Command and control server - C2

João Marques (20210408), Maria Martins (20211010), Mário Nascimento (20210387) e Rebeca Sampaio (20211332)

Universidade Europeia, IADE – curso de Engenharia Informática

Trabalho realizado no âmbito da unidade curricular de Sistemas Operativos sob a supervisão do professor Pedro Rosa

Lisboa, 03 de março de 2024

Link para o repositório GitHub: https://github.com/Command-and-Control/C2

Descrição do Problema a Resolver

Este projeto tem como objetivo o desenvolvimento de um sistema de comando e controlo (C2), concebido especificamente para facilitar a interação de equipas de especialistas em segurança informática.

Através deste sistema, é possível gerir de forma eficaz um vasto número de máquinas comprometidas, potencializando a realização de testes de penetração mais eficientes, ao melhorar significativamente a comunicação e a coordenação entre os operadores e os sistemas alvo. Mais específicamente, os casos de uso serão os seguintes:

- Operações de Teste de Penetração: Utilizando o sistema C2 para coordenar ataques simulados a infraestruturas, identificando vulnerabilidades e testando a eficácia das medidas de segurança existentes, gerando um relatório da segurança da infraestrutura.
- Formação e Educação em Cibersegurança: Empregar o sistema em ambientes controlados para treinar especialistas em segurança, melhorando as suas competências em operações ofensivas e defensivas e ajudando-os a construir relatórios de teste de penetração.
- Investigação de Cibersegurança: Utilização do sistema para estudar técnicas de ataque e defesa, abstraindo as partes básicas de manter acesso, controlar máquinas, fazer *pivoting*, persistência, etc. permitindo o foco nas áreas importantes.

Descrição da Solução a Implementar

Descrição genérica da solução a implementar

A arquitetura do sistema baseia-se na integração de um servidor C2, a sua interface para os atacantes e *beacons*, promovendo uma comunicação bidirecional. Os *beacons*, uma vez implementados nos sistemas comprometidos, estabelecem um canal de comunicação contínuo com o servidor C2, ficando à espera de instruções.

A interação entre os operadores e o sistema C2 é mediada por uma interface de texto, que inclui uma interface de *shell* personalizada, desenhada para proporcionar uma experiência de utilizador superior, juntamente com a habilidade de

executar comandos customizados, adequados às exigências específicas de cada operação.

Para além disso, o sistema enriquece-se com a incorporação de módulos personalizados, englobando funcionalidades como a implementação de *keylogging*, técnicas de manutenção de acesso (persistência), escalamento de privilégios e procedimentos de enumeração, entre outros. Ainda será possível controlar todas as máquinas em modo *botnet*. Os *beacons* serão *environment-aware* pelo que não executarão se detetarem um ambiente virtualizado ou de teste.

Enquadramento nas áreas da Unidade Curricular

Este projeto será incluído nas Unidades Curríclares de Sistemas Operativos e Compliadores. Iremos utilizar as duas UC's da seguinte forma:

- Sistemas Operativos:
    - Gestão de processos e threads
    - Multiprocessing and multithreading. O desenvolvimento dos beacons, a execucao dos payloads customizados, entre outras partes do projeto envolvem criação, execução e terminação de vários processes e threads e a sua manutenção.
    - Comunicação IPC: A comunicação bidirecional entre o C2 e os beacons, assim como os modulos personalizados e coordenação e execução de tarefas distribuídas depending de uma boa comunicação inter-processos.
    - Gestão de recursos: É necssário gerir bem o consumo de recursos do sistema operativo para evitar a deteção e a sobrecarga do sistema.
    - Segurança e isolamento: As técnicas implementadas para evitar deteção e análise de sistemas virtualizados, assim como as técnicas usadas nos payloads e modulos necesitarão de uma interação muito forte com as APIs internas e não-documentadas do sistema operativo.
- Compiladores:
    - Criação de código: Os módulos personalizados e beacons poderão correr em vários sistemas operativos e têm que ser

executados dinamicamente, pelo que envolve conceitos de compilação, mais especificamente a interpretação de código.

- Optimização de código: Para reduzir o overhead e melhorar a evasão, utilizaremos várias técnicas de optimização de código de forma a reduzi-lo.
- Análise Lexica e sintática: As interfaces de shell terão de utilizar estes conceitos de forma a permitir comandos complexos e personalizados

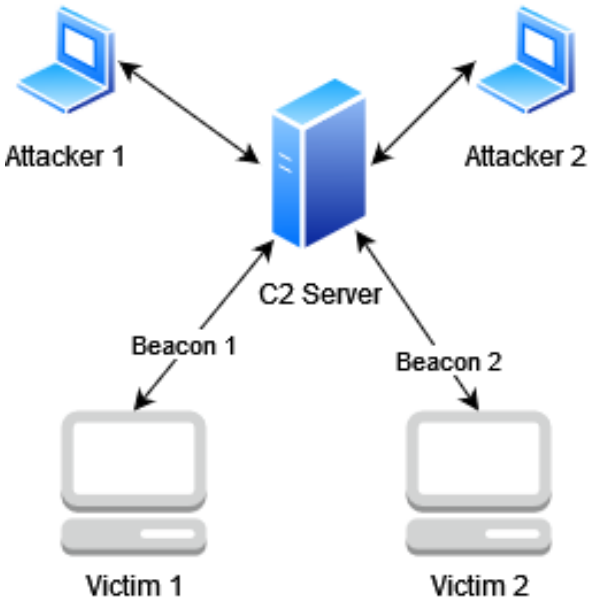
Requisitos Técnicos para desenvolvimento do projeto

- Python: Para o desenvolvimento da lógica principal do servidor C2, scripts de automação e integração de módulos.
- C e Assembly: Para a criação de payloads e módulos personalizados que exigem execução de baixo nível e eficiência máxima, especialmente em operações que envolvem escalamento de privilégios, keylogging, evasão de detecção, ou um espaço reduzido.
- Conhecimento em Redes e Protocolos de Comunicação: Essencial para a configuração de comunicações seguras e eficientes entre o servidor C2, beacons, e clientes. Também importante para a criação dos vários módulos que envolvam a rede, enumeração ou evasão através do uso de diferentes protocolos, incluindo protocolos personalisados.
- Criptografia: Para proteger as comunicações e dados, será necessário implementar técnicas de criptografia e autenticação e possivelmente desenvolver sistemas de criptografia customizados para comunicações entre beacons e o servidor C2.

Requisitos

id	Descrição	Categoria
RF1	Comunicação bidirecional entre beacon e C2	Must have
RF2	Interface shell personalizada	Must have
RF3	Controlo individual e de grupo sobre as máquinas	Must have
RF4	Deteção de ambientes virtualizados e de teste	Must have
RF5	Criação de um relatório de estado de segurança da maquina	Must have
RF6	Modulos customizados, permitindo keylogging, exfiltração de dados	Must have
RF7	Comunicação bidirecional entre cliente e C2	Should have
RF8	Criação de interface de texto	Should have
RF9	Capacidade de interpretar código personalidado como payloads	Should have
RF10	Modulos de ransomware e enumeração	Should have
RF11	Movimento lateral	Nice to have
RF12	Escalamento de privilégios	Nice to have
RF13	Mecanismos contra reverse-engineering	Nice to have
RF14	Interpretador de linguagem de programação personalizada	Nice to have
RF15	Comunicação segura com RSA + AES256	Nice to have
RF16	Integração com plataformas de segurança	Nice to have

Arquitetura da Solução

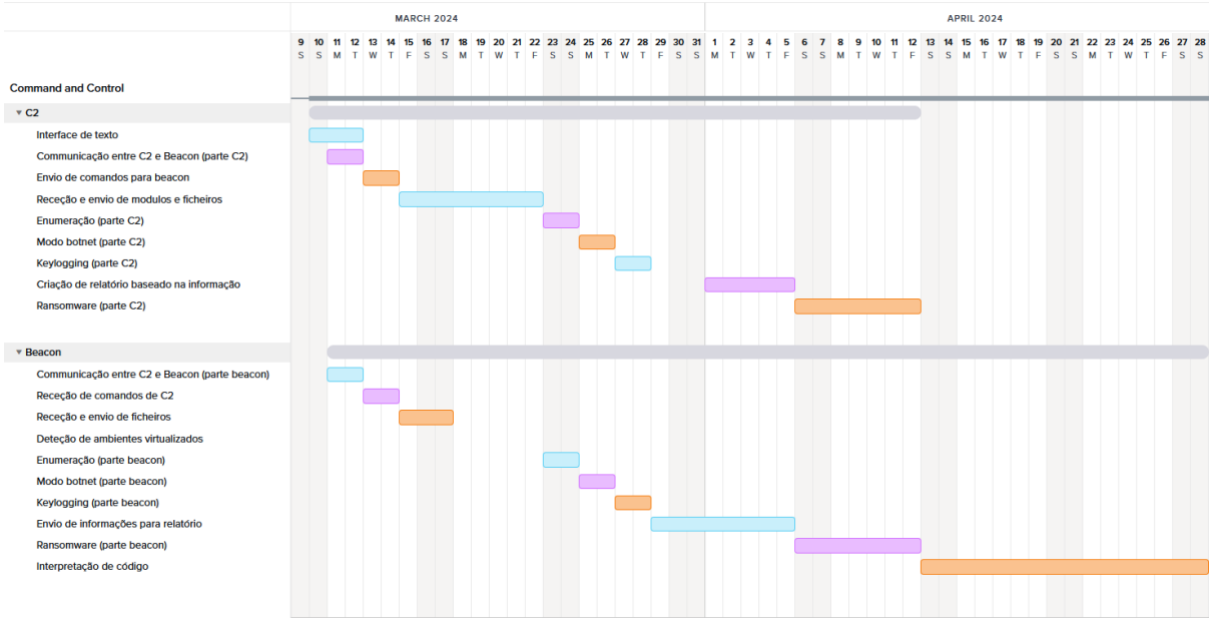


Tecnologias a utilizar

- VSCode: Para o desenvolvimento em Python, C e Assembly.
- Git: Para controlo de versão e colaboração entre o grupo de desenvolvimento.

- SSL/TLS: Para criptografar comunicações entre o servidor C2 e os beacons, garantindo a segurança dos dados em trânsito e melhorando a evasão.
- OpenSSL: Biblioteca para implementação de protocolos de criptografia e para a criação de certificados digitais.

Calendarização - Gráfico de Gantt



Bibliografia

Anley, C., & Koziol, J. (Eds.). (2007). *The shellcoder's handbook: Discovering and exploiting security holes* (2. ed). Wiley.

C2 Frameworks—Red Canary Threat Detection Report. (n.d.). Red Canary. Retrieved March 2, 2024, from https://redcanary.com/threat-detection-report/trends/c2-frameworks/

Eisenberg, D. A., Alderson, D. L., Kitsak, M., Ganin, A., & Linkov, I. (2018). Network Foundation for Command and Control (C2) Systems: Literature Review. *IEEE Access*, *6*, 68782–68794. https://doi.org/10.1109/ACCESS.2018.2873328

Erickson, J. (2008). *Hacking: The art of exploitation* (2nd ed). No Starch Press.