# Field Report: Building a game engine for 300 DEFCON hackers to smash

GopherCon 2020

Kevin Tyers
Alex Stockwell

Global Hacker conference in Las Vegas

The game was played on August 11th 2018

Scheduled to run for 6.5 hours starting at 10 AM

Over 300 players in 146 teams were registered to compete

$10,000 grand prize

Awarded 100s of prizes (Shirts, Stickers, Badges)

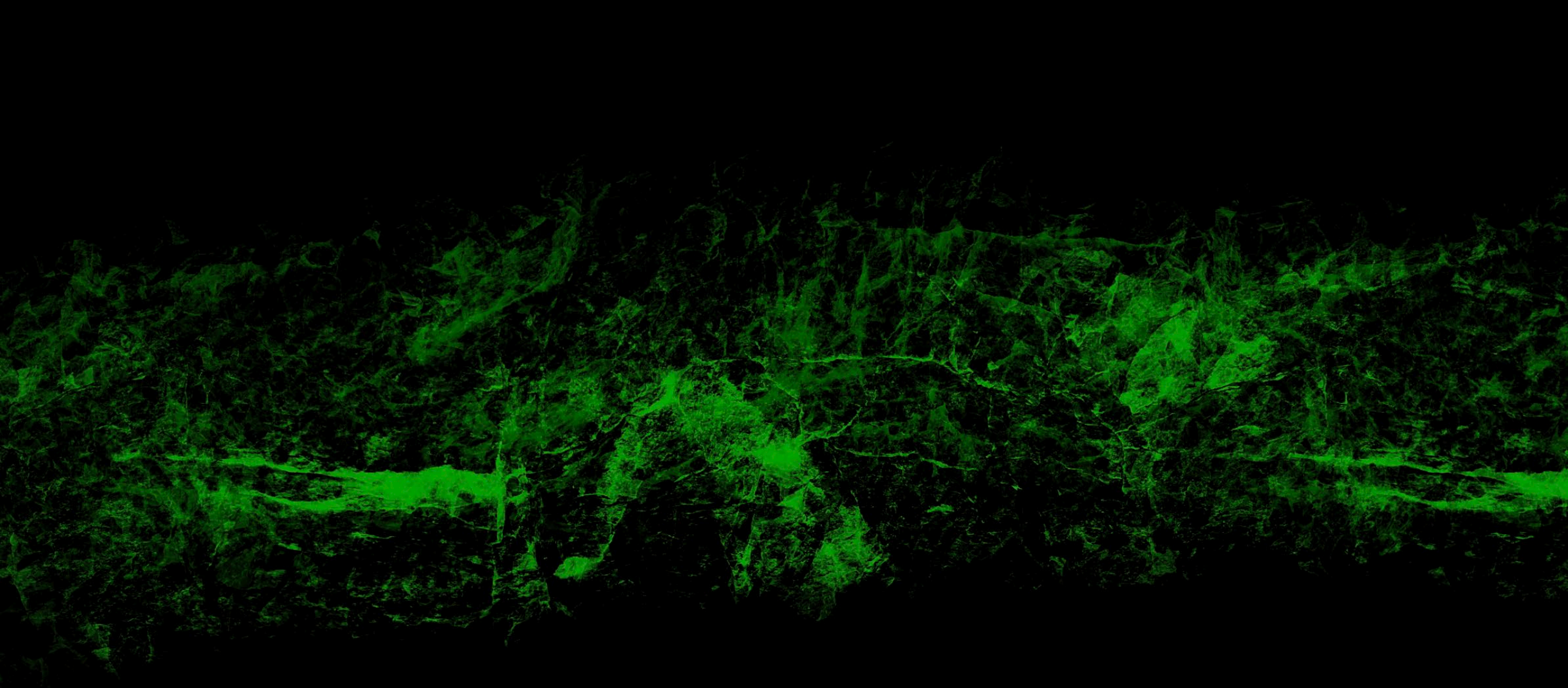
**DEFCON**

Host a unique, immersive, capture-the-flag event

Spend a lot of money on event and players

Engaging story narrative with 1:1 interactions with a “Handler”

## The Idea



# Designing the Experience

A Hacker has broken into Space-Y

Player is an analyst working with an intelligence agency

Players find a deeper conspiracy involving aliens, Darknet arms dealers, and the fate of Earth itself

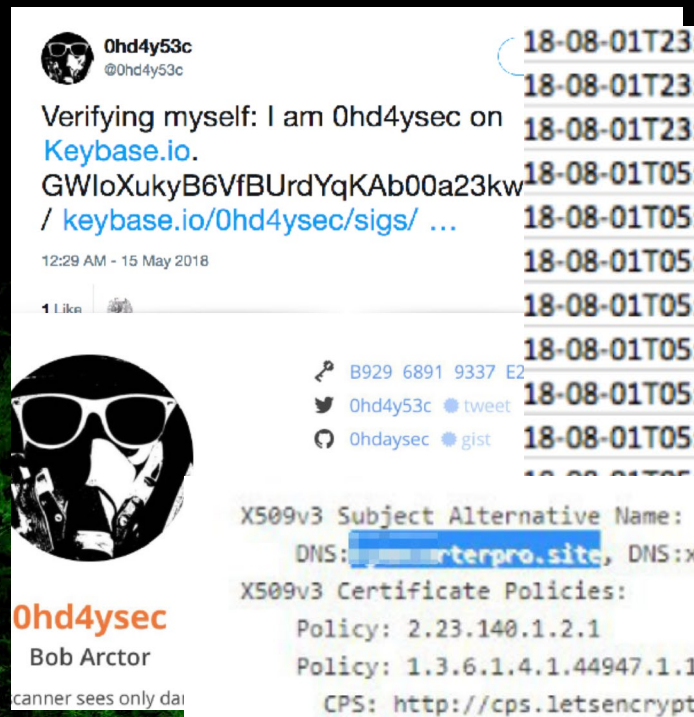
## The Story

Year long seeding campaign

Various synthetic data points throughout the Internet

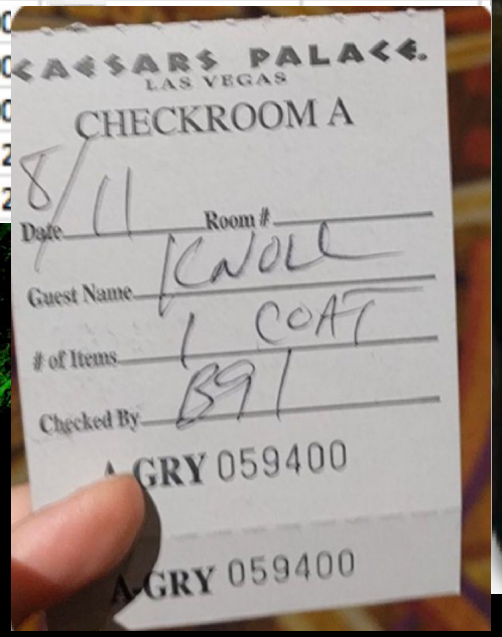
Domains
&Websites
&Social-media
&Github
&Tor
&Certificates
&Honeypots
&Netflow
&Many Others

# Synthetic Data




| | | | | |
|---|---|---|---|---|
| 18-08-01T23:48:36Z | 103.38.226.67 | 7107 | 99.108.101 | 3389 |
| 18-08-01T23:52:46Z | 103.207.39.175 | 21625 | 99.108.101 | 3389 |
| 18-08-01T23:58:15Z | 85.152.57.23 | 65353 | 99.108.101 | 3389 |
| 18-08-01T05:00:00Z | 27.0.0.197 | 20000 | 99.108.101 | 4453 |
| 18-08-01T05:00:05Z | 27.0.0.197 | 20001 | 99.108.101 | 4453 |
| 18-08-01T05:00:10Z | 27.0.0.197 | 20002 | 99.108.101 | 4453 |
| 18-08-01T05:00:42Z | 23.20.0.222 | 42000 | 99.108.101 | 4453 |
| 18-08-01T05:00:49Z | 23.20.0.222 | 42001 | 99.108.101 | 4453 |
| 18-08-01T05:01:00Z | 23.20.0.222 | 42002 | 99.108.101 | 4453 |
| 18-08-01T05:03:20Z | 52.41.100.165 | 1700 | | |

```
X509v3 Subject Alternative Name:
    DNS:[redacted]rterpro.site, DNS:xn--cd-[redacted].stream
X509v3 Certificate Policies:
    Policy: 2.23.140.1.2.1
    Policy: 1.3.6.1.4.1.44947.1.1.1
      CPS: http://cps.letsencrypt.org
      User Notice:
        Explicit Text: This Certificate may only be relied u
```



# OSINT | SIGINT | HUMINT

We had to stay vigilant and prioritize work as we had a hard deadline quickly approaching

Feb-Mar Ideas

Apr Sponsors and content

May Registration build

Jun-Jul Game Engine build

Aug DEFCON 26

## Timeline/Constraints

# Building the Game Engine

Support real-time interactions with hundreds of players

Secure and Resilient

Check Submitted Flags and Keep State per team

Realistically interact with players (programmatically)

Register teams and validate email addresses

Ensure rules compliance

Etc...

## Requirements

1:1 Interactive feel? – “A bot! Slack bot!”

We need a game engine – “Go!”

We have 2 developers – “Divide & Conquer!”

How will game & bots interact? – “REST API!”

What if the bot dies? – “Stateless bot!”

How to achieve that? – “Engine drives all things!”

Managing players/teams? – “More bots!”

Helping players? – “Hints! Time based! Hint bot!”

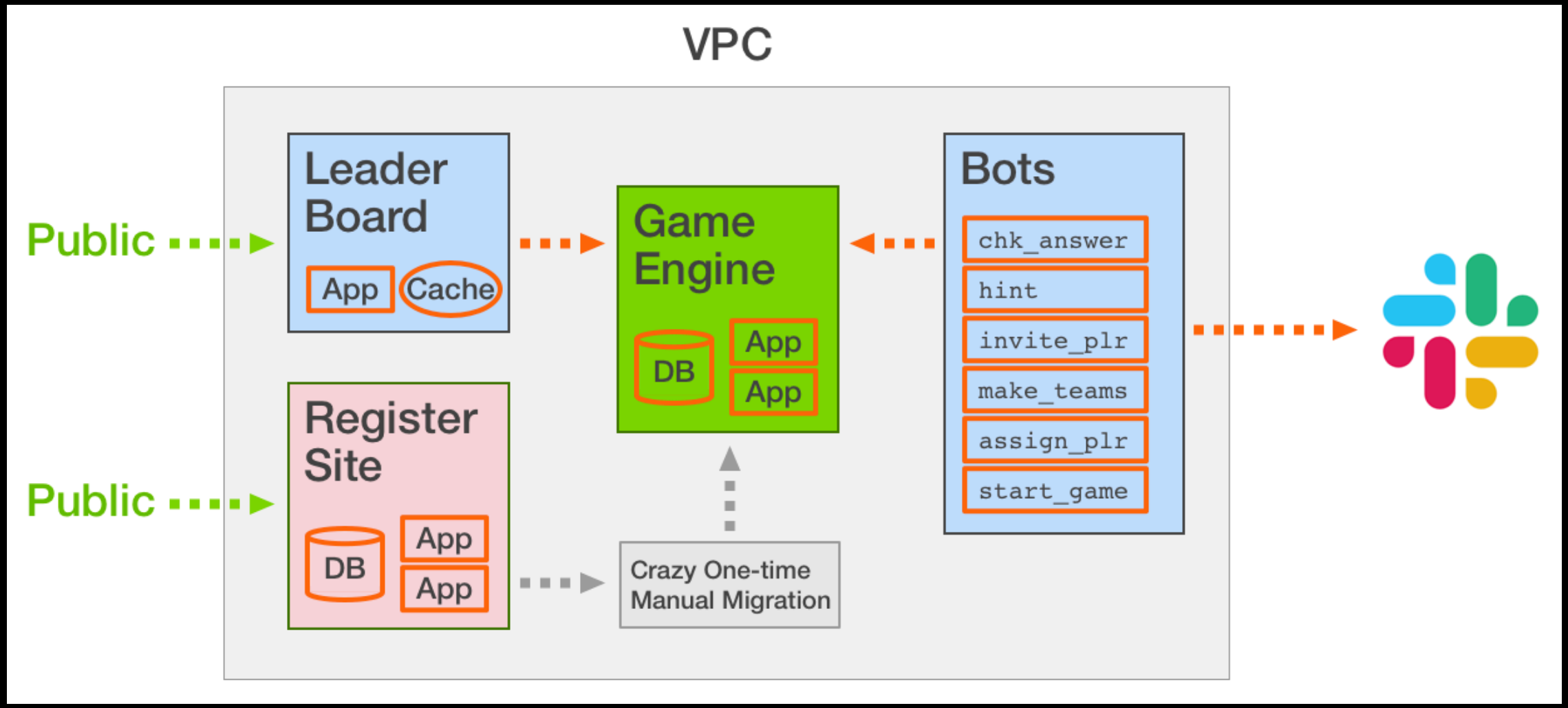
**Decisions**

Game Engine: API, holds all interaction logic

Bots: Series of Slack (and other) bots that ferry messages/actions between players & the Handler

Slack is the only player interface to the game.

# Architecture: Overview


VPC
Public
Leader Board
App
Cache
Public
Register Site
DB
App
App
Game Engine
DB
App
App
Crazy One-time Manual Migration
Bots
chk_answer
hint
invite_plr
make_teams
assign_plr
start_game

Process & respond to players – /check-answer

Manage players – /needing-invite; /set-slack-id

Manage teams – /no-slack-channel; /set-channel

Hints – /current-hints; /ack-hint

## Architecture: Game Engine API

ctr_game_api_routes.go
ctr_game_api_test.go
ctr_game_api.go
ctr_game_control.go
ctr_game_event.go
ctr_game_flag_answer.go
ctr_game_flag_hint.go
ctr_game_flag.go
ctr_game_helpers.go
ctr_game_leaderboard.go
ctr_game_player.go
ctr_game_registration_test.go
ctr_game_registration.go

```
181     apiRouter.NotFound = notFoundHandler(dso)
182     apiRouter.PanicHandler = panicHandler(dso)
183
184     // Gameplay API
185     apiRouter.POST("/games/:game_id/api/v1/check-answer", mwLogger(GamePlayAPI_Handler_Ch
186     apiRouter.GET("/games/:game_id/api/v1/teams/all", mwLogger(GamePlayAPI_Handler_GameTe
187     apiRouter.GET("/games/:game_id/api/v1/teams/no-slack-channel", mwLogger(GamePlayAPI_H
188     apiRouter.POST("/games/:game_id/api/v1/teams/get-team-state", mwLogger(GamePlayAPI_Ha
189     apiRouter.POST("/games/:game_id/api/v1/teams/set-slack-channels", mwLogger(GamePlayAP
190     apiRouter.GET("/games/:game_id/api/v1/players/needing-invite", mwLogger(GamePlayAPI_H
191     apiRouter.POST("/games/:game_id/api/v1/players/set-invite-sent", mwLogger(GamePlayAPI
192     apiRouter.GET("/games/:game_id/api/v1/players/no-slack-id", mwLogger(GamePlayAPI_Hand
193     apiRouter.POST("/games/:game_id/api/v1/players/set-slack-id", mwLogger(GamePlayAPI_Ha
194     apiRouter.GET("/games/:game_id/api/v1/hints/current", mwLogger(GamePlayAPI_Handler_Ga
195     apiRouter.POST("/games/:game_id/api/v1/hints/set-hint-received", mwLogger(GamePlayAPI
196
```

```
2018/08/11 14:08:41 172.31.46.25:33124 Completed POST for /check-answer in 235.104573ms
2018/08/11 14:08:42 172.31.46.25:33130 Completed POST for /teams/get-team-state in 10.151216ms
2018/08/11 14:08:44 172.31.46.25:33134 Completed GET for /players/no-slack-id in 7.722205ms
2018/08/11 14:08:44 172.31.46.25:33138 Completed GET for /teams/all in 5.482691ms
2018/08/11 14:08:54 172.31.46.25:33140 Completed GET for /players/no-slack-id in 7.930311ms
2018/08/11 14:08:54 172.31.46.25:33144 Completed GET for /teams/all in 5.10384ms
2018/08/11 14:08:56 EVENT: {"ID":0,"CreatedAt":"0001-01-01T00:00:00Z","UpdatedAt":"0001-01-01T00:00:00Z"
2018/08/11 14:08:56 172.31.46.25:33148 Completed POST for /check-answer in 18.414851ms
2018/08/11 14:09:04 172.31.46.25:33154 Completed GET for /players/no-slack-id in 8.590259ms
2018/08/11 14:09:05 172.31.46.25:33158 Completed GET for /teams/all in 5.096029ms
2018/08/11 14:09:11 EVENT: {"ID":0,"CreatedAt":"0001-01-01T00:00:00Z","UpdatedAt":"0001-01-01T00:00:00Z"
2018/08/11 14:09:11 172.31.46.25:33162 Completed POST for /check-answer in 17.557781ms
2018/08/11 14:09:15 172.31.46.25:33168 Completed GET for /players/no-slack-id in 8.421973ms
2018/08/11 14:09:15 172.31.46.25:33172 Completed GET for /teams/all in 5.139556ms
```

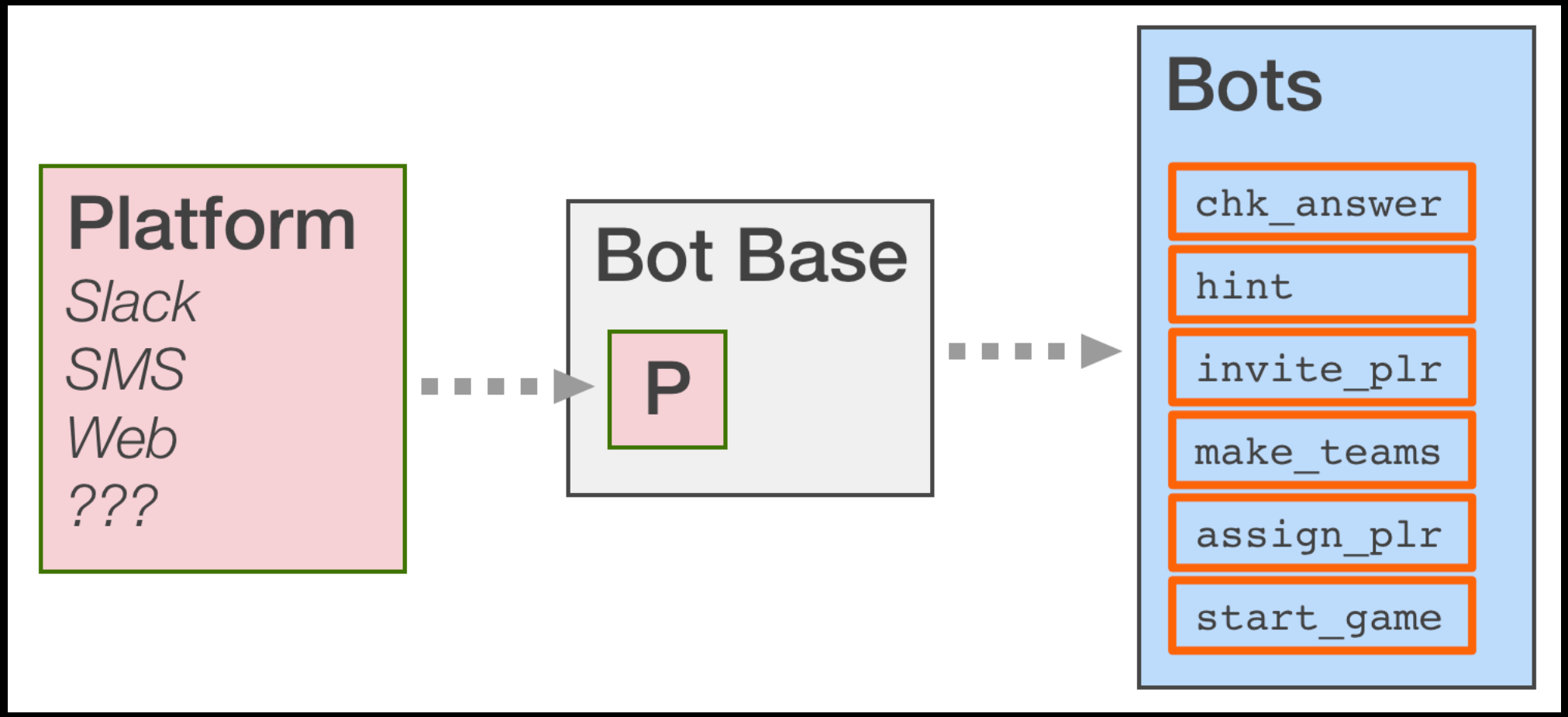
Built bots that each performed a specific task

Bots were stateless, the engine handled everything

# Architecture: Bots

| | |
|---|---|
| `chk_answer` | /check-answer |
| `hint` | /hints/current<br>/hints/set-hint-received |
| `invite_plr` | /players/needing-invite<br>/players/set-invite-sent |
| `make_teams` | /teams/no-slack-channel<br>/teams/set-slack-channels |
| `assign_plr` | /players/no-slack-id<br>/players/set-slack-id |
| `start_game` | /teams/all |

# Architecture: Bots mapped to APIs


Platform
Slack
SMS
Web
???
Bot Base
P
Bots
chk_answer
hint
invite_plr
make_teams
assign_plr
start_game

```go
package platforms

// Platform interface defines the API that all custom platform bots
// (e.g. a Slack bot, or webapp bot) must implement.
// It is critical that any implementations prevent Next() and Send()
// from blocking indefinitely.
type Platform interface {
    Start() error
    Stop() error

    // For CheckAnswer bot
    Next() ([]InboundMessage, bool, error) // bool == true if timeout was reached awaiting message
    Send(OutboundMessage) error

    // For TeamAssignment bot
    AssignMedium(TeamAssignmentRequest) (TeamAssignmentResponse, error)

    // For PlayerInvite bot
    InviteUser(PlayerInviteRequest) error

    // For PlayerAssignment bot
    AssignUserID(PlayerIDRequest) (PlayerIDResponse, bool, error) // bool == false if player was not found in platform
}
```

# Platform

```go
7   type Base struct {
8       ErrHandlerFn func(error)
9
10      adapter platforms.Platform
11      game    GameConfig
12
13      healthcheckC chan struct{}
14      quitC        chan struct{}
15  }
16
17  func NewBase(adapter platforms.Platform, game GameConfig) *Base {
18      return &Base{ adapter: adapter, game: game }
19  }
20
21  func (b *Base) HandleErr(err error) {
22      if b.ErrHandlerFn ≠ nil {
23          b.ErrHandlerFn(err)
24          return
25      }
26  }
```

```go
78  // Start initializes the bot
79  func (t *CheckAnswerBot) Start() {
80      // Start platform adapter
81      err := t.adapter.Start()
82      if err ≠ nil { ... }
83
84      for {
85          select {
86          case ←t.quitC:
87              ...
88          default:
89              // Do work
90              err, deliveryErrs := t.FetchAndProcessMessages()
91              if err ≠ nil { ... }
92              if deliveryErrs ≠ nil { ... }
93          }
```

# Bot Base & Check Answer Bot

```go
128 func (t *CheckAnswerBot) ProcessMessage(inboundMessage platforms.InboundMessage) error {
129     // Validate potential answer? Some is already done in the Platform.
130     if len([]rune(inboundMessage.GetMessage())) > 384 { ... }
131
132     // Ack message received back to Platform
133     ackMsg := &platforms.OutboundMessageBase{ MediumID: inboundMessage.GetMediumID(), UserID:   inboundMessage.GetUserID(), Message:  MSG_THINK
134
135     serr := t.adapter.Send(ackMsg)
136
137     // Start forming response
138     outboundMsg := &platforms.OutboundMessageBase{}
139     outboundMsg.SetMediumID(inboundMessage.GetMediumID())
140
141     // Check answer with gameplay API
142
143     checkAnswerResp, err := t.CheckAnswer(inboundMessage)
144
145     // If the game was won
146     if checkAnswerResp.WonTheGame ≠ nil && *checkAnswerResp.WonTheGame == true { ... }
147
148     // If the answer was right
149     if checkAnswerResp.CorrectAnswer ≠ nil && *checkAnswerResp.CorrectAnswer == true {
150         outboundMsg.SetMessage(fmt.Sprintf("%v \n\n%v", checkAnswerResp.CurrentFlag.CorrectMessage, checkAnswerResp.NextFlag.Prompt))
151
152         outboundMsg.SetAnswerCorrect(true)
153         serr := t.adapter.Send(outboundMsg)
154
155         }
156         return nil
157     }
158
159     // If the answer was wrong
160     if checkAnswerResp.CorrectAnswer ≠ nil && *checkAnswerResp.CorrectAnswer == false {
```

# Check Answer Bot: Message Processing

```go
199 func route_GamePlayAPIv1_CheckAnswer_POST(dso *DataSourceOrchestration) httprouter.Handle {
200     return func(w http.ResponseWriter, r *http.Request, rp httprouter.Params) {
201         // Get ctx
202         ...
203
204         // Ensure team isn't finished!
205         if team.FinishedTheGame ≠ nil { ... }
206
207         // Ensure team isn't banned!
208         if team.IsBanned { ... }
209
210         // Ensure guess is not blank
211         if requestPayload.SubmittedAnswer == "" { ... }
212
213         // Via team's `current_flag_id`, get all possible answers from db [3]
214         currentFlagAnswers, err := currentFlag.ReturnAllAnswers(dso)
215         if err ≠ nil { ... }
216         if len(currentFlagAnswers) < 1 { ... }
217
218         // Check the guess against the flag's answers
            guessIsCorrect := currentFlagAnswers.GuessIsCorrect(requestPayload.SubmittedAnswer)
221         // If guess is wrong: issue HTTP response
222         if !guessIsCorrect { ... }
223
224         // AT THIS POINT:
225         // The user/team's guess is correct.
226
227         // Get the next flag (or learn that they finished the game)
228         nextFlag, finishedTheGame, err := currentFlag.GetNextFlag(dso)
229         if err ≠ nil { ... }
230
231         if finishedTheGame {
```

# Game Engine API: /check-answer (1 of 2)

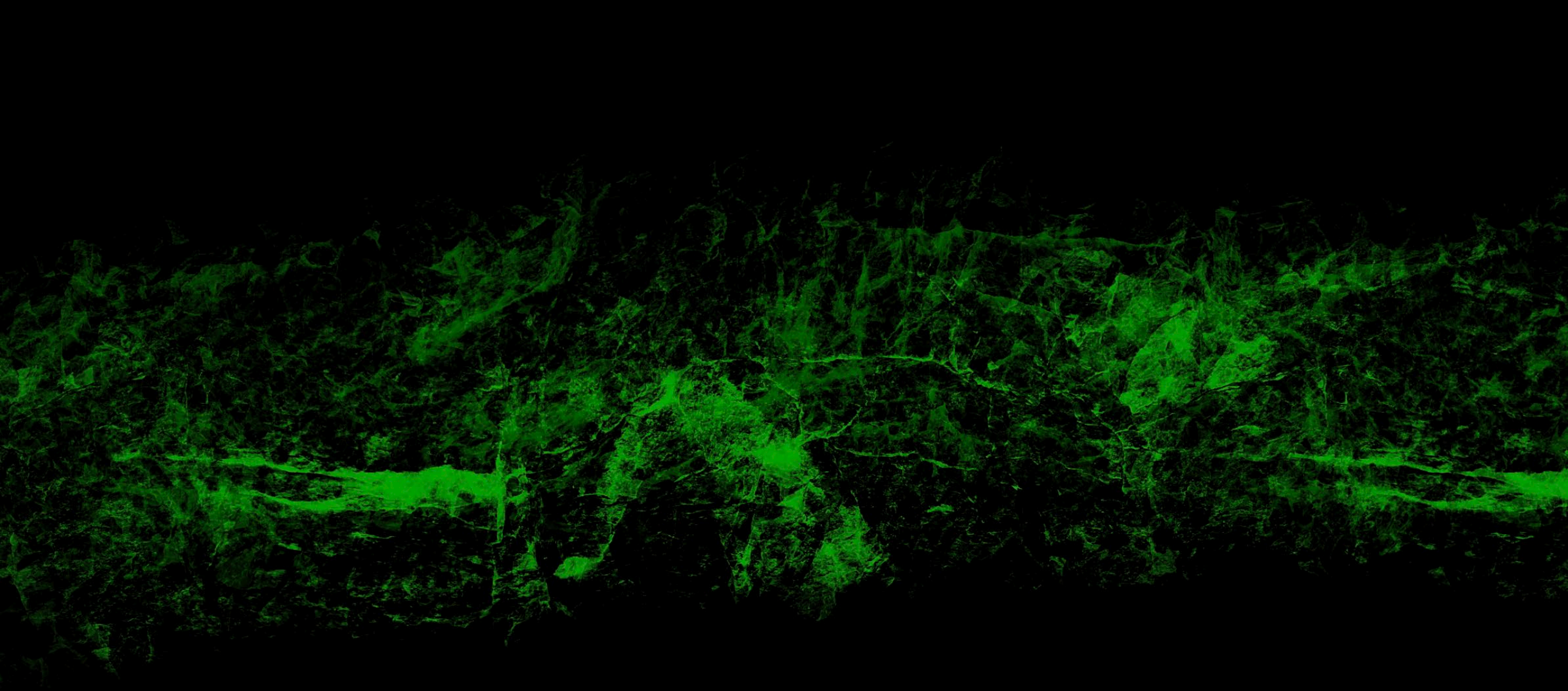
```go
        // Determine the highest flag any team has reached
        // topFlagOrder := teamsWithFlagOrder[0].FlagOrder
        topFlagID := teamsWithFlagOrder[0].GameTeam.CurrentFlagID

        // Persist new flag's ID + timestamp to team's db record
        // Reset current_flag_hint_delivered in team's db record
        team.CurrentFlagID = nextFlag.ID
        team.CurrentFlagStartTime = now
        team.CurrentFlagHintTime = now
        team.CurrentFlagHintsDelivered = 0
        if err := dso.AppDB.Save(&team).Error; err ≠ nil { ... }
        // Check if the team was on the lead flag
        if teamWasOnLeadFlag {
            // If they were, reset CurrentFlagStartTime for any other teams on that flag
            ...
        }

        SaveGameEvent(GameEvent{ GameID: game.ID, GameTeamID: team.ID, GameFlagID: currentFlag.ID, EventType: EVENT_TYPE_GUESS, Message: request
        a := CorrectAnswer(true)
        resp := Response_GamePlayAPIv1_GameTeams_CheckAnswer{ CorrectAnswer: &a, Team: team, CurrentFlag: currentFlag, NextFlag: nextFlag, }

        rndr.JSON(w, http.StatusOK, resp)
        return
    }
}
```

# **Game Engine API:** /check-answer (2 of 2)



# The Event

KEURIG
BREW

EIKI
SAVE A
DC480
The Technology

We had no scoreboard, and built one the night before

Vital for Spectator Interest

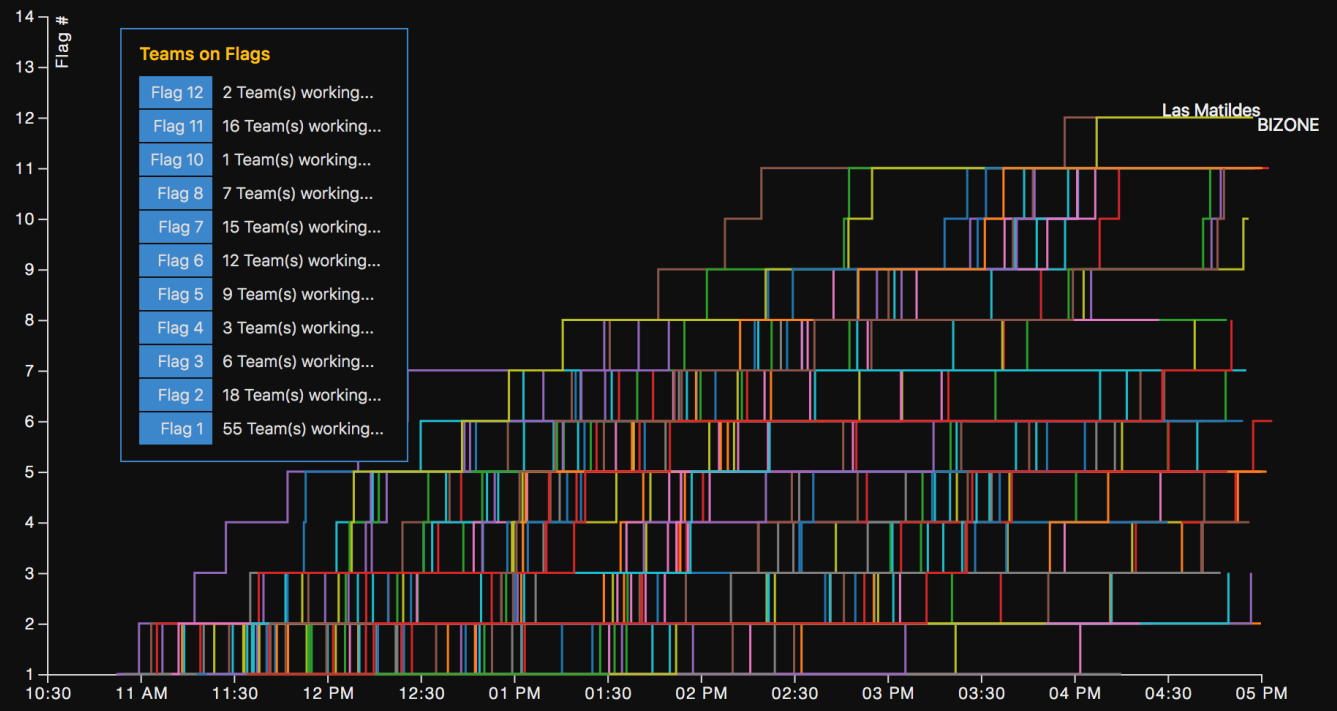
**11:09 PM the night before Leaderboard**

iCTF Leaderboard

Alex

Not Secure 52.33.147.213:9000

| 145 Teams | 14 Flags | GAME OVER | Las Matildes Lead Team | 12 Lead Flag |
|---|---|---|---|---|



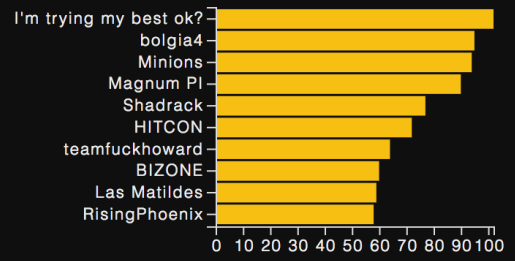

**Teams on Flags**

| | |
|---|---|
| Flag 12 | 2 Team(s) working... |
| Flag 11 | 16 Team(s) working... |
| Flag 10 | 1 Team(s) working... |
| Flag 8 | 7 Team(s) working... |
| Flag 7 | 15 Team(s) working... |
| Flag 6 | 12 Team(s) working... |
| Flag 5 | 9 Team(s) working... |
| Flag 4 | 3 Team(s) working... |
| Flag 3 | 6 Team(s) working... |
| Flag 2 | 18 Team(s) working... |
| Flag 1 | 55 Team(s) working... |

**Leaderboard**

| | |
|---|---|
| 1 | Las Matildes (Flag 12 since 3:56 PM) |
| 2 | BIZONE (Flag 12 since 4:07 PM) |
| 3 | Killjoys (Flag 11 since 2:47 PM) |
| 4 | BAH-Humbug (Flag 11 since 3:25 PM) |
| 5 | Morcy (Flag 11 since 3:31 PM) |
| 6 | Triag3 (Flag 11 since 3:37 PM) |
| 7 | Taint (Flag 11 since 3:43 PM) |
| 8 | TheRealBroCTF (Flag 11 since 3:46 PM) |
| 9 | boring_wozniak (Flag 11 since 3:47 PM) |
| 10 | GoMartys (Flag 11 since 3:58 PM) |

**Top Guessers**




**Latest Submissions**

| | |
|---|---|
| INCORRECT | Minions submitted for flag 6 at 17:03:37 |
| INCORRECT | Minions submitted for flag 6 at 17:03:09 |
| INCORRECT | Cyber W00kiees submitted for flag 11 at 17:02:34 |
| INCORRECT | Cyber W00kiees submitted for flag 11 at 17:02:21 |
| INCORRECT | Jopple submitted for flag 5 at 17:01:49 |
| INCORRECT | Cyber W00kiees submitted for flag 11 at 17:01:10 |
| INCORRECT | Minions submitted for flag 6 at 17:00:48 |

Hit a snag during the team data migration from registration to the game engine

We were patching code up until 8 minutes after game launched

Slack rate-limited our registrations

**7:25 AM - 10:30 AM Launch**

Engine was designed to allow us to hot-modify all game components

Added an additional correct answer to a flag that one team found but we hadn't thought of!

## 11:52 AM Early Progress

Instituted a number of abuse counter-measures in the engine/bots

One unforeseen avenue of abuse was exploited

Only had to ban one team

## 1:14 PM Abuse & Counter-measures

We had zero technical issues

Allowed for focus on managing the players and the flow of the game

Later, coordinating the human interactions in the final stages of the game

## 3:01 PM Focus on the Players

Cafè Americano
Concierge
Hotel Registration
Seven Stars and Diamond Registration
Emperors Essentials
Taxi Stand
Bell Desk
Brad
Lobby Bar
H3
Palace Court Tables
JULIUS TOWER
Bacchanal Buffet
Snackus Maximus
Fortuna Pool
Neptune Pool
Temple Pool
Venus Pool Club
Casino Host
Cashier
Nicole
To Julius Tower
Beijing Noodle No.9
Main Valet
Palace Casino
Brioche by Guy Savoy
Marc
Trev
Apostrophe Bar
Rao's
MR CHOW
Venus Garden
Security
Palace Court Slots
Cleopatra's Barge
Pronto by Giada
Entrance to MR CHOW
Shannon
TJ
NH
Jack
Emperor's Essentials
To Conference Center
Sun Gods
Alto Bar
Montecristo Cigar Bar
Statue of David
Appian Way Shops
Nobu Hotel
Business Center
To Palace Tower & Conference Center
Conference Center
Searsucker
OMNIA Pit
Old Homestead Steakhouse
Nobu Way
Nobu Restaurant & Lounge
Appian Way Shops
PALACE TOWER
OMNIA Nightclub
Mesa Grill
To Forum Tower
FORUM TOWER
Vista Cocktail Lounge
Bonner
Numb Bar
Total Rewards
Race & Sports Book
Kamen
Poker Room
Forum Casino
Gordon Ramsay Pub & Grill
Box Office

Only one team completed the final flag

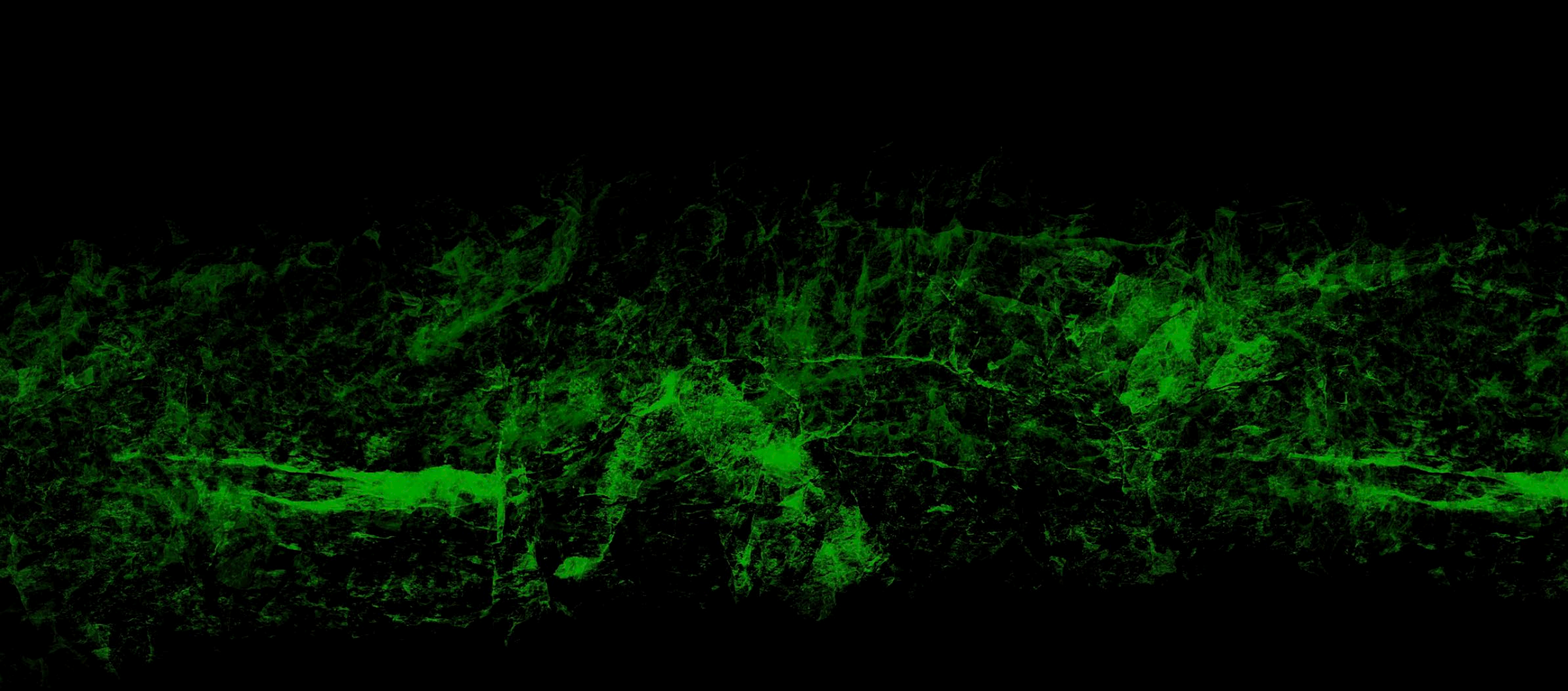
Gave out tons of prizes

Had a big party!

Finally Slept

**4:31 PM FIN**

WE KNOW YOUR PASSWORD
PayPal
DC480
Carnegie Mellon University



# Post Mortem

Design allowed our team to monitor all player/Handler interactions

Able to assume the role of the Handler at any time

Having ability to modify the game content at any time proved a very potent way to mitigate a ton of risks

Good application design facilitated all of this: stateless bots, engine API, good visibility capabilities/UI

Thoughtful planning, good application security, and a solid platform (Go), let us worry less about the infrastructure, and focus more on the experience and the game itself, which was incredible.

Running a contest is hard, and people can get crazy when thousands of dollars are on the line

# Thank You