You Have One New Appwntment: Exploiting iCalendar Properties in Enterprise Applications

Eugene Lim

July 2022

Abstract

First defined in 1998, the iCalendar standard remains ubiquitous in enterprise software. However, it did not account for modern security concerns and allowed vendors to create proprietary extensions that expanded the format’s attack surface. I demonstrate how flawed RFC implementations led to vulnerabilities in popular enterprise applications. Attackers can trigger exploits remotely with zero user interaction due to automatic parsing of event invitations. Furthermore, I explain how iCalendar’s integrations with the SMTP and CalDAV protocols enable multi-stage attacks. Despite attempts to secure these technologies separately, the interactions that arise from features such as emailed event reminders require a “full-stack” approach to calendar security. I conclude that developers should strengthen existing iCalendar standards in both design and implementation.

iCalendar, the Little Standard that Could

The iCalendar standard was born out of necessity in 1998, when the growth of enterprise calendar software created a need for a single, interoperable format. Indeed, the abstract of RFC 2445, the first standard for iCalendar, warned that “Current group scheduling and Personal Information Management (PIM) products are being extended for use across the Internet, today, in proprietary ways.”

With the support of the major enterprise software vendors (Frank Dawson from Lotus Notes and Derik Stenerson from Microsoft authored RFC 2445), iCalendar became the de-facto standard across enterprise applications. RFC 5545 (2009) and RFC 7986 (2016) further refined the standard.

Even though iCalendar is a simple text-based format, software developers have extended and built upon this foundation to produce a plethora of advanced functionality, such as click-to-join video calls and traveling time reminders.

However, at the heart of all this complexity lies a legacy standard that did not originally consider the multitude of modern security and privacy concerns. Additionally, despite the standard’s noble aspirations to reduce proprietary fragmentation, vendors have created numerous extensions to iCalendar that further expand the attack surface.

With the rise of remote work, a reckoning is due as millions of applications and devices exchange iCalendar files daily. Although both Red Team operators and threat actors have exploited iCalendar event invitations to execute phishing attacks, vulnerability researchers have yet to subject the iCalendar format to deep scrutiny. One exception is then-NCC Group’s Andy Grant, who discovered a file exfiltration vulnerability in Apple Calendar on MacOS (CVE-2020-3882). He concluded that “I certainly did not look at them all and there’s plenty for others to explore” - this turned out to be true.

By throwing a wide net across flawed RFC implementations, poorly documented proprietary properties, and interactions with other protocols, I discovered multiple vulnerabilities in Apple Calendar, Google Calendar,
Microsoft Outlook, VMware Boxer, and more. This paper will discuss these vulnerabilities and their root causes before tackling higher-level questions about securing legacy standards. I will end with observations about the shift towards proprietary calendar APIs and its implications.

**Brief Background on the iCalendar Standard**

The most relevant standard for iCalendar today is RFC 5545 “Internet Calendaring and Scheduling Core Object Specification (iCalendar)”, supplemented by RFC 7986 “New Properties for iCalendar”.

In short, the iCalendar file format is a text-based file organized into content lines that define the properties of the calendar. Each property can also include semicolon-separated parameters that define additional attributes about the property. Since the format is straightforward, it would be easiest to illustrate this with an example from https://icalendar.org/:

```plaintext
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//ZContent.net//Zap Calendar 1.0//EN
CALSCALE:GREGORIAN
METHOD:PUBLISH
BEGIN:VEVENT
SUMMARY:Abraham Lincoln
UID:c7614cfe-3549-4a00-9152-d25cc1fe077d
SEQUENCE:0
STATUS:CONFIRMED
TRANSP:TRANSPARENT
RRULE:FREQ=YEARLY;INTERVAL=1;BYPMONTH=2;BYMONTHDAY=12
DTSTART:20080212
DTEND:20080213
DTSTAMP:20150421T141403
```

Figure 1: Click-to-Join Video Call Reminder
We see here that the root component is the VCALENDAR, which can contain VEVENT components. There are other components, such as VTIMEZONE, VTODO, and VALARM, the last of which is a good example of a “dangerous by default” standard.

### Dangerous by Default

VEVENT components can include multiple VALARM components, which define a reminder for the event. VALARM components must include the ACTION property, which determine how the reminder if triggered. RFC 5545 allows for the following actions:

1. **AUDIO**: Specifies an alarm that plays a sound to alert the user. The optional ATTACH property points to an audio sound file.
2. **DISPLAY**: Displays the text value of the DESCRIPTION property to the user.
3. **EMAIL**: Sends an email to all the addresses specified by the ATTENDEE properties in the VALARM component. The DESCRIPTION property determines the body text of the message, and the SUMMARY property the subject. ATTACH properties add attachments to the message.

```
BEGIN:VALARM
TRIGGER;VALUE=DATE-TIME:19970317T133000Z
REPEAT:4
DURATION:PT15M
ACTION:AUDIO
ATTACH;FMTTYPE=audio/basic:ftp://example.com/pub/sounds/bell-01.aud
END:VALARM

BEGIN:VALARM
TRIGGER:-PT30M
REPEAT:2
DURATION:PT15M
ACTION:DISPLAY
DESCRIPTION:Breakfast meeting with executive team at 8:30 AM EST.
END:VALARM

BEGIN:VALARM
TRIGGER;RELATED=END:-P2D
ACTION:EMAIL
ATTENDEE:mailto:john_doe@example.com
SUMMARY:*** REMINDER: SEND AGENDA FOR WEEKLY STAFF MEETING ***
DESCRIPTION:A draft agenda needs to be sent out to the attendees to the weekly managers meeting (MGR-LIST). Attached is a pointer the document template for the agenda file.
ATTACH;FMTTYPE=application/msword:http://example.com/templates/agenda.doc
END:VALARM
```
Already, we can see how these actions could pose security risks in unsafe implementations. If a user accepts an event with an `EMAIL` alarm, the attacker could automatically send emails with attachments on the victim’s behalf once the alarm triggers. In another scenario, an `AUDIO` alarm could open arbitrary files from remote locations!

RFC 5545 recommends the following mitigation:

Note: Implementations should carefully consider whether they accept alarm components from untrusted sources, e.g., when importing calendar objects from external sources. One reasonable policy is to always ignore alarm components that the calendar user has not set herself, or at least ask for confirmation in such a case.

Nevertheless, this is not an official requirement of the standard. Even today, Apple Calendar supports a deprecated `ACTION` type: the `PROCEDURE` alarm. This opens any file using a default application determined by Launch Services, as noted by Grant.⁵

In short, RFC 5545 defines behavior that users would consider dangerous by default in today’s security context. Most major vendors have wised up to these issues and mitigate or ignore parts of the specification entirely. For example, Microsoft ignores the `VALARM ACTION` property on import.⁶ However, when they follow the standard blindly, the impact can be devastating. For example, Grant’s CVE-2020-3882 exploit manipulated the `ATTACH`, `ORGANIZER`, and `ATTENDEE` properties to send meeting invitations that contained attachments from a victim’s system. I discovered more examples.

### Insecure Implementations of Standard RFC Properties

One issue I observed over time was that iCalendar suffered from “one standard, many interpretations” - whether due to imprecise phrasing or vendors’ initiative, vendors could implement even relatively simple properties such as `DESCRIPTION` very differently. This not only undermined the interoperability of the calendar files, but also introduced fresh vulnerabilities where there had been none previously.

#### VMware Boxer Stored XSS via `DESCRIPTION`

The `DESCRIPTION` property contains text descriptions of the associated `VEVENT` or `VTODO` component - this often appears as a rich text input for calendars. In practice, implementations vary widely. For example, Microsoft specifically interprets `DESCRIPTION` as plain text, relying instead on the non-standard `X-ALT-DESC` property for rich text (HTML). Meanwhile, Google Calendar has no qualms with HTML in `DESCRIPTION`.

In the case of VMware Boxer, version 21.11 of the iOS client in December 2021 added rich text display of the `DESCRIPTION` field⁷, but failed to consider the implications of doing so, creating a stored XSS vulnerability. Attackers could chain the XSS with deeplinks to launch unwanted applications and actions.

The loose definition of the property in RFC 5545 leaves it open to interpretation, creating opportunities for such mistakes to occur. Furthermore, the fact that this only occurred in the iOS client demonstrates the difficulty of implementing iCalendar securely across multiple platforms and clients.

#### Disclosure Timeline

- 06-02-2022: Initial disclosure
- 07-02-2022: Acknowledgement
- 23-02-2022: Initial security advisory and patch (CVE-2022-22944)

---

⁵https://research.nccgroup.com/2020/05/05/exploring-macos-calendar-alerts-part-1-attempting-to-execute-code/

⁶https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-oxcical/7fd3531e-05f4-4cf5-9100-0638aa9794de

Synology Calendar Stored XSS via ATTENDEE/ORGANIZER

Even if implemented properly, naive assumptions about specific fields can also lead to vulnerabilities. For example, Synology Calendar properly parsed the mailto: prefix for ORGANIZER and ATTENDEE values but failed to sanitize the actual address, creating a stored XSS vulnerability when it displayed this field in the guestlist. Despite a comprehensive Content-Security Policy, a single loophole via data: script sources enabled exploitation. Thanks to the default behavior of automatically adding event invitations to calendar, this created a wormable stored XSS.

BEGIN:VEVENT
CREATED:20220319T190656
LAST-MODIFIED:20220319T190656
DTSTAMP:20220319T190656
UID:20220319T190656-8f7a05620192.168.50.106
SEQUENCE:2
SUMMARY:Hack
TRANSP:OPAQUE
DTSTART;TZID=Asia/Singapore:20220320T193000
DTEND;TZID=Asia/Singapore:20220320T203000
DESCRIPTION:Hack
LOCATION:Hack
ORGANIZER;PARTSTAT=ACCEPTED;ROLE=CHAIR:mailto:
  <iframe srcdoc="<script src=data:%3a%3btext/javascript,alert
  ("${document.domain} hacked by spaceraccoon")"></script>"></iframe>
  @test.local
ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-ACTION;ROLE=REQ-PARTICIPANT:mailto:
  <iframe srcdoc="<script src=data:%3a%3btext/javascript,alert
  ("${document.domain} hacked by spaceraccoon")"></script>"></iframe>
END:VEVENT

Disclosure Timeline

- 20-03-2022: Initial disclosure
- 20-03-2022: Acknowledgement
- 17-05-2022: Initial public release
- 12-07-2022: Disclosed vulnerability details (CVE-2022-22682)

Apple Calendar Custom URI via DESCRIPTION

Meanwhile, other vendors have built entirely new parsers on top of existing properties. Apple Calendar embeds FaceTime calls in DESCRIPTION using a particular format and set of control characters:

DESCRIPTION:----( FaceTime )----

\[FaceTime\] \nhttps://facetime.apple.com/

join#v=1&p=...&k=...

This appears as a clickable link on MacOS and iOS.

However, attackers can replace the benign FaceTime URL with non-HTTPS URIs, including file://. ftp://, and more. Depending on the chain involved, this could lead to arbitrary code execution - not a far-fetched scenario given the recent UXSS in Safari8 or Gatekeeper bypass.9 On iOS, custom URIs could launch arbitrary applications and their associated shortcuts.

Apple Calendar also implements similar parser logic for Zoom event invitations. By tacking on “format-in-a-format” custom parsing logic, applications risk unnecessary complexity and harm interoperability with other calendar applications.

---

8https://www.ryanpickren.com/safari-uxss
Figure 2: Synology Stored XSS

Figure 3: Apple Calendar Clickable FaceTime Call
BEGIN:VEVENT
DTSTART;TZID=Asia/Singapore:20220227T142000
DTEND;TZID=Asia/Singapore:20220227T150000
TRANSP:OPAQUE
ORGANIZER;CN="Eugene Lim":mailto:eugenelim@example.com
UID:B4E6A2FD-8AED-417C-8D00-B0263637A47D
DTSTAMP:20220127T043925Z
X-APPLE-TRAVEL-ADVISORY-BEHAVIOR:AUTOMATIC
DESCRIPTION:----( FaceTime )----
[FaceTime]
FiLe://
---===---
SEQUENCE:1
SUMMARY:Important
LAST-MODIFIED:20220127T044419Z
CREATED:20220127T033947Z
ATTENDEE;CN="Eugene Lim";CUTYPE=INDIVIDUAL;EMAIL="eugenelim@example.com"
;PARTSTAT=ACCEPTED;ROLE=CHAIR:mailto:eugenelim@example.com
ATTENDEE;CN="Eugene Lim";CUTYPE=INDIVIDUAL;EMAIL="eugenelim@example.co"
;PARTSTAT=NEEDS-ACTION;RSVP=TRUE:mailto:eugenelim@example.co
END:VEVENT

Disclosure Timeline

- 27-01-2022: Initial disclosure
- 27-01-2022: Acknowledgement
- 16-05-2022: Released security advisory and patch (macOS Monterey 12.4)
Apple Calendar Automatic Download via ATTACH

Sometimes, non-standard behavior can result in interesting side effects without amounting to a full-fledged vulnerability. While testing for bypasses for the CVE-2020-3882 patch, I noticed that Apple Calendar automatically downloads any ATTACH with a URI value of *.icloud.com regardless of the owner of the uploaded iCloud file. This could deliver malicious payloads without user interaction. Furthermore, Apple Calendar followed redirects, allowing for de-anonymization attacks when chained with an open redirect on *.icloud.com.

Vulnerable Proprietary Extensions to iCalendar

Despite the difficulty of safely implementing just the standard properties, vendors have pressed ahead with creating their own proprietary non-standard properties. RFC 5545 provides a “standard mechanism for doing non-standard things” using properties prefixed with X-. However, it did not institute a central registration authority for these extension properties. As a result, there are many poorly documented or undocumented proprietary properties used in the wild today, with varying levels of support. Unsurprisingly, these have also created their own security issues.

Historical Microsoft Outlook Proprietary Properties

Before diving into modern proprietary properties, it is useful to consider the evolution of Microsoft Outlook’s approach. In the initial years of the enterprise application wars, Microsoft Outlook took full advantage of non-standard properties to differentiate itself. In fact, Microsoft pioneered multiple calendar and live meeting integrations long before Teams. Released in 1995, Microsoft NetMeeting allowed users to collaborate on documents with voice and chat. After discontinuing it, Microsoft released new videoconferencing products such as Windows Meeting Space, Microsoft Office Live Meeting, Skype for Business, and finally Microsoft Teams. These products integrated into Outlook’s calendar and used proprietary iCalendar properties.

Some of these legacy properties started with NetMeeting, which featured interesting properties such as X-MS-OLK-AUTOSTARTCHECK and X-MS-OLK-COLLABORATEDOC. These properties allowed an event to automatically start a meeting and open an Office document. After first intended as a feature, today users would consider such behavior a significant vulnerability - imagine receiving a meeting invitation that automatically started a video conference without user interaction!

A quick perusal of an Outlook-generated NetMeeting calendar event should trigger interesting ideas for exploitation:

BEGIN:VEVENT
ATTENDEE;CN=johndoe@example.com;RSVP=TRUE:mailto:johndoe@example.com
CATEGORIES:Gifts,Goals/Objectives
CLASS:PUBLIC
CREATED:20220225T130431Z
DTEND:20220226T053000Z
DTSTAMP:20220226T035947Z
DTSTART:20220226T050000Z
LAST-MODIFIED:20220225T130431Z
LOCATION:Test
ORGANIZER;CN="Jane Doe":mailto:janedoe@example.com
PRIORITY:5
SEQUENCE:0
SUMMARY:Test
TRANSP:OPAQUE
UID:0400000008200E000745B7101A82E00800000000000CEAC3D822AD8010000000000000000
0100000005E6554276593344A5F7590817490760
X-MICROSOFT-CDO-BUSYSTATUS:BUSY
For example, what happened if `X-MS-OLK-COLLABORATEDOC` pointed to an executable, an SMB file share, or a URL? Could `X-MS-OLK-DIRECTORY` be manipulated to leak authentication data?

**Microsoft Outlook Custom URI via X-MS-OLK-MWSURL**

While current versions of Outlook no longer support many of these proprietary properties, they remain dormant in legacy code, which attackers can trigger in interesting ways. For example, Outlook 365 no longer explicitly supports Meeting Workspace, but the `X-MS-OLK-MWSURL` property still activates a “View Meeting Workspace” option in the reminder popup that can launch any custom URIs, including SMB shares. This is a standard vector to leak Net-NTLMv2 hashes. Furthermore, custom URI to code execution chains continue
to crop up from time to time and remains a highly active area of research.\textsuperscript{10}

![Figure 6: View Meeting Workspace Option](image)

Other than launching a custom URI, responding to any event with X-MS-OLK-MWSURL sends a HTTP SOAP request to the specified URL instead of an email response, which attackers could exploit for de-anonymization attacks.

```
POST /mwsurl/_vti_bin/meetings.asmx HTTP/1.1
Authorization: Bearer
X-MS-CookieUri-Requested: t
X-FeatureVersion: 1
X-IDCRL_ACCEPTED: t
Content-Type: text/xml; charset=utf-8
SOAPAction: http://schemas.microsoft.com/sharepoint/soap/meetings/SetAttendeeResponse
X-Office-Version: 16.0.14827
User-Agent: Microsoft Office/16.0 (Windows NT 10.0; Microsoft Outlook 16.0.14827; Pro)
Host: example.burpcollaborator.net
Content-Length: 818
Connection: Keep-Alive
Cache-Control: no-cache

<?xml version="1.0"?>
<q:Envelope xmlns:q="http://schemas.xmlsoap.org/soap/envelope/"

<q:Header>
</q:Header>

<q:Body>
<mt:SetAttendeeResponse
xmlns:mt="http://schemas.microsoft.com/sharepoint/soap/meetings/"

<mt:attendeeEmail>johndoe@example.com</mt:attendeeEmail>
<mt:recurrenceId>0</mt:recurrenceId>
```

\textsuperscript{10}https://positive.security/blog/ms-officecmd-rce
Disclosure Timeline

- 28-01-2022: Initial disclosure
- 28-01-2022: Acknowledgement
- 31-01-2022: Triaged
- 23-02-2022: Validated and resolved

Google Calendar Arbitrary Iframe via X-GOOGLE-CALENDAR-CONTENT-*

Another example of “zombie properties” is Google Calendar’s X-GOOGLE-CALENDAR-CONTENT-* properties:

- X-GOOGLE-CALENDAR-CONTENT-TITLE
- X-GOOGLE-CALENDAR-CONTENT-ICON
- X-GOOGLE-CALENDAR-CONTENT-URL
- X-GOOGLE-CALENDAR-CONTENT-TYPE
- X-GOOGLE-CALENDAR-CONTENT-WIDTH
- X-GOOGLE-CALENDAR-CONTENT-HEIGHT
- X-GOOGLE-CALENDAR-CONTENT-DISPLAY

These properties generated Google Calendar gadgets - iFramed widgets in event popups. While Google deprecated this functionality in 2018, gadgets continued to work in the embedded classic version of Google Calendar. By embedding arbitrary iFrames and icons, attackers could de-anonymize the user or pop JavaScript phishing prompts. Indeed, the ability to embed remote content in calendar events poses privacy and security risks in today’s context.

For example, the following iCalendar event would load an icon from https://a.com and an iFrame from https://evilhost.com/exploit.html. By inviting victims to this event, which would automatically add it to their calendars as a “transparent” event invitation, an attacker could remotely exploit this vulnerability without user interaction.

BEGIN:VCALENDAR
CALSCALE:GREGORIAN
METHOD:PUBLISH
X-WR-CALNAME;VALUE=TEXT:Word of the Day
VERSION:2.0
BEGIN:VEVENT
DTSTART;VALUE=DATE:20220228
DTEND;VALUE=DATE:20220229
SUMMARY:My Payload
X-GOOGLE-CALENDAR-CONTENT-TITLE:My Payload
X-GOOGLE-CALENDAR-CONTENT-ICON:https://a.com
X-GOOGLE-CALENDAR-CONTENT-URL:https://evilhost.com/exploit.html
In short, proprietary properties continue to haunt vendors long after deprecation. The variety of clients and use-cases vendors must support contributes to the difficulty of securing these non-standard extensions to the iCalendar format.

Disclosure Timeline
- 28-02-2022: Initial disclosure
- 28-02-2022: Acknowledgement
- 01-03-2022: Triaged
- 03-03-2022: Accepted
- 23-03-2022: Bounty awarded

Binary-Encoded Apple Calendar Proprietary Properties
Although RFC 5545 supports binary content through URIs and Base64 encoding, Apple Calendar takes it one step further by adding proprietary binary formats. For example, various properties associated with travel data in Apple Calendar, such as flights, contain base64-encoded data.

X-APPLE-SUGGESTION-INFO-OPAQUE-KEY:|flightA2|SFO|JFK|A242|DUMMY
X-APPLE-SUGGESTION-INFO-UNIQUE-KEY:|2|flightA2|SFO|JFK|A242|DUMMY|18|com.apple.Safari|AC4EB118-B1B3-4743-9B07-0B53C9FD5112
X-APPLE-STRUCTURED-LOCATION;VALUE=URI;X-ADDRESS="San Francisco, CA 94128, United States";X-APPLE-MAPKIT-HANDLE=CAES/gEIrkQqPw6Lk1YSzARoSCWm55hjozkJAEQAAPSumF7AI8KDVVuaXR1ZCBtZGFOZXMSA1VTGjpDYWxpZm9yYmlhIGJDQSoJU
Base64-decoding the value of X-APPLE-STRUCTURED-DATA reveals the magic bytes bplist00 at the start of the decoded data. This matches Apple’s binary property list format. Attempting to convert it to XML with plutil reveals that it also uses the NSKeyedArchiver plist format:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>$version</key>
  <integer>100000</integer>
  <key>$archiver</key>
  <string>NSKeyedArchiver</string>
  <key>$top</key>
  <dict>
    <key>root</key>
    <dict>
      <key>CF$UID</key>
      <integer>1</integer>
    </dict>
  </dict>
  <key>$objects</key>
  <array>
    ...
  </array>
</dict>
</plist>
```

Due to the difficulty in analyzing NSKeyedArchiver files, it is necessary to convert it into a more readable key-value format. I used Allyn Stott’s unarchive-plist tool for this.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>SGEventMetadataKey</key>
  <dict>
    <key>SGEventMetadataBundleIdKey</key>
    <string>com.apple.Safari</string>
    <key>SGEventMetadataCategoryDescriptionKey</key>
    <string>Flight</string>
    <key>SGEventMetadataConfidenceKey</key>
    <integer>1</integer>
    <key>SGEventMetadataEventActivitiesKey</key>
    <array/>
  </dict>
</dict>
</plist>
```

---

The decoded plist is far more readable and uses the FlightReservation schema to contain flight information. Meanwhile, X-APPLE-SUGGESTION-INFO-UNIQUE-KEY, X-APPLE-STRUCTURED-LOCATION, X-APPLE-MAPKIT-HANDLE, and other properties also contain location-related data in undocumented formats. These are interesting targets for fuzzing and further research.

Interactions with SMTP and CalDAV

While the iCalendar format helped standardize calendar files, it did not address remote protocols to synchronize calendars between clients and servers. Vendors published RFC 4791 “Calendaring Extensions to WebDAV” in 2007 to align the nascent proprietary solutions to this problem.¹²

Several companies have implemented calendar access protocols using HTTP to upload and download iCalendar [RFC2445] objects, and using WebDAV to get listings of resources. However, those implementations do not interoperate because there are many small and big decisions to

be made in how to model calendaring data as WebDAV resources, as well as how to implement required features that aren't already part of WebDAV. This document proposes a way to model calendar data in WebDAV, with additional features to make an interoperable calendar access protocol.

Besides CalDAV, the RSVP functionality described in RFC 5545 necessitated further integration with the SMTP email protocol to send and receive scheduling changes.

Due to the multitude of parsers and protocols involved in these complex interactions, mistakes inevitably occur. While it is not within the scope of this white paper to describe these protocols in detail, I will highlight the key characteristics that led to vulnerabilities.

NextCloud Calendar SMTP Command Injection via ORGANIZER

Security researcher Inti de Ceukelaire has produced excellent research on SMTP vulnerabilities arising from exploiting standard RFC implementations. When I discussed the iCalendar RFC with him, he pointed out the risk of newline characters in ATTENDEE and ORGANIZER emails. The SMTP standard allows for special characters via quoted strings in email addresses.

```
Quoted-string  = DQUOTE *QcontentSMTP DQUOTE

QcontentSMTP  = qtextSMTP / quoted-pairSMTP

quoted-pairSMTP = %d92 %d32-126
                   ; i.e., backslash followed by any ASCII
                   ;     graphic (including itself) or SPace

qtextSMTP      = %d32-33 / %d35-91 / %d93-126
                   ; i.e., within a quoted string, any
                   ;     ASCII graphic or space is permitted
                   ; without backslash-quoting except
                   ;     double-quote and the backslash itself.
```

For example, RFC-compliant parsers consider "%0d%0aContent-Length:%200%0d%0a%0d%0a"@example.com and "recipient@test.com>\nRCPT TO:<victim+"@test.com valid email addresses. This becomes an issue when applications pass newlines directly to SMTP command pipelining, which allows SMTP clients to send multiple commands in multiline batches instead of a “one command-one response” structure. Since most iCalendar implementations automatically send an email to the ORGANIZER email when the recipient RSVPs to an event invitation, poisoned event invitations could exploit this behavior to trigger the injection.

Indeed, by inserting newlines in ATTENDEE and ORGANIZER emails, I discovered that automated email responses to calendar invitations caused SMTP command injections in NextCloud Calendar. The first instance of the vulnerability occurred in the appointment feature, while the second instance occurred in the email inbox via iCalendar file attachments. For example, the following simple proof-of-concept would inject the EHLO SMTP command:

```
BEGIN:VCALENDAR
CALSCALE:GREGORIAN
VERSION:2.0
PRODID:-//Nextcloud Mail
BEGIN:VEVENT
CREATED:20220319T044448Z
DTSTAMP:20220319T080250Z
SEQUENCE:2
```

[13]https://www.youtube.com/watch?v=Bpnc1-g3fMk
The attacker would send this as an email attachment to the victim. The iCalendar attachment activated additional menus within NextCloud Mail inbox to add it to the victim's calendar. When the victim clicked those menus, the backend automatically sent an email RSVP to janedoe@example.com without sanitizing the newlines, triggering the SMTP injection. The eventual impact depended on the commands supported by the backend SMTP server.

Disclosure Timeline

- 19-03-2022: Initial disclosure
- 19-03-2022: Acknowledgement
- 21-03-2022: Triaged
- 23-03-2022: Validated and resolved
- 12-04-2022: Initial security advisory and patch (CVE-2022-24838)

Back to the Future: Recommendations and Next Steps

Despite the initial goal of interoperability, vendors have increasingly sought to piggyback on the iCalendar standard to introduce their own proprietary features. One consistent trend is vendors replacing iCalendar (and in extension CalDAV) altogether with their own proprietary APIs. In 2013, Google made a premature attempt to deprecate CalDAV in favor of the Google Calendar API but backtracked after opposition from developers.15 More recently, Microsoft has begun to transition to Microsoft Graph, a JSON-based REST API that unites data and services across its Microsoft 365 ecosystem, including calendars. This appears as JSON values in proprietary iCalendar properties.

X-MICROSOFT-SKYPETEAMSPROPERTIES:{"cid":"19:meeting_HGU4PVR1ZWYtNmM1MC00ZTBmEFjkNmPtYTJkD2Q0NTdjNGY2@thread.v2","private":true,"type":0,"mid":0,"rid":0,"uid":null}
X-MICROSOFT-LOCATIONS:[{"DisplayName":"https://us04web.zoom.us/j/8102702073?pwd=2s4cdHxxaKpL-J6DhUnRY14V7uTDwE.2&from=addon","LocationAnnotation":"","LocationUrl":"","LocationStreet":"","LocationCity":"","LocationState":"","LocationCountry":"","LocationPostalCode":"","LocationFullAddress ":"}]
The clumsy nesting of yet another key-value format within the existing key-value iCalendar format reflects the uneasy compromises vendors are making to maintain support for the standard. Meanwhile, CalDAV has become something of a second-class citizen in calendar applications. For example, ByteDance’s LarkSuite office suite restricts CalDAV as a read-only representation of its own calendar database.

While there are a few good reasons for transiting away from CalDAV, including security (CalDAV uses Basic Authentication), these moves undoubtedly threaten the interoperability of calendar services and relegates iCalendar to the lowest common denominator without fixing the underlying issues. Additionally, developers will struggle to compete with major vendors to support undocumented proprietary properties and API-only features.

Overall, developers should do more to strengthen existing standards on both the design and implementation side. Firstly, developers should resolve the problem of “one standard, many interpretations” by centralizing client and server libraries. With a common frame of reference, developers can avoid vulnerabilities slipping through the cracks. Microsoft does this well as Windows Calendar, Outlook, and Outlook web conform to a clearly documented iCalendar to Appointment Object Conversion Algorithm (MS-OXCICAL).16 However, developers have a long way to go to secure iCalendar implementations across the vast array of clients and platforms.

Next, developers should avoid unnecessary proprietary extensions to standards. For example, the use of custom parsing for DESCRIPTION and introducing X-ALT-DESC for rich text. These kludges create additional complexity and reduces compatibility.

By contributing to open-source iCalendar libraries, developers can secure the entire ecosystem instead of vendor-by-vendor implementations. In the meantime, security researchers will no doubt discover many bugs in inconsistent applications of the iCalendar standard.

16 https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-oxcical/a685a040-5b69-4c84-b084-795113f4012