악성코드 분석 웹서비스

악성코드 자동화 분석 시스템

팀: 시큐리티
팀원: 김근오 [91514660]
지도교수: 이병천

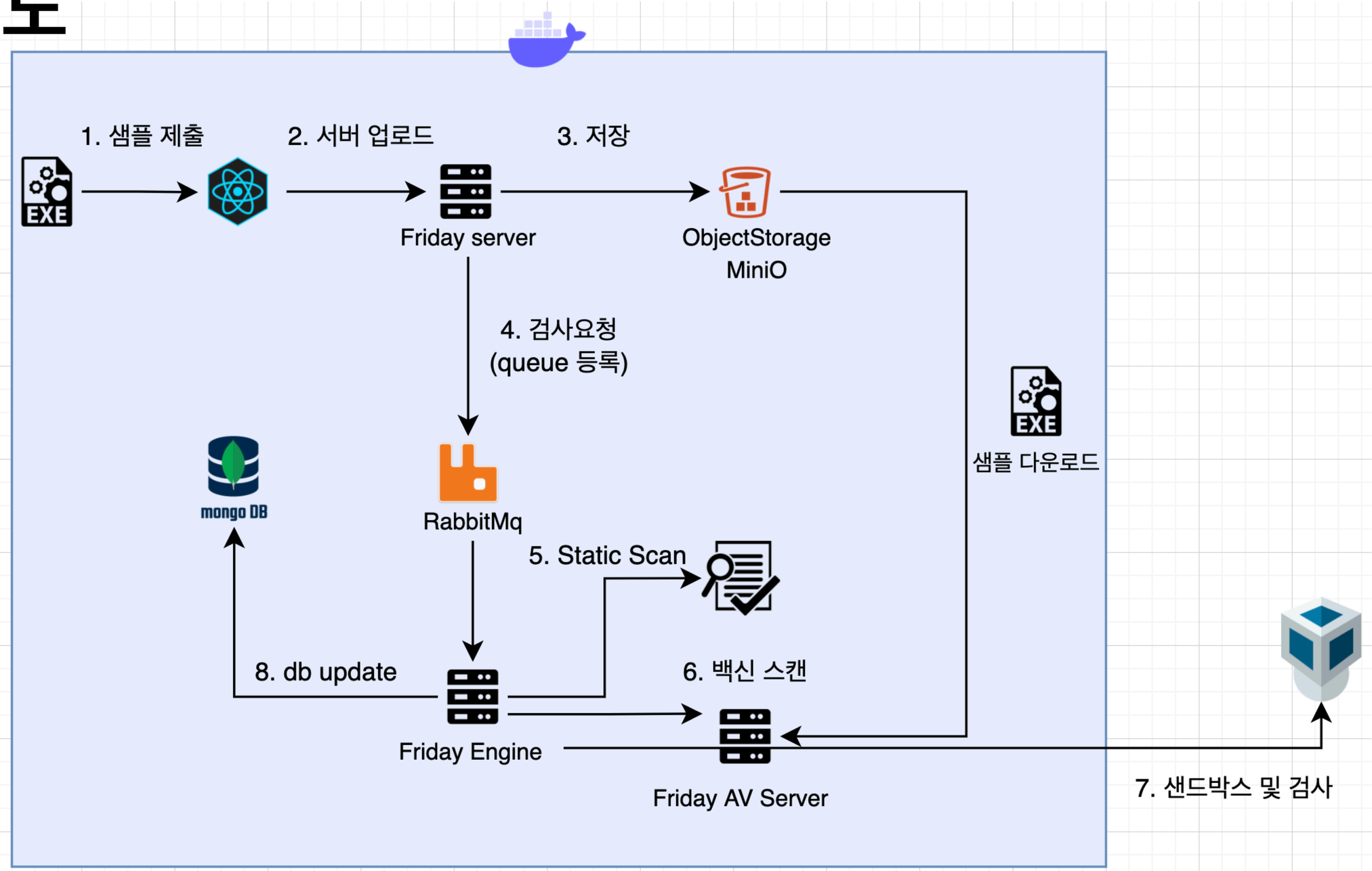
중부대학교

목차

추진 경과

수행업무 / 추진기간 (2021)	4월	5월	6월	7월	8월	9월	10월
자료조사 및 연구							
백앤드 API 설계							
DB 구축 및 프론트 앤드 개발							
테스트 및 도커화							

구상도


1. 샘플 제출
2. 서버 업로드
3. 저장
EXE
Friday server
ObjectStorage
MiniO
4. 검사요청
(queue 등록)
EXE
샘플 다운로드
mongo DB
RabbitMq
5. Static Scan
8. db update
6. 백신 스캔
Friday Engine
Friday AV Server
7. 샌드박스 및 검사

개발 환경 및 개발 내용 (1 / 8)

- Golang
- RabbitMq
- MiniO
- React.js
- Yara
- MongoDB
- Virtual Box
- Procmon
- Docker

개발 환경 및 개발 내용 (2 / 8)

```go
func (m *Mongo) FileSearch(ctx context.Context, sha256 string) (schema.File, error) {
    coll := m.client.Database(m.Config.DB).Collection(m.Config.FileCollection)
    var file schema.File
    err := coll.FindOne(ctx, bson.M{mongoSchema.FileSha256Key: sha256}).Decode(
    return file, err
}

func (m *Mongo) FileCreate(ctx context.Context, uploadFile schema.File) (schema
    coll := m.client.Database(m.Config.DB).Collection(m.Config.FileCollection)
    _, err := coll.InsertOne(ctx, uploadFile)
    if err != nil : schema.File{}, nil
    return uploadFile, nil
}

func (m *Mongo) FileUpdate(ctx context.Context, uploadFile schema.File) (schema
    coll := m.client.Database(m.Config.DB).Collection(m.Config.FileCollection)
    _, err := coll.UpdateOne(ctx, bson.M{"sha256": uploadFile.Sha256}, bson.M{
        "$set": uploadFile,
    })
    if err != nil : schema.File{}, nil
    return uploadFile, nil
}
```

```go
func NewRabbitMq(cfg RabbitMqConfig) (*RabbitMq, func(), error) {
    client, err := amqp.Dial(cfg.URI)
    if err != nil : nil, nil, err
    cleanup := func() {
        client.Close()
    }
    return &RabbitMq{
        Config: cfg,
        Client: client,
    }, cleanup, nil
}

func (r *RabbitMq) Channel() (*amqp.Channel, error) {
    ch, err := r.Client.Channel()
    if err != nil : nil, err
    return ch, nil
```

DB 및 RabbitMQ 선언

개발 환경 및 개발 내용 (3 / 8)

```go
func (s *Server) AmqpHandler(msg amqp.Delivery) error {
    body := bytes.ReplaceAll(msg.Body, []byte("NaN"), []byte("0"))
    var resp rabbitmq.ResponseObject
    if err := json.Unmarshal(body, &resp); err != nil {
        return errors.New( text: "ailed to parse message body")
        if err := msg.Reject( requeue: false); err != nil {
            logrus.Errorf("failed to reject message #{err}")
        }
    }
    filePath := filepath.Join(s.Config.TempPath, resp.Sha256)
    file, err := os.Create(filePath)
    if err != nil {
        logrus.Errorf("failed creating file #{err}")
    }
    ctx, cancel := context.WithTimeout(context.Background(), 30*time.Second)
    defer cancel()
    if err := s.minio.Download(ctx, resp.MinioObjectKey, file); err != nil {
        logrus.Errorf("failed downloading file #{err}")
    }
    file.Close()
    logrus.Infof("file downloaded to #{filePath}")

    res := schema.Result{}
    res.Status = processing
    var buff []byte

    // static 분석 작업 전 업데이트 [status 변경]
```

```go
func (s *Server) defaultScan(path string, res *schema.Result) {
    b, _ := ioutil.ReadFile(path)
    result := utils.ByteHashing(b)
    res.Size = int64(len(b))
    res.Ssdeep = result.Ssdeep
    res.Md5 = result.Md5
    res.Sha1 = result.Sha1
    res.Sha256 = result.Sha256
    res.Sha512 = result.Sha512
    res.Crc32 = result.Crc32
    logrus.Infof( format: "file hashing finished ")
    magic, err := magic.Scan(path)
    if err != nil {
        logrus.Errorf("magic scan failed with #{err}")
    }
    res.Magic = magic
    packerRes, err := packer.Scan(path)
    if err != nil {
        logrus.Errorf("packer scan failed with #{err}")
    }
    exif, err := exif.Scan(path)
    if err != nil {
        logrus.Errorf("exif scan failed with #{err}")
    }
    res.Exif = exif
```

RabbitMq 로 받아온 작업 (파일 정적 스캔)

개발 환경 및 개발 내용 (4 / 8)

```go
func getSha512(b []byte) string {
    hash := sha512.New()
    hash.Write(b)
    return hex.EncodeToString(hash.Sum( b: nil))
}

func getCrc32(b []byte) string {
    checkSum := crc32.ChecksumIEEE(b)
    hash := fmt.Sprintf("0x#{checkSum}")
    return hash
}

func getSsdeep(b []byte) (string, error) {
    return ssdeep.FuzzyBytes(b)
}

func ByteHashing(b []byte) CryptResult {
    fuzzy, err := getSsdeep(b)
    if err != nil && err != ssdeep.ErrFileTooSmall {
        log.Fatalf("ssdeep 실패 #{err}")
    }
    result := CryptResult{
        Md5:    getMd5(b),
        Ssdeep: fuzzy,
        Sha1:   getSha1(b),
        Sha256: getSha256(b),
        Sha512: getSha512(b),
        Crc32:  getCrc32(b),
```

```go
const tools = "exiftool"

func Scan(path string) (map[string]string, error) {
    args := []string{path}
    result, err := utils.CMD(tools, args...)
    if err != nil : nil, err
    return output(result), nil
}

func output(output string) map[string]string {
    var ignoreTags = []string{
        "Directory",
        "File Name",
        "File Permissions",
    }

    lines := strings.Split(output, sep: "\n")

    if utils.StringInSlice( a: "File not found", lines) : nil

    data := make(map[string]string, len(lines))

    for _, line := range lines {
        keyvalue := strings.Split(line, sep: ":")
        if len(keyvalue) != 2 {
            continue
        }
        if !utils.StringInSlice(strings.TrimSpace(keyvalue[0]), ignoreTags) {
            data[strings.TrimSpace(str.UpperCamelCase(keyvalue[0]))] = strings.TrimSpace(keyvalue[1])
        }
```

```go
const tools = "/opt/die/diec.sh"

func Scan(path string) ([]string, error) {
    args := []string{path}
    result, err := utils.CMD(tools, args...)
    if err != nil : nil, err
    return output(result), nil
}

func output(output string) []string {
    results := []string{}
    lines := strings.Split(output, sep: "\n")
    for _, line := range lines {
        if line != "" {
            results = append(results, line)
        }
    }
    return results
}
```

파일 해시화 및 기본 정보 스캔

개발 환경 및 개발 내용 (5 / 8)

```
switch engine {
case "comodo":
    res, err = comodo_client.ScanFile(comodo_api.NewComodoScannerCl:
case "clamav":
    res, err = clamav_client.ScanFile(clamav_api.NewClamAVScannerCl:
case "windefender":
    res, err = windefender_client.ScanFile(windefender_api.NewWinDe
case "drweb":
    res, err = drweb_client.ScanFile(drweb_api.NewDrWebScannerClien
}

if err != nil {
```

```
func (s *Server) parallelAvScan(path string) map[string]interface{} {

    comodoChannel := make(chan avs.ScanResult)
    clamavChannel := make(chan avs.ScanResult)
    winChannel := make(chan avs.ScanResult)
    drwebChannel := make(chan avs.ScanResult)

    go s.avScan( engine: "comodo", path, comodoChannel)
    go s.avScan( engine: "clamav", path, clamavChannel)
    go s.avScan( engine: "windefender", path, winChannel)
    go s.avScan( engine: "drweb", path, drwebChannel)

    avScanResults := map[string]interface{}{}
    engineCounts := 4
    count := 0
    for {
        select {
        case comodoResponse := <-comodoChannel:
            avScanResults["comodo"] = comodoResponse
            count++
        case clamavResponse := <-clamavChannel:
            avScanResults["clamav"] = clamavResponse
            count++
        case windefenderResponse := <-winChannel:
            avScanResults["windefender"] = windefenderResponse
            count++
        case drwebResponse := <-drwebChannel:
```

백신 서버로 악성코드 스캔 요청

개발 환경 및 개발 내용 (6 / 8)

```go
func (s *Server) RegisterHandlers() {
    api := s.Group( prefix: "/api")
    api.GET( path: "/", s.index)
    api.GET( path: "/system", s.system)
    api.POST( path: "/download", s.malwareDownload)
    api.POST( path: "/start", s.startAnalysis)
}
```

```go
if resp.FileType == "pe"{
    logrus.Infof( format: "job start")
    vm, err := virtualbox.GetMachine( id: "win7")
    if err != nil {
        logrus.Errorf("can not find machine #{err}")
    }
    logrus.Infof("#{vm.Name} sandbox found")
    logrus.Infof("cpu #{vm.CPUs}, memory #{vm.Memory}")

    logrus.Infof("VM 상태 : #{vm.State}")
    if vm.State == "poweroff"{
        s.VmRestore()
            {
            time.Sleep(1 * time.Second)
            if vm.State == "poweroff" || vm.State == "saved"{
                vm.State = "running"
                break
            }

        :ate == "running" || vm.State == "saved"{
        for {
            time.Sleep(1 * time.Second)
            s.VmStart()
            logrus.Infof("#{vm.Name} sandbox start")
```

```go
func (s *Server) startAnalysis(c echo.Context) error {
    var resp schema.ResponseObject
    if err := c.Bind(&resp); err != nil : err
    var file string
    s.checkProcmon()

    if resp.FileType == "pe" {
        file = fmt.Sprintf("#{resp.Sha256}.exe")
    }
    logrus.Infof("#{file} 분석 시작")
    s.processCapture()
    time.Sleep(time.Second * 5)

    var imageCapture []string
    var objectKeys []string
    path := fmt.Sprintf("#{s.Config.Volume}\\#{file}")

    imageCapture1 := time.NewTimer(time.Second * 10)
    go func() {
        <-imageCapture1.C
        imagePath := s.captureImage(resp.Sha256, num: 1)
        imageCapture = append(imageCapture, imagePath)
    }()

    s.startMalware(path)
    logrus.Infof("#{file} 악성코드 실행 완료")
```

VirtualBox 실행 및 Agent 로 스캔 요청

개발 환경 및 개발 내용 (7 / 8)

```
type DBModel struct {
    ScreenShots    []string          `json:"screen_shots"`
    ProcessCreate  []ProcessCreate   `json:"process_create"`
    CreateFile     []CreateFile      `json:"create_file"`
    ReadFile       []ReadFile        `json:"read_file"`
    RenameFile     []RenameFile      `json:"rename_file"`
    DeleteFile     []DeleteFile      `json:"deleted_file"`
    OpenRegKey     []OpenRegKey      `json:"open_reg_key"`
    GetRegKey      []GetRegKey       `json:"get_reg_key"`
    RegCreateKey   []RegCreateKey    `json:"create_reg_key"`
    SetRegValue    []SetRegValue     `json:"set_reg_value"`
    DeleteRegKey   []DeleteRegKey    `json:"deleted_reg_key"`
    DeleteRegValue []DeleteRegValue  `json:"deleted_reg_value"`
    UDP            []UDP             `json:"udp"`
    TCP            []TCP             `json:"tcp"`
    MalName        string            `json:"malware_name"`
    MalPid         string            `json:"malware_pid"`
    SubPid         string            `json:"sub_pid"`
}
```

```
type ProcessCreate struct {
    PID         string `json:"pid"`
    ChildPID    string `json:"child_pid"`
    ProcessName string `json:"process_name"`
    ProcessPath string `json:"process_path"`
    Operation   string `json:"operation"`
}
type CreateFile struct {
    PID         string `json:"pid"`
    ProcessName string `json:"process_name"`
    ProcessPath string `json:"process_path"`
    CreatePath  string `json:"create_path"`
}
type ReadFile struct {
    PID         string `json:"pid"`
    ProcessName string `json:"process_name"`
    ProcessPath string `json:"process_path"`
}
type RenameFile struct {
    PID         string `json:"pid"`
    ProcessName string `json:"process_name"`
    ProcessPath string `json:"process_path"`
    OriginName  string `json:"origin_name"`
    ChangeName  string `json:"change_name"`
```

프로세스 행동 추적을 위한 Data Struct

개발 환경 및 개발 내용 (8 / 8)

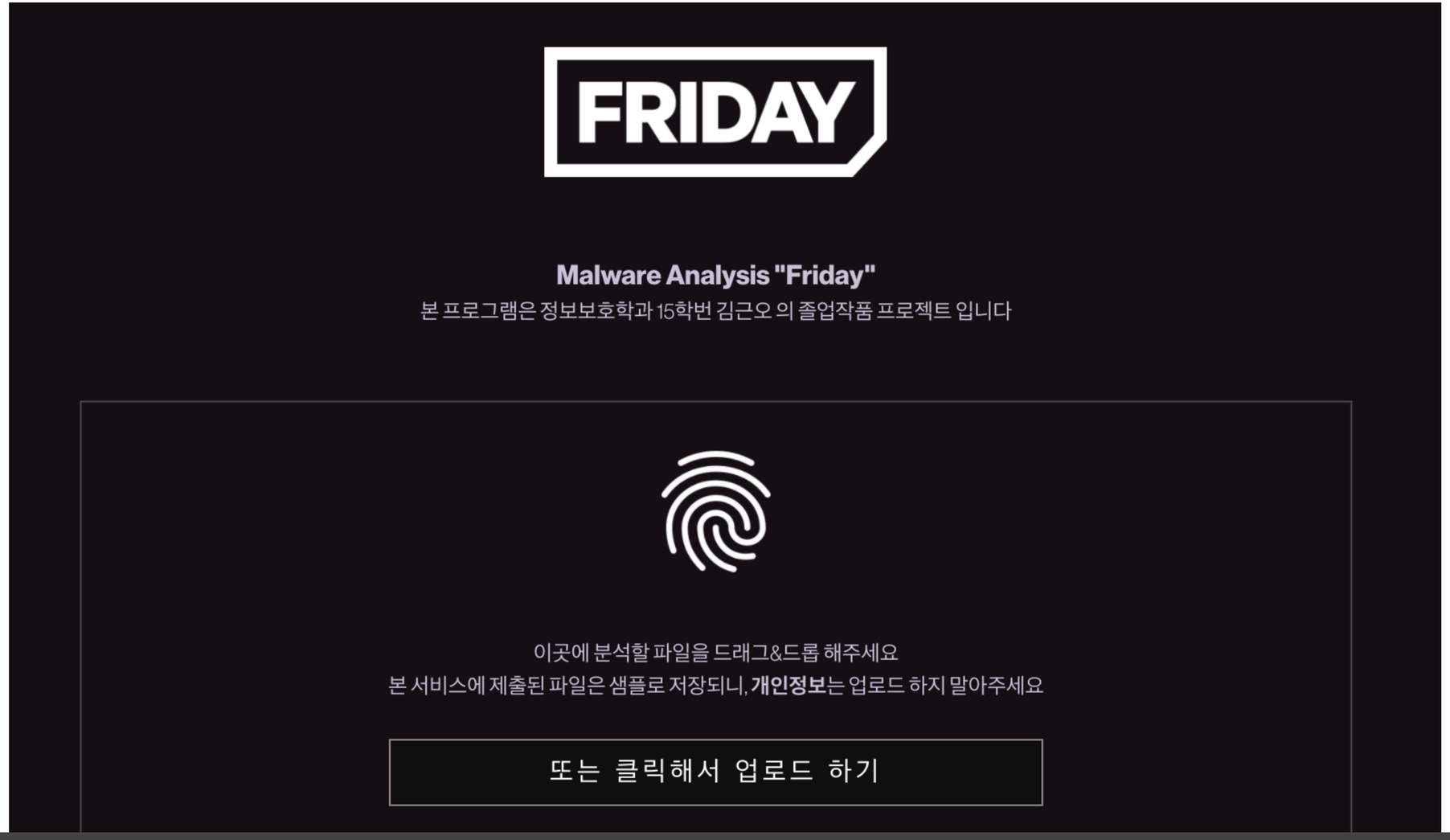
```tsx
const MainPage: React.FC = () => {
  const history = useHistory();
  const { getRootProps, getInputProps, open, acceptedFi
    useDropzone({
      noKeyboard: true,
      maxFiles: 1,
    });
  useEffect(() => {
    acceptedFiles.forEach(async (file: any) => {
      const UploadFileData = new FormData();
      UploadFileData.append('file', file);
      let resp = await API.post('/api/file', UploadFile
      let sha256 = resp.data.sha256;
      let file_key = resp.data.file_key;
      history.push(`/file-analysis/${sha256}`);
    });
  }, [acceptedFiles]);

  return (
    <div className={styles.root}>
      <header>
        <div className={styles.logo}>
          <img src="/logo.png" alt="logo" />
        </div>
      </header>
      <p className={styles.message}>
        <span style={{ fontSize: '15px', fontWeight: 'b
          Malware Analysis "Friday"
        </span>
        <br />본 프로그램은 정보보호학과 15학번 김근오 의 졸업작품 
        입니다
      </p>
```

```tsx
const AnalysisPage: React.FC = () => {
  const history = useHistory();
  const { sha256 } = useParams<{ sha256: string }
  const [analysisInfo, setAnalysisInfo] = useStat
  const [infected, setInfected] = useState<boolea
  const [size, setSize] = useState<string>();
  const [detectInfo, setDetectInfo] = useState<an
  const [status, setStatus] = useState<number>(0)
  const [infectedCount, setInfectedCount] = useSt
  const [engineCount, setEngineCount] = useState<
  const [activeIndex, setActiveIndex] = useState(
  const [progress, setProgress] = useState(282.74

  const tabClickHandler = (index: number) => {
    setActiveIndex(index);
  };

  const tabContArr = [
    {
      tabTitle: (
        <li
          key="detect"
          className={`${styles.tab} ${
            activeIndex === 0 ? styles.is_active
          }`}
          onClick={() => tabClickHandler(0)}
        >
          <div className={styles.tab_link}>탐지</d
        </li>
```

```tsx
      </div>
      <div className={styles.detections_info_body}>
        <div className={styles.row}>
          <div className={styles.object_info}>
            <div className={styles.id}>
              <div>
                SHA-256: {' '}
                <b>{analysisInfo && analysisInfo.sha256}</b>
              </div>
              <div className={styles.file_sub_title}>
                파일명: {' '}
                <b>{analysisInfo && analysisInfo.file_key}</b>
              </div>
            </div>
          </div>
          <div className={styles.object_sub_info}>
            <div className={styles.file_sub_}>{size}</div>
            <div className={styles.file_sub_title}>파일 사이즈</div>
          </div>
          <div className={styles.object_sub_info}>
            <div className={styles.file_sub_}>
              {analysisInfo && analysisInfo.submissions.length}
            </div>
            <div className={styles.file_sub_title}>제출 횟수</div>
          </div>
          <div className={styles.object_sub_info}>
            <div className={styles.file_sub_}>
              {analysisInfo &&
                moment(
```

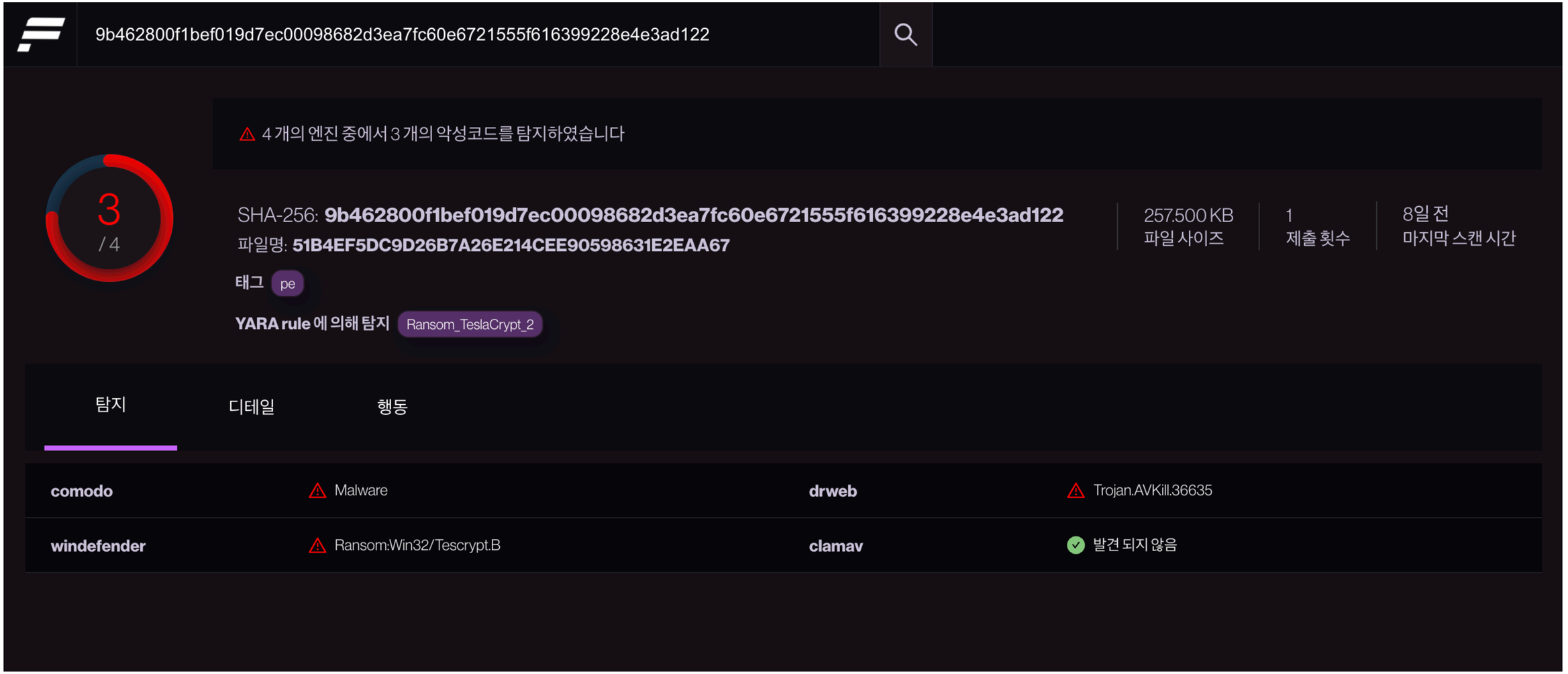
프론트앤드

개발 시스템 운영



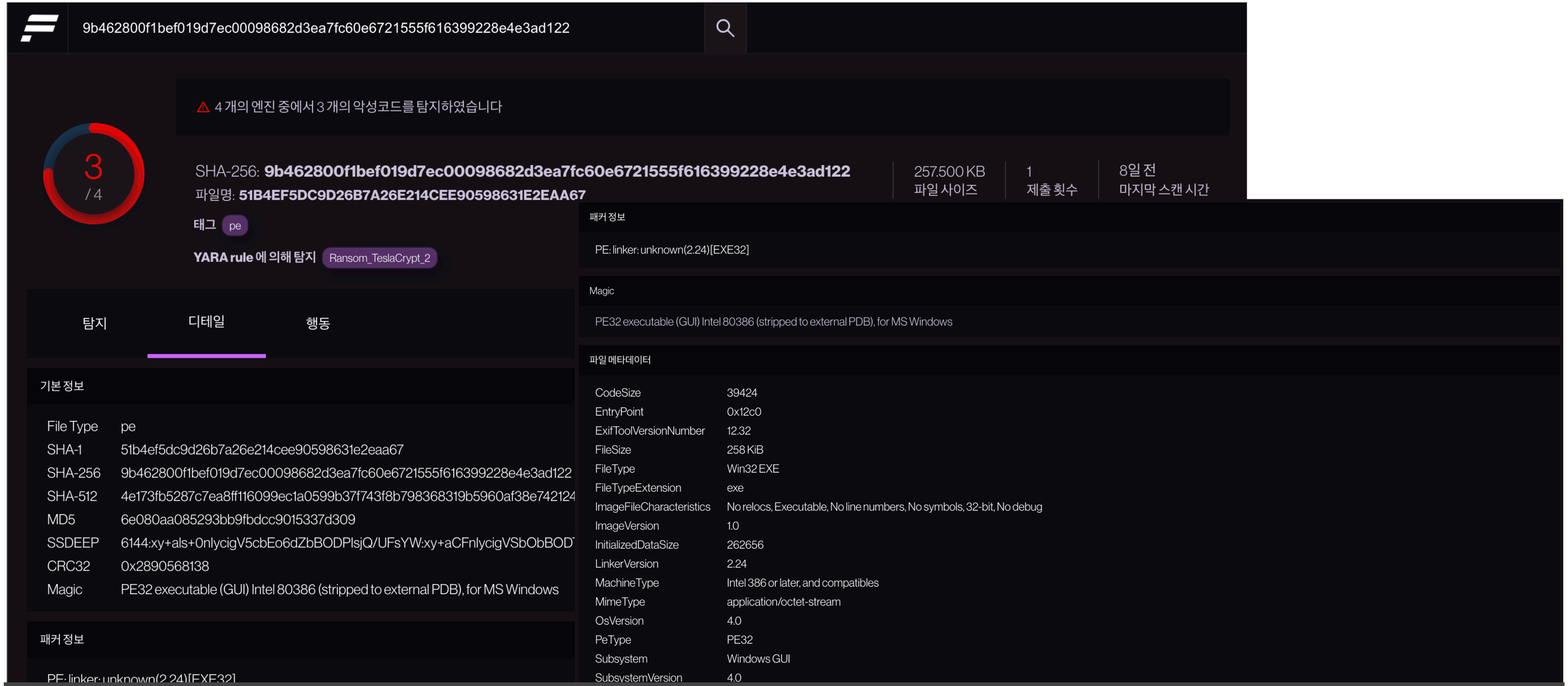
샘플 파일 제출 화면

개발 시스템 운영 (2 / 5)



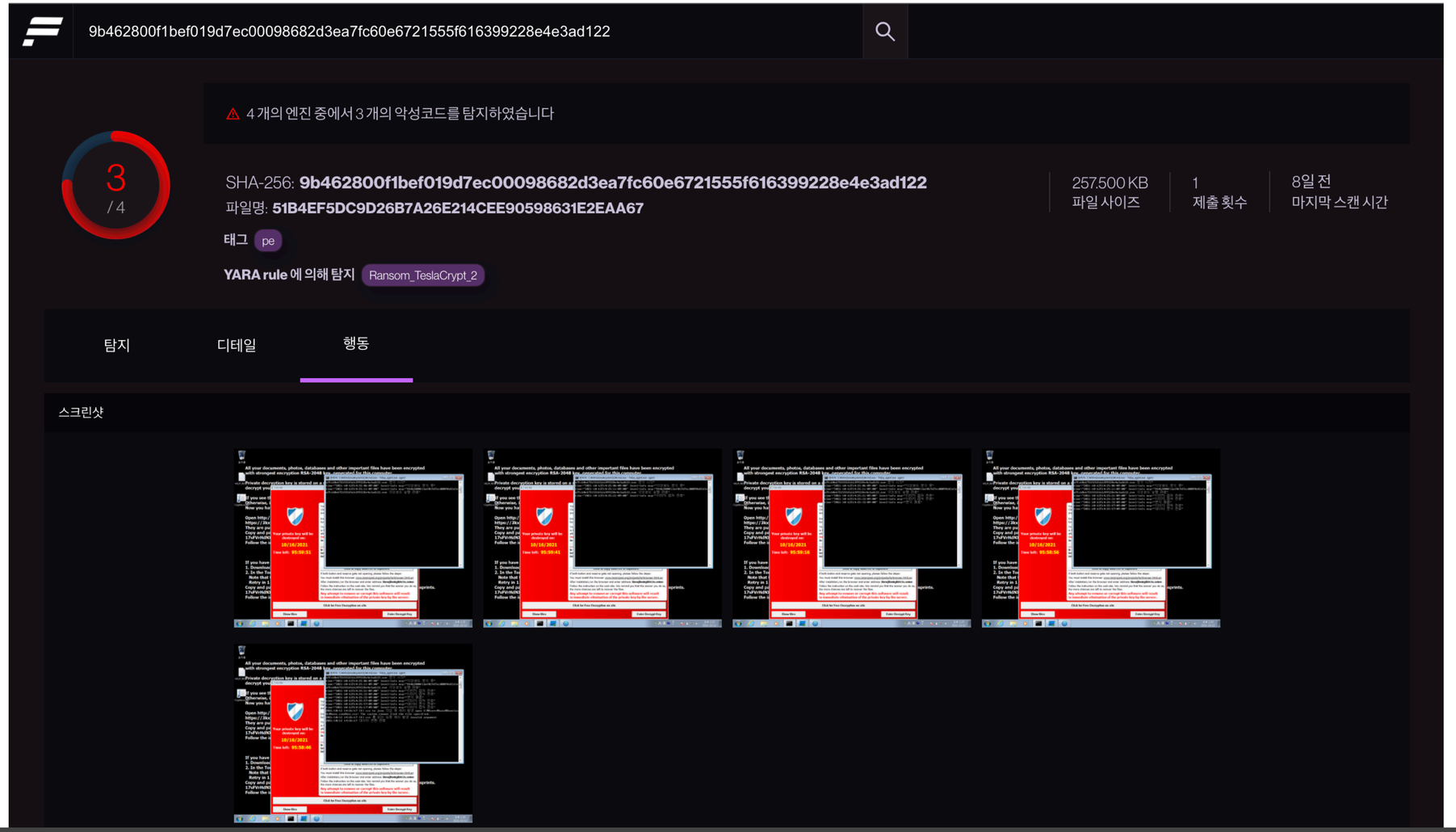
백신 검사 결과 제공 화면

개발 시스템 운영 (3 / 5)



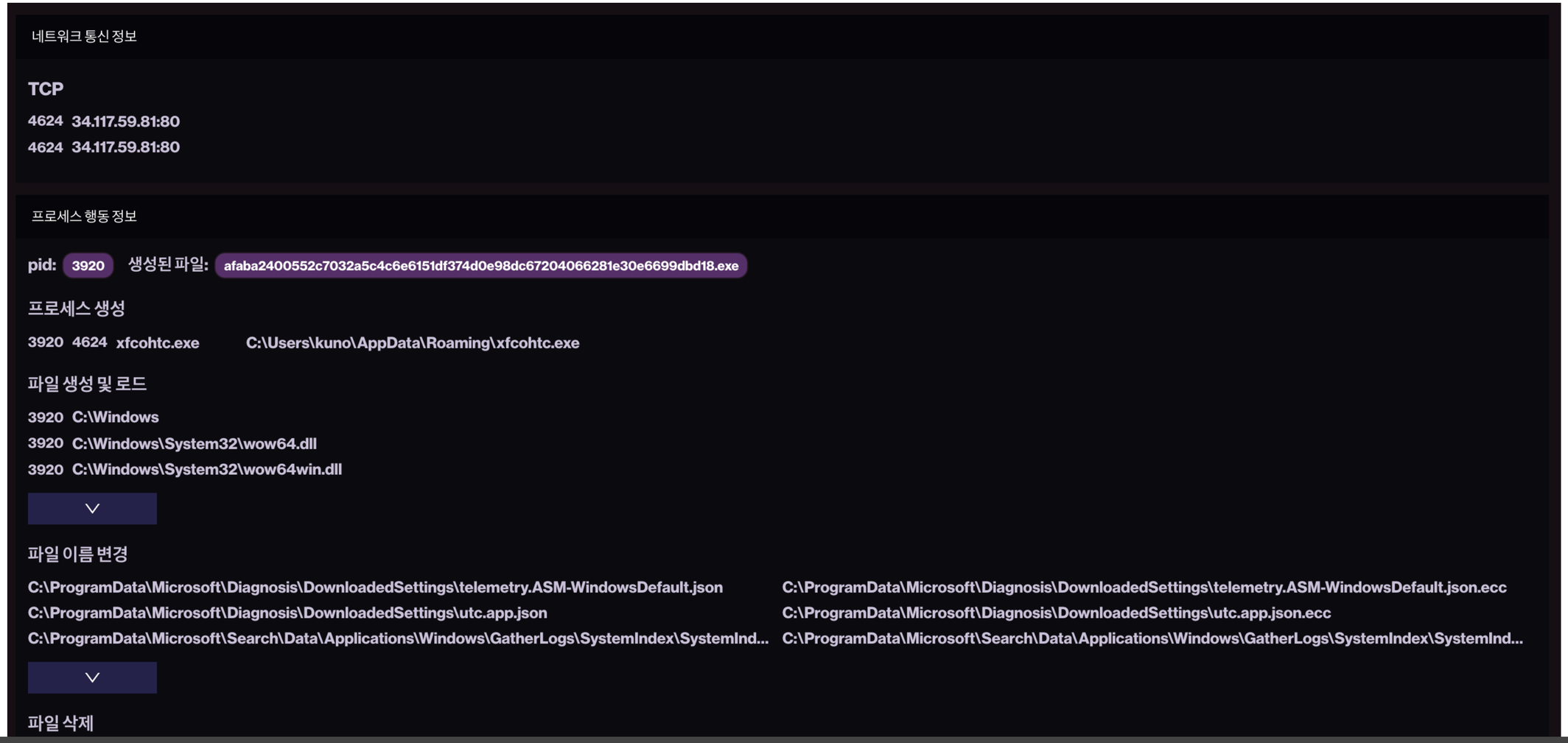
기본 정적 스캔 결과정보 화면

개발 시스템 운영 (4 / 5)



VirtualBox 스크린샷 제공 및 악성코드 활동 화면 (1 / 2)

개발 시스템 운영 (5 / 5)



VirtualBox 스크린샷 제공 및 악성코드 활동 화면 (1 / 2)

결론 및 기대 효과

결　론

- 백신 및 실제 샌드박스에서 악성코드의 행동을 분석하는 시스템을 구현

기대 효과

- 악성코드 분석 및 연구를 할 때 좀 더 쉽게 행동을 추적 가능

Q&A

감사합니다