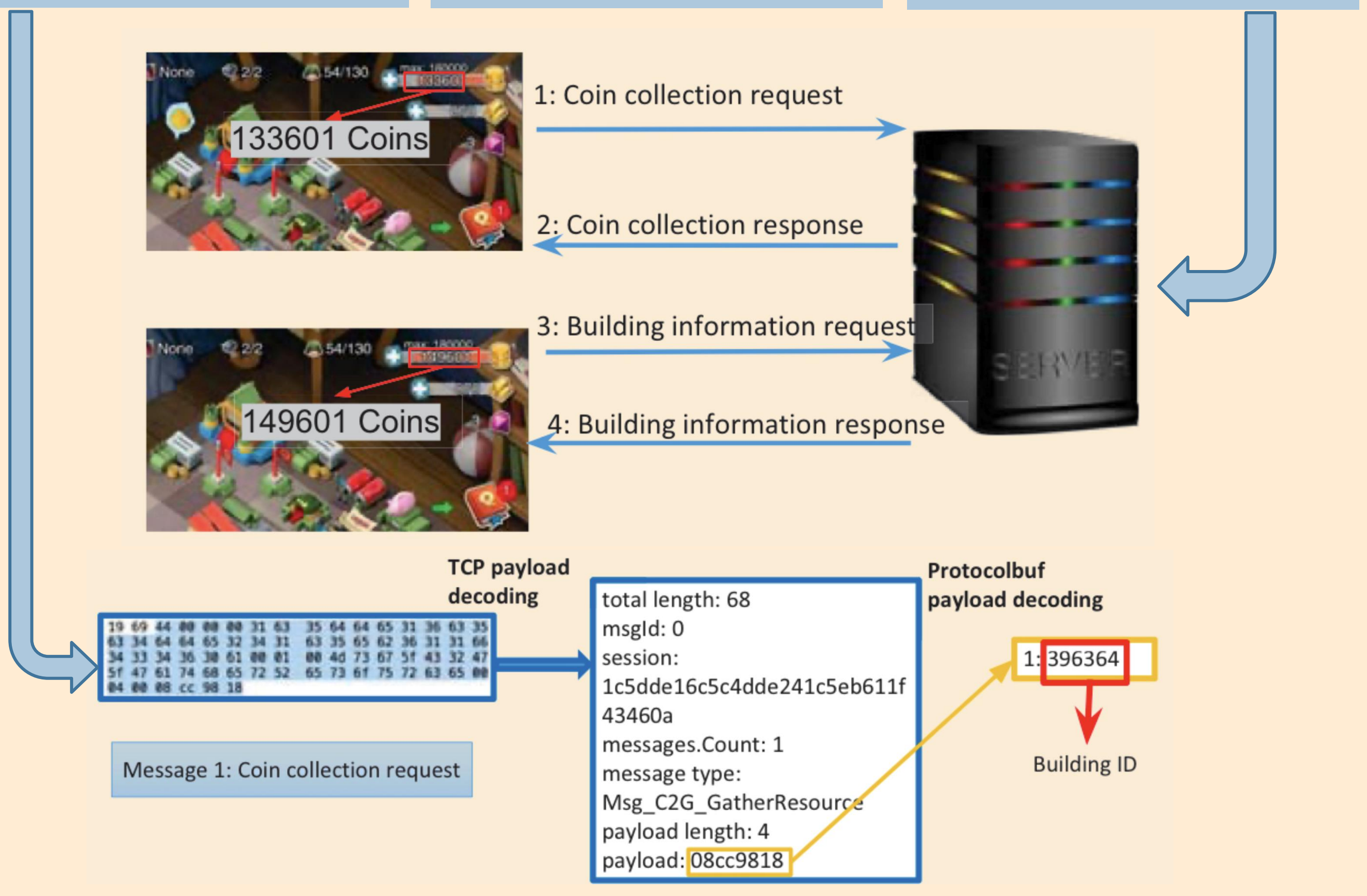
Hard

Require advanced code analysis

Bypass Proxy Check	Decryption	Break Signing Algorithm
Get the certificates list and disguise us as valid proxies.	Learn and use the encryption and decryption code.	Find our the function name used in signing

valid_certificates = {
"gWt3yQK46" ,
"YRw87ee30V" ,
"sO2xTr4c70X"
…
};
	"score" : 0x82E1;
	↓ decrypt
	"score" : 0;
	↓ tamper
	"score" : 99999;
	↓ encrypt
	"score" : 0xF7EF;
		S(data) xor f
		↓ de-sign
		data
		learn ↓ pattern
		Make new payload
		↓ re-sign
		T(data) xor f

Customized Protocol	**Code Obfuscation**	**Client-Server Synchronization**
Message format and encoding could be defined by developers.	Make codes unanalyzable. Server side library could also help.	No result could be generated locally. Client only sends events to server.


133601 Coins
1: Coin collection request
2: Coin collection response
149601 Coins
3: Building information request
4: Building information response
SERVER
TCP payload decoding
Protocolbuf payload decoding
Building ID

```
19 69 44 00 00 00 31 63  35 64 64 65 31 36 63 35
63 34 64 64 65 32 34 31  63 35 65 62 36 31 31 66
34 33 34 36 30 61 00 01  00 4d 73 67 5f 43 32 47
5f 47 61 74 68 65 72 52  65 73 6f 75 72 63 65 00
04 00 08 cc 98 18
```

Message 1: Coin collection request

```
total length: 68
msgId: 0
session:
1c5dde16c5c4dde241c5eb611f
43460a
messages.Count: 1
message type:
Msg_C2G_GatherResource
payload length: 4
payload: 08cc9818
```

1: 396364

References

Tian, Yuan, Chen, Eric, Ma, Xiaojun, Chen, Shuo, Wang, Xiao, & Tague, Patrick. (2016). Swords and shields: A study of mobile game hacks and existing defenses. Proceedings of the 32nd Annual Conference on Computer Security Applications, 5-9, 386-397

Yahyavi, A., Pang, J., & Kemme, B. (2013). Towards providing security for mobile games. Proceedings of the Eighth ACM International Workshop on Mobility in the Evolving Internet Architecture, 47-52.

Cheat Engine. (n.d.). Retrieved November 7, 2019, from https://www.cheatengine.org/.