# CVE-2024-21378 в MS Outlook

*Александр Родченко*

В феврале этого года была обнаружена уязвимость СVE-2024-21378, при успешной эксплуатации которой удалённый злоумышленник может выполнить произвольный код в системе, где установлен MS Outlook. Я проанализировал процесс эксплуатации этой уязвимости и написал утилиту, которая позволяет детектировать атаки с использованием CVE-2024-21378.

## Вид уязвимости

С точки зрения сложности мониторинга (то есть на взгляд аналитика SOC), уязвимости удалённого выполнения кода (RCE) можно разделить на три категории:

1. те RCE, что вызывают старт процесса,
2. те, что загружают библиотеку/скрипт в уязвимый процесс,
3. и те, которые просто тихонько хостят недетектящийся шеллкод в памяти и не отсвечивают лишний раз.

По этой классификации CVE-2024-21378 – уязвимость второго типа. В оригинальном [листе](https://gist.github.com/Homer28/7f3559ff993e2598d0ceefbaece1f97f), кстати, в качестве полезной нагрузки использовалась [библиотека](https://github.com/Homer28/easy_shellcode_generator) с шеллкодом, который только лишь «резолвил» хост вида "new.d%USERDOMAIN%.u%COMPUTERNAME%.attacker.com". Это лишний раз подчёркивает, что эксплоит достаточно беспалевный: нет старта никакого «злого» процесса. А значит, никакие дефолтные правила на старт процесса от Outlook нам не помогут.

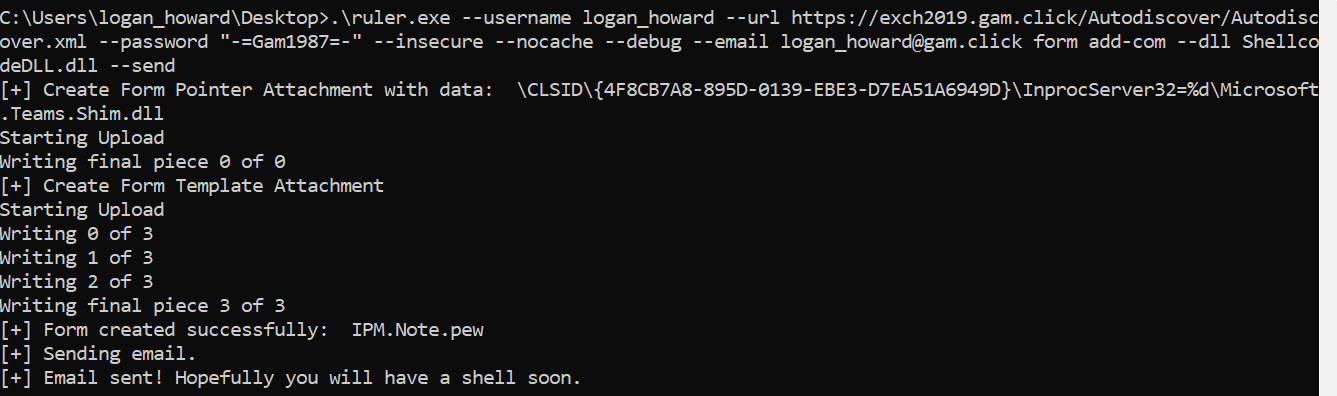
Давайте разберёмся, что же происходит? Если кратко, то Exchange и Outlook общаясь по протоколу MAPI, могут пересылать друг другу не только сообщения ( MessageClass “IPM.Note ”), но и контакты, приглашения на встречи, задачи, логи, отчёты и [многое другое](https://learn.microsoft.com/en-us/office/vba/outlook/concepts/forms/item-types-and-message-classes), зачастую вообще [немыслимое](https://learn.microsoft.com/en-us/office/vba/outlook/concepts/forms/form-name-and-message-class-overview).

В частности, существует возможность создать свой собственный тип сообщения и указать клиенту Outlook, что для его отрисовки следует использовать какую-то особую форму. Эту форму также можно передать в виде сообщения. Такому сообщению соответствует класс [IPM.Microsoft.FolderDesign.FormsDescription](https://github.com/NetSPI/ruler/blob/44d1b60a343054e11607db7c6485dd02774a729d/forms/rulerforms.go#L121), а сама форма представляет собой DLL-библиотеку, которая хранится как вложение в этом сообщении.

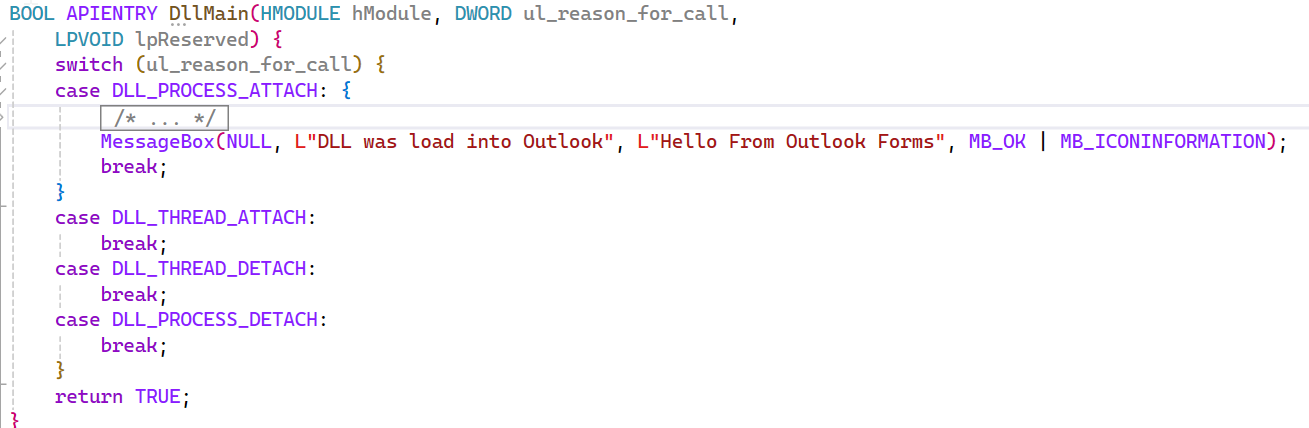
Это даже сложно назвать уязвимостью – по всей видимости, именно так и планировалось для удобства. Другое дело, что в этой библиотеке может быть вредоносный код, который Outlook выполнит без всяких проверок.

## Как проводится атака

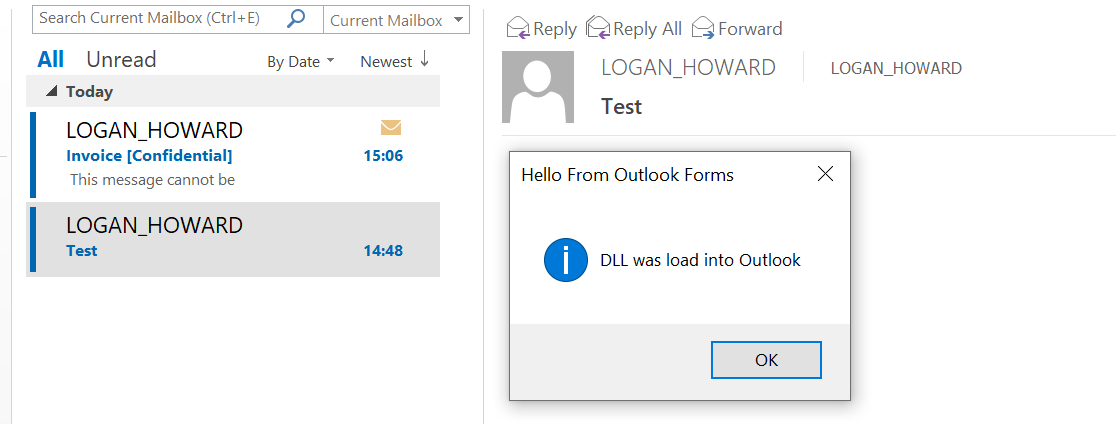
Чтобы проэксплуатировать данную уязвимость, злоумышленник создаёт кастомную форму (библиотека Shellcode.DLL) и посылает её жертве:



В нашем тестовом примере ShellcodeDLL не делает ничего особенного – просто выводит сообщение в Message Box:



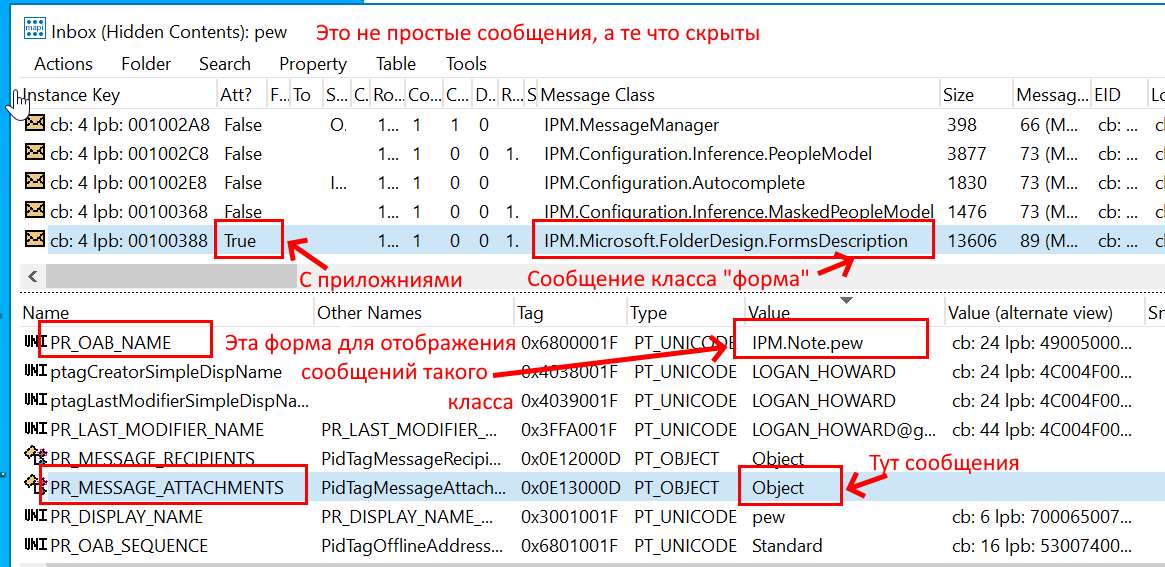
После отправки формы, которая открывает определённый класс сообщений, злоумышленник посылает жертве письмо-триггер, принадлежащее к заданному классу сообщений. В результате при открытии Outlook видим выполнение отправленной библиотеки:



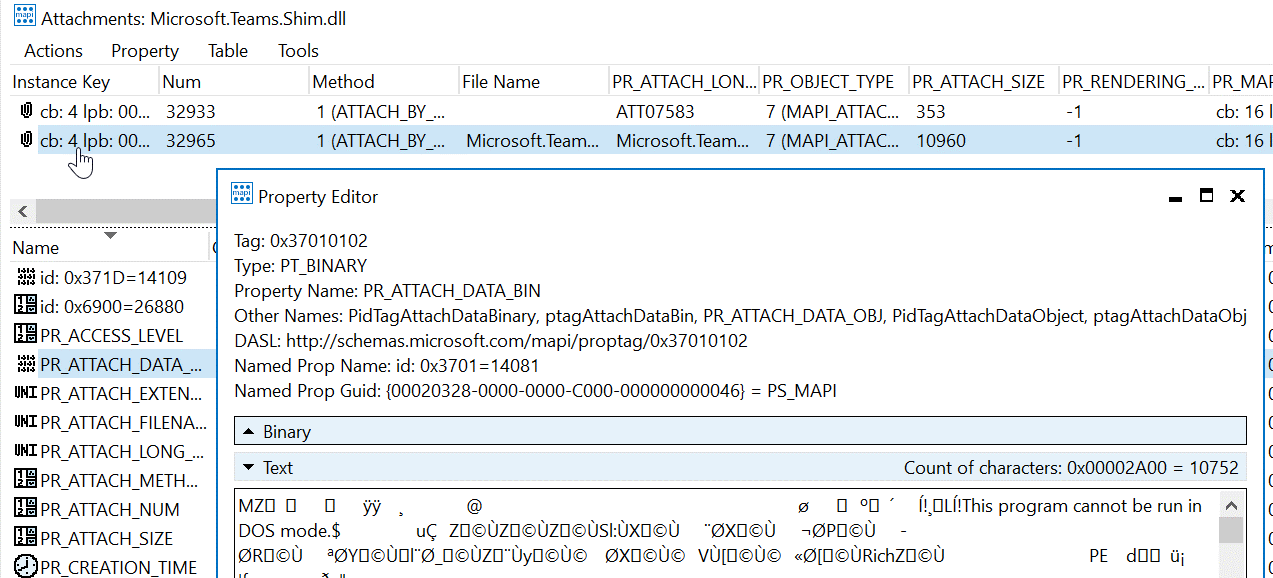
## Как найти вредоносную форму

Как уже было сказано, письмо с дефолтной темой «Invoice [Confidential]» не содержит в себе ничего злодейского. Это лишь триггер, который загрузит форму. А где же сама форма?

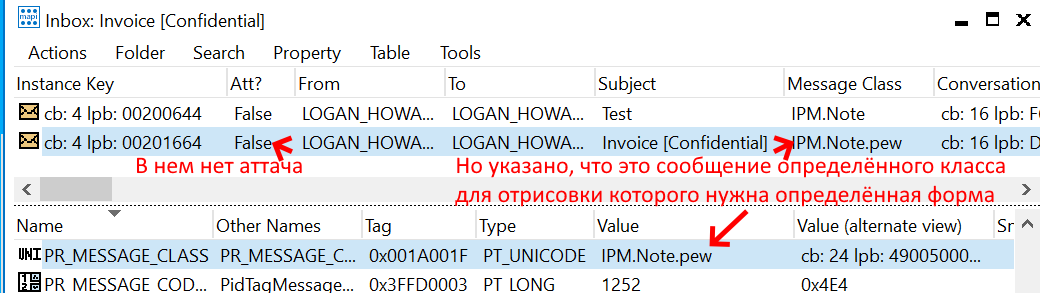
Воспользуемся MFCMAPI и посмотрим на сообщение с формой:



Можно увидеть, что во вложении к письму содержится библиотека:

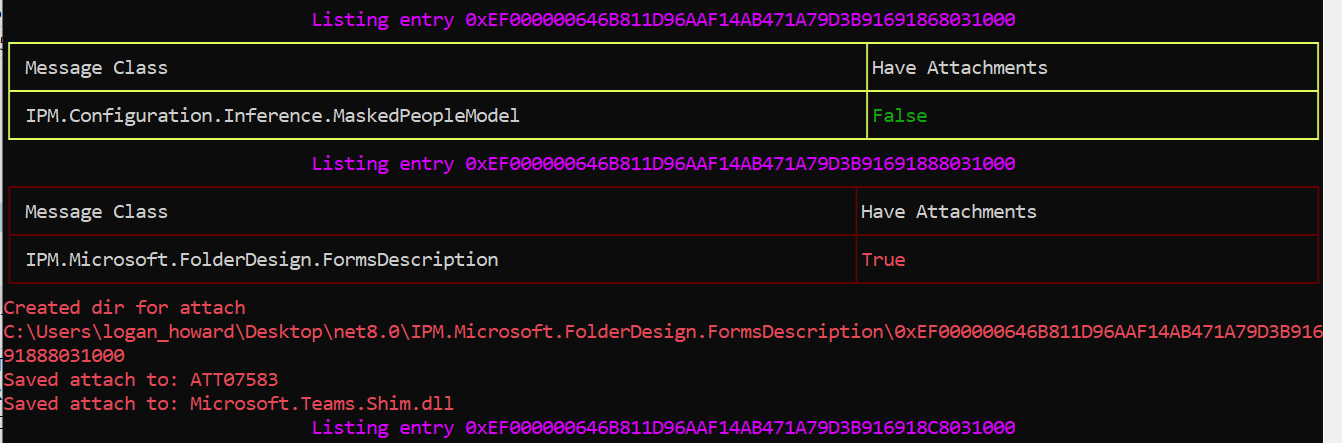


Теперь посмотрим на письмо с триггером:



## Детектирование попыток эксплуатации

Я написал [утилиту](https://github.com/gam4er/OutlookFormFinder), которая сканирует скрытые сообщения (это различные служебные сообщения, включая правила обработки писем) в папке "Входящие", и при обнаружении в них вложения сохраняет его в текущую директорию (CWD):



[Здесь хорошо бы дописать еще пару строчек текста, чтобы статья не обрывалась просто скриншотом. Например – что делать после того, как утилита нашла вредоносную форму? Или можно сформулировать какой-то краткий общий вывод статьи. Кстати, а они исправили эту уязвимость? Если есть обновление, можно про это написать тоже]